Decentralized DNN Task Partitioning and Offloading Control in MEC Systems with Energy Harvesting Devices

Feng Wang, *Member, IEEE*, Songfu Cai, *Member, IEEE*, and Vincent K. N. Lau, *Fellow, IEEE*

***Abstract*—In this paper, we study a decentralized deep neural network (DNN) task partitioning and offloading control problem for a multi-access edge computing (MEC) system with multiple wireless devices (WDs) powered by renewable energy sources. Instead of generic computational tasks, we focus on the case when the WDs and the MEC systems are computing DNN inferencing tasks. For each WD, we build a virtual local-computing CPU workload queue and an energy queue to model the DNN computation dynamics and the battery energy dynamics, respectively. We first introduce a new Lyapunov function based on the DNN tasks' cumulative execution latency and the total energy consumptions of the entire MEC system. Using a Lyapunov optimization approach, the DNN task partitioning and offloading control at the WDs are formulated as a Lyapunov drift minimization problem. To facilitate a distributed implementation, we exploit the linear structure of the cumulative latency and the independency of each WD's local-computing CPU rate, and we decompose the centralized Lyapunov drift minimization problem into multiple distributed subproblems, each of which is associated with a single WD. Next, we develop a parametric online learning algorithm such that each sub-Lyapunov drift minimization problem is solved locally at its associated WD. The proposed solution is completely decentralized in the sense that the sequential offloading and DNN task segmentation control at each WD is determined by its local battery energy queue state information (EQSI), the channel state information (CSI), and the DNN inferencing task workload queue state information (TQSI). Numerical results reveal that the proposed online learning algorithm can achieve significant performance gain over various state-of-the-art baselines.**

***Index Terms*—Multi-access edge computing (MEC), energy harvesting, deep neural network (DNN) inferencing, computation offloading, Lyapunov optimization, parametric online learning.**

I. Introduction

Deep neural networks (DNNs) have been increasingly employed by various smart internet-of-things (IoT) applications (such as image recognition and robotics) for real-time inferencing [1], [2]. Due to the limited computation capability and battery capacity of individual wireless devices (WDs), it is quite challenging for these WDs to locally compute the real-time DNN inferencing tasks [3]–[5]. As a complement to conventional cloud computing technology, multi-access edge computing (MEC)[1] has been proposed to provide cloud-like computing abilities and information technology services at the network edge (such as cellular base stations and Wi-Fi access points) [6]–[11]. With MEC, the WDs can offload some DNN inferencing workload to the network edge for remote execution. The MEC system is expected to significantly augment the WDs' computation capability to support device-based DNN inferencing for future edge intelligence [12]–[18].

In some mission-critical applications, it is not feasible or is inconvenient to recharge or replace the battery of WDs on a regular basis. In this paper, we consider MEC systems with WDs powered by renewable energy sources (such as solar and wind) [19]–[23]. The presence of a renewable energy source brings some technical challenges to the design of offloading solutions for MEC systems. For example, the energy arrivals of renewable energy sources are random in nature, and a battery will be needed to buffer the intermittent energy arrivals for future consumption. As a result, the task offloading solution has to be adaptive not only to the fading wireless channel conditions but also to the instantaneous energy queue level. The energy queue dynamics has not been considered in the aforementioned MEC works [10]–[18]. In the literature, some works exist on task offloading designs for MEC systems with renewable sources [24]–[29]. The stochastic offloading and energy management algorithms based on Lyapunov optimization are developed for energy harvesting MEC systems with a single user [24] or multiple users [25]. The work in [26] proposes a reinforcement learning algorithm to learn a joint offloading and autoscaling policy for MEC systems with a renewable-powered base station. In [27], a reinforcement learning-based privacy-aware offloading scheme is proposed for an energy harvesting powered IoT device to process the healthcare sensing data. By modeling the dynamic computation offloading design problem as a Markov decision process (MDP), the work in [28] develops an after-state reinforcement learning algorithm to minimize the long-term task execution latency. Based on the stochastic dual-subgradient, the work in [29]

Manuscript received June 8, 2022; revised August 26, 2022; accepted November 9, 2022. This work was supported in part by the National Natural Science Foundation of China under Project 61901124, in part by the Natural Science Foundation of Guangdong Province under Project 2021A1515012305, in part by the Guangzhou Science and Technology Program under Project 202102020856, and in part by the Hong Kong Scholars Program under Project XJ2020044. The Guest Editor coordinating the review of this paper and approving it for publication was Dr. Marco Levorato. *(Corresponding author: Songfu Cai.)*

F. Wang is with the School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China, and is also with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: fengwang13@gdut.edu.cn).

S. Cai and V. K. N. Lau are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: scaiae@connect.ust.hk, eeknlau@ust.hk).

[1]Note that the term MEC and the more general edge computing is interchangeable in this paper.

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

presents a distributed online MEC resource allocation and load management solution, where the edge server is powered by both renewable energy and a smart grid. In another line of research, by incorporating wireless power transfer (WPT) technology into MEC systems, energy-efficient WPT energy allocation and computation offloading designs are investigated under different MEC system setups [30]–[38].

In the above works, however, the MEC offloading designs are targeted for generic computation tasks and hence, they fail to tailor to the unique properties of DNN inferencing tasks. For example, the DNN task is computed on a layer-by-layer basis with strong sequential dependencies [1]–[3]. While the generic task [24]–[29] is assumed to be completed on each time slot, the DNN task may span over multiple time slots. Due to these special properties for DNN inferencing tasks, the aforementioned works [24]–[35] cannot be directly applied to DNN offloading applications in MEC systems with renewable energy. In this paper, we consider an energy harvesting multiuser MEC system, where both the multilayer DNN inferencing tasks and renewable energy randomly arrive at the WDs over an infinite time horizon. Each WD is equipped with a finite-size battery to buffer the harvested renewable energy. The main contributions of this paper are summarized as follows.

- **Sequential DNN Task Segmentation and Offloading Control with Energy Availability Constraints:** Since a DNN task may span over multiple time slots, the decision includes the local-computing rate, the transmit power for offloading intermediate results, and the partitioning of each multilayer DNN task into a local portion and an uploading portion. At any time slot, each WD has to strike a balance between how much energy to consume (i.e., EQSI), the good offloading opportunities induced by the fading channel (i.e., CSI), and the instantaneous urgency induced by the DNN inferencing workload (i.e., TQSI). As such, we propose a Lyapunov optimization framework to address this challenge. Specifically, we first introduce a novel Lyapunov function, which is a function of the EQSI, CSI, and TQSI at each WD, for characterizing the DNN tasks' execution delay and energy consumption of the entire MEC system. Since a negative Lyapunov drift represents a stabilizing force to the task queues and the renewable energy queue, joint DNN task partitioning and offloading control design at the WDs is then formulated as a Lyapunov drift minimization problem.
- **Decentralized Solution via Lyaponov Drift Decomposition:** Due to the curse of dimensionality and the difficulty of collecting global EQSI/TQSI/CSI of all the WDs, brute-force solving the Lyapunov drift minimization problem is infeasible. By exploiting the additive structure of the DNN tasks' execution delay of the WDs and the decoupling nature of each WD's energy availability constraint, we decouple the centralized Lyapunov drift minimization problem into multiple lower dimensional subproblems, each of which is associated with a single WD. The resultant solution becomes completely decentralized in the sense that the DNN task segmentation and offloading control solution of each WD are determined only based on its local EQSI, TQSI, and CSI.
- **Parametric Online Learning and Convergence Analysis:** Although the proposed Lyapunov drift decomposition enables a fully decentralized solution, solving each of the decomposed subproblems is still very challenging. This is because the infinite time horizon DNN tasks' execution delay in the objective function of each optimization subproblem does not have a closed-form characterization. To address this challenge, we propose a novel parametric online learning algorithm, where each WD's DNN task execution delay is parameterized by a weighted sum of sigmoid functions. The weight parameters, as well as the DNN task segmentation and offloading control actions, are learned in an online manner based on the projected gradient descent (PGD) method. By exploiting the dynamic evolutions of each WD's DNN task execution delay under different offloading control actions, we further establish the convergence results of the proposed online learning algorithm.

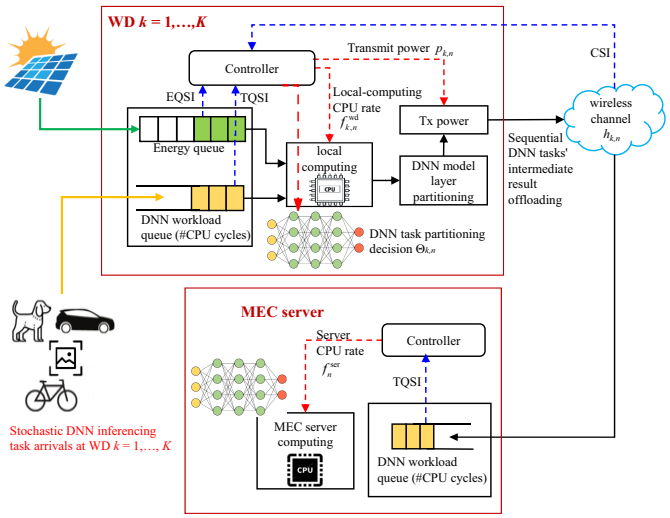


Fig. 1. System model of an energy harvesting multiuser MEC system with the stochastic DNN inferencing task arrivals and sequential offloading.

The rest of this paper is organized as follows. Section II introduces the energy harvesting multiuser MEC system model with dynamic DNN inferencing task arrivals. Using the Lyapunov optimization approach, Section III formulates the dynamic DNN task partitioning and offloading control problem as a Lyapunov drift minimization problem. The decentralized decomposition of the control problem is presented in Section IV, and the parametric online learning algorithm is developed in Section V. Section VI provides numerical results to evaluate the proposed solution, and Section VII concludes the paper. The main symbols and notations are summarized in Table I.

II. System Model

As illustrated in Fig. 1, we consider an energy harvesting MEC system, which consists of a number of K WDs and a base station (BS) collocated with an MEC server. These K WDs are capable of harvesting energy from renewable

TABLE I
SUMMARY OF THE MAIN SYMBOLS AND NOTATIONS

$\mathcal{K}$, k	Set and index of WDs	A_i	Number of task input-bits arrived at slot i
$\mathcal{N}$, n	Set and index of time slots.	B_k	System bandwidth for WD K's offloading
$\beta_{k,n}$	Indicator of WD k's DNN task arriving at slot n	$p_{k,n}$	Transmit power of WD k's offloading at slot n
Θ_k	Number of layers of WD k's DNN model	$\theta_{k,n}$	Layer index of WD k's DNN task at slot n
$w_{k,n}^{\text{front}}$	CPU workload of front-end DNN subtask of WD k at slot n	W_k	CPU workload (CPU-cycles) for one WD k's DNN task
$w_{k,n}^{\text{back}}$	CPU workload of back-end DNN subtask of WD k at slot n	$q_{k,n}^{\text{wd}}$	Length of WD k's CPU workload queue at slot n
$q_{k,n}^{\text{vir-wd}}$	Length of WD k's virtual CPU workload queue at slot n	ξ_k^{wd}	CPU's switched capacitance of WD k
q_n^{ser}	Length of MEC server's CPU workload queue at slot n	$\alpha_{k,n}$	Energy harvesting rate of WD k at slot n
$h_{k,n}$	Channel power gain for WD k's offloading at slot n	ξ_k^{wd}	Effective switched capacitance of WD k's CPU architecture
$f_{k,n}^{\text{wd}}$	CPU rate of WD k at slot n	f^{ser}	CPU rate of MEC server
$E_{k,n}$	Energy amount of WD k at slot n	σ^2	Power of the AWGN at AP
$\boldsymbol{s}_n$	Overall system state at slot n	$\boldsymbol{s}_{k,n}$	system state of WD k at slot n
$r_{k,n}$	Data rate for WD k's offloading at slot n	τ	Duration of each slot
$Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$	Expected discounted cumulative function	γ	Discount factor
$\boldsymbol{\phi}_{k,n}$, $\boldsymbol{\mu}_{k,n}$	Feature and parameter vectors of WD k at slot n	$c_{k,n}$	Immediate cost of WD k at slot n

resources (such as solar energy). Relying on harvested energy, each WD $k \in \mathcal{K} \triangleq \{1, ..., K\}$ is responsible for computing the randomly arrived DNN inferencing tasks via local computing and/or computation offloading. We consider that the well-trained multilayer DNN model is deployed at all the WDs and the MEC server. Each DNN task can be partitioned into two subsequential subtasks, where the first subtask is locally executed by the WD, and the computed intermediate result of the first subtask is offloaded to the BS, so that the MEC server therein can process the remaining subtask. In particular, each WD $k \in \mathcal{K}$ takes the battery energy queue state, the local CPU workload queue state of the DNN task arrivals, and the offloading channel conditions as input, and generates the DNN task partitioning decision, the local-computing CPU rate (i.e., the frequency of running CPU cycles), and the transmit power of offloading as output. We consider a slotted system, where each slot $n \in \mathcal{N} \triangleq \{1, 2, ...\}$ is with length τ.

A. DNN Inferencing Task and Partitioning Model

Let $\beta_{k,n} \in \{0, 1\}$ denote whether one DNN inferencing task of WD k arrives or not at the beginning of the n-th slot, where $\beta_{k,n} = 1$ represents that one DNN inferencing task arrives at WD $k \in \mathcal{K}$. The sequence $\{\beta_{k,n}\}$ is independent and identically distributed (i.i.d.) over the slots. It is assumed that $\beta_{k,n}$ follows a Bernoulli distribution with probability $b_k \in (0, 1)$, i.e., $\Pr(\beta_{k,n} = 1) = b_k$ and $\Pr(\beta_{k,n} = 0) = 1 - b_k$ for each slot $n \in \mathcal{N}$ and each WD $k \in \mathcal{K}$.

Let there be Θ_k layers in WD k's multilayer DNN model. At each slot $n \in \mathcal{N}$, the DNN task layer partitioning decision for each WD $k \in \mathcal{K}$ can be modeled as an integer variable $\theta_{k,n} \in \{-1, 0, 1, ..., \Theta_k\}$. In particular, when $\theta_{k,n} = -1$, the arrived DNN inferencing task of WD k is not to be executed but buffered at the WD local task queue. For $\theta_{k,n} \geq 0$, the layers $\{0, 1, ..., \theta_{k,n}\}$ of the DNN model (i.e., the front-end DNN subtask) are executed by WD k's local computing, and the remaining layers $\{\theta_{k,n} + 1, ..., \Theta_k\}$ of the DNN (i.e., the back-end DNN subtask) are uploaded to and executed at the MEC server. Due to the dependency of the DNN layers [1]–[3], the output of the front-end DNN subtask is the input of the back-end DNN subtask. This implies that WD k needs to offload the computed intermediate result of the front-end DNN subtask to the BS, so that the MEC therein can start to execute the back-end DNN subtask. Additionally, the cases of $\theta_{k,n} = 0$ and $\theta_{k,n} = \Theta_k$ correspond the cases that the DNN inferencing task is executed only by WD k's local computing or by the MEC server's remote computing.

Denote by W_k the CPU computation workload (in the unit of CPU cycles) required for completing one DNN inferencing task of WD $k \in \mathcal{K}$. With the DNN partitioning layer decision $\theta_{k,n} \geq 0$, let $w_{k,n}^{\text{front}} \in [0, W]$ and $w_{k,n}^{\text{back}} = W_k - w_{k,n}^{\text{front}}$ denote the CPU workload of the front-end DNN subtask and the back-end DNN subtask, respectively. Let $d_{k,n}^{\text{inter}}$ denote the size (i.e., in unit of bits) of intermediate result of the front-end DNN subtask. Then, we have a four-tuple $(\Theta_{k,n}, w_{k,n}^{\text{front}}, d_{k,n}^{\text{inter}}, w_{k,n}^{\text{end}})$ for each DNN inferencing task of WD k. For the multilayer DNN model of WD k, we consider that there exists a deterministic one-to-one mapping relationship between the WD k's DNN task partitioning decision $\theta_{k,n}$ and the CPU workload $w_{k,n}^{\text{front}}$, i.e.,

$$w_{k,n}^{\text{front}} = \mathcal{U}_k(\theta_{k,n}),\ \forall k \in \mathcal{K}, \tag{1}$$

where $\mathcal{U}_k(\cdot)$ is a predetermined one-to-one function. For the special case of without DNN task execution, we define $w_{k,n}^{\text{front}} = \mathcal{U}_k(-1) \triangleq 0$. Regarding the other two degenerated cases of $\theta_{k,n} = 0$ and $\theta_{k,n} = \Theta_k$, it follows that $w_{k,n}^{\text{front}} = \mathcal{U}_k(0) = \widetilde{W}_k$ and $w_{k,n}^{\text{front}} = \mathcal{U}_k(0) = W_k$, where $\widetilde{W}_k$ denotes the size of the raw-data (e.g., an image required to be classified) feeding to WD k's DNN model. Also, we use a function $\mathcal{D}_k(\cdot)$ to model the functional relationship between the workload $w_{k,n}^{\text{front}}$ of the front-end DNN subtask and its corresponding intermediate result size $d_{k,n}^{\text{inter}}$, i.e.,

$$d_{k,n}^{\text{inter}} = \mathcal{D}_k(W_k - w_{k,n}^{\text{front}}),\ \forall k \in \mathcal{K}, \tag{2}$$

where $\mathcal{D}_k(x)$ is a predetermined function with respect to $0 \leq x \leq W_k$ and is associated with WD k's DNN model.

B. DNN Task Workload Queuing at the WDs

As shown in Fig. 1, each WD $k \in \mathcal{K}$ maintains a local-computing CPU workload queue (TQSI) for the arrived but not yet executed DNN inferencing tasks. Let $q_{k,n}^{\text{wd}}$ denote the length (in the unit of CPU cycles) of WD k's local-computing CPU workload queue at the beginning of the n-th slot. Let $f_{k,n}^{\text{wd}}$

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

denotes the local-computing CPU rate of each WD $k \in \mathcal{K}$. Under the DNN layer partitioning decision $\theta_{k,n}$, WD k needs to locally complete the CPU workload $w_{k,n}^{\text{front}}$ of the DNN front-end subtask at slot n, which means that

$$\left\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \right\rceil W_k = w_{k,n}^{\text{front}} = \mathcal{U}_k(\theta_{k,n}),\ \forall k \in \mathcal{K}, \tag{3}$$

where $\lceil x \rceil$ denotes the ceiling function which maps x to the least integer no less than x. The equation chain (3) indicates the one-to-one mapping relationship between the local-computing CPU rate $f_{k,n}^{\text{wd}}$ and the DNN task partitioning decision $\theta_{k,n}$ for WD $k \in \mathcal{K}$ at each slot $n \in \mathcal{N}$.

With the DNN task layer partitioning decision $\theta_{k,n}$ and the local-computing CPU rate $f_{k,n}^{\text{wd}}$ at the n-th slot, the local-computing CPU workload queue dynamics of each WD $k \in \mathcal{K}$ evolves according to

$$q_{k,n+1}^{\text{wd}} = \left[q_{k,n}^{\text{wd}} - \left\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \right\rceil W_k \right]^+ + \beta_{k,n} W_k, \tag{4}$$

where $[x]^+ \triangleq \max(x, 0)$, the term $\left\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \right\rceil W_k$ denotes the WD k's local-computing CPU workload at the n-th slot, and $\beta_{k,n} W$ denotes WD k's new arrival workload of DNN tasks within the n-th slot. Considering the limited computational resources of WDs, each WD $k \in \mathcal{K}$ is assumed to locally complete at most one DNN task, i.e.,

$$0 \le \tau f_{k,n}^{\text{wd}} \le W_k, \tag{5}$$

which means $\left\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \right\rceil \in \{0, 1\}$. To avoid the case that there does not exist a sufficient amount of the CPU workload to be processed at each WD $k \in \mathcal{K}$, we impose an additional constraint for WD k's local-computing CPU workload such that $\left\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \right\rceil W_k \le q_{k,n}^{\text{wd}}$. Accordingly, we have the following constraints at the n-th slot as

$$\tau f_{k,n}^{\text{wd}} \le \left\lfloor \frac{q_{k,n}^{\text{wd}}}{W_k} \right\rfloor W_k,\ \forall k \in \mathcal{K}. \tag{6}$$

By combining the local-computing CPU rate constraints (5) and (6), the WDs' local-computing CPU rate constraints at the n-th constraints are given as

$$f_{k,n}^{\text{wd}} \le \frac{1}{\tau} \min\left(W_k, \left\lfloor \frac{q_{k,n}^{\text{wd}}}{W_k} \right\rfloor W_k \right),\ \forall k \in \mathcal{K}. \tag{7}$$

At the n-th slot, the amount of energy consumption due to the local computing of WD $k \in \mathcal{K}$ is expressed as [8], [9]

$$E_{k,n}^{\text{loc}} = \tau \xi_k^{\text{wd}} (f_{k,n}^{\text{wd}})^3, \tag{8}$$

where $\xi_k^{\text{wd}} > 0$ denotes the effective switched capacitance of the CPU architecture of WD k.

C. Computation Offloading from the WDs to the MEC Server

We consider that the intermediate output-data needs to be successfully offloaded within one slot from each WD $k \in \mathcal{K}$ to the MEC server. Let $h_{k,n} > 0$ denote the wireless offloading channel power gain from WD k to the BS, and let $p_{k,n} > 0$ denote the transmit power for WD k's task offloading at the slot just after the n-th local-computing slot. The data rate (bits/s) for task offloading from WD $k \in \mathcal{K}$ to the BS at the n-th slot is written as $r_{k,n} = B_k \log_2(1 + \frac{h_{k,n} p_{k,n}}{\Gamma \sigma^2})$, where $\Gamma \ge 1$ denotes the gap of the signal-to-noise ratio (SNR) due to the employed modulation and coding scheme [40], B_k denotes the spectrum bandwidth uniquely assigned to WD k's uplink task offloading, and σ^2 denotes the additive white Gaussian noise (AWGN) power at the BS receiver. Furthermore, we consider a quasi-static fading channel model, such that the channel power gain $h_{k,n}$ for task offloading remains constant within each slot, and it follows an i.i.d. exponential distribution over different slots with the large-scale pathloss as the mean value [40].

Note that at the end of the n-th slot, each WD $k \in \mathcal{K}$ generates the intermediate output data of one DNN inferencing task by locally completing the execution of the front-end part of a DNN task. Under the given local-computing CPU workload $\tau f_{k,n}^{\text{wd}}$ of each WD $k \in \mathcal{K}$, the output data size is given by $d_{k,n}^{\text{inter}} = \mathcal{D}_k(\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \rceil W_k - \tau f_{k,n}^{\text{wd}})$. Since the intermediate output data of one DNN inferencing task should be successfully offloaded to the MEC server within one slot, it follows that $r_{k,n}\tau \ge \mathcal{D}_k(\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \rceil W_k - \tau f_{k,n}^{\text{wd}})$. Therefore, the transmit power for WD k's computation offloading at the n-th slot is given by

$$p_{k,n} = \frac{1}{h_{k,n}} \mathcal{G}_k(f_{k,d}^{\text{wd}}), \tag{9}$$

where $\mathcal{G}_k(f_{k,n}^{\text{wd}}) \triangleq \Gamma \sigma^2 (2^{\frac{\mathcal{D}_k(\lceil \frac{\tau f_{k,n}^{\text{wd}}}{W_k} \rceil W_k - \tau f_{k,n}^{\text{wd}})}{\tau B_k}} - 1)$, $\forall k \in \mathcal{K}$.

Based on (9), the amount of WD k's transmit energy for computation offloading at the n-th slot is expressed as

$$E_{k.n}^{\text{off}} = \tau p_{k,n} = \frac{\tau}{h_{k,n}} \mathcal{G}_k(f_{k,n}^{\text{wd}}),\ \forall k \in \mathcal{K}. \tag{10}$$

Remark 1: Note that for an MEC system running a DNN inferencing task of WD $k \in \mathcal{K}$, the control action includes WD k's local-computing CPU rate $f_{k,n}^{\text{wd}}$, the transmit power $p_{k,n}$ for offloading the intermediate result with $d_{k,n}^{\text{inter}}$ bits (i.e., the output of the front-end DNN subtask), and the DNN task partitioning decision $\theta_{k,n}$. Due to the DNN layer structure and task computation causality, once each WD $k \in \mathcal{K}$ completes the front-end DNN subtask by its local-computing CPU rate $f_{k,n}^{\text{wd}}$, then WD k's DNN task partitioning decision $\theta_{k,n}$ and the transmit power $p_{k,n}$ for offloading the output of the front-end DNN subtask are readily determined as shown in (3) and (9), respectively. Under the local-computing CPU rate constraint in (7), the local-computing CPU rate $f_{k,n}^{\text{wd}}$ is one-to-one mapping to the DNN task partitioning layer decision $\theta_{k,n}$ by (3) and the transmit power $p_{k,n}$ for task offloading by (9). Therefore, the DNN task partitioning and offloading control can come down to the local-computing CPU rate control at the K WDs.

D. Renewable Energy Arrivals and Energy Queue Dynamics at the WDs

Consider that each WD is *solely* powered by the renewable energy harvesting process. Let $\alpha_{k,n}$ denote the random renewable energy harvesting rate (Joules/second) within the n-th decision slot at each WD $k \in \mathcal{K}$. We consider that the energy harvesting process $\{\alpha_{k,n}\}$ of each WD $k \in \mathcal{K}$ is

i.i.d. across different slots according to a general distribution $\Pr(\alpha_k)$ with an average energy harvesting rate $\mathbb{E}[\alpha_{k,n}] = \bar{\alpha}_k$. Furthermore, we assume that each WD $k \in \mathcal{K}$ is equipped with a finite-size battery to buffer the renewable energy arrivals. Let $E_{k,n} \in [0, E_k^{\max}]$ denote the energy amount of WD k at the beginning of the n-th slot, where $E_k^{\max}$ is the battery capacity of WD $k \in \mathcal{K}$. We consider that each WD $k \in \mathcal{K}$ *causally* knows its own current energy availability $E_{k,n}$ at the n-th slot, but the new energy arrival $\tau\alpha_{k,n}$ can only be available for WD k's use at the subsequent slots starting from the $(n+1)$-th slot. As a result, the renewable energy queue (EQSI) dynamics at WD $k \in \mathcal{K}$ evolves as

$$E_{k,n+1} = \min\Big\{E_k^{\max}, \Big[E_{k,n} - \tau\xi_k^{\text{wd}}(f_{k,n}^{\text{wd}})^3 - \frac{\tau}{h_{k,n}}\mathcal{G}_k(f_{k,n}^{\text{wd}})\Big]^+ + \tau\alpha_{k,n}\Big\}. \tag{11}$$

At the n-th slot, due to the energy demand-supply causality, the sum-energy amount for WD k's task offloading and local computing *cannot* be larger than the renewable energy amount $E_{k,n}$ currently stored at the battery. Accordingly, we have the *energy availability* constraints for the WDs at the n-th slot as

$$\tau\xi_k^{\text{wd}}(f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}}\mathcal{G}_k(f_{k,n}^{\text{wd}}) \le E_{k,n},\ \forall k \in \mathcal{K}. \tag{12}$$

As indicated by the renewable energy availability constraints in (12), the amount of available energy must be larger than what is consumed for each WD $k \in \mathcal{K}$ at each slot $n \in \mathcal{N}$. In addition, from the left-hand-side of (12), it follows that the energy consumption amount of WD k due to performing local-computing and offloading operations at the n-th slot is a function with respect to its local-computing CPU rate $f_{k,n}^{\text{wd}}$.

E. DNN Task Workload Queuing at the MEC Server

After the BS has received the intermediate output of the partitioned DNN inferencing task from the K WDs, the MEC server (collocated with the BS) starts to process the back-end DNN inferencing subtasks. The MEC sever maintains a CPU workload queue for the back-end DNN tasks. Let q_n^{ser} denote the length (in the unit of CPU cycles) of the CPU workload queue of the MEC server at the beginning of the n-th slot. Let f^{ser} denote the fixed CPU-cycle frequency assigned to execute the back-end parts of the WDs' DNN inferencing tasks at the n-th slot. The DNN inferencing computation workload queue dynamics at the MEC server is given by

$$q_{n+1}^{\text{ser}} = [q_n^{\text{ser}} - \tau f^{\text{ser}}]^+ + \sum_{k=1}^{K}\Big(\Big\lceil\frac{\tau f_{k,n}^{\text{wd}}}{W_k}\Big\rceil W_k - \tau f_{k,n}^{\text{wd}}\Big), \tag{13}$$

where $(\lceil\frac{\tau f_{k,n}^{\text{wd}}}{W_k}\rceil W_k - \tau f_{k,n}^{\text{wd}})$ represents the amount of WD k's new DNN inferencing computation workload arrival at the MEC server at the end of the n-th remote-computing slot.

III. Problem Formulation

In this section, by taking into account the stochastic DNN inferencing task arrivals and time-varying wireless channels for offloading, as well as the dynamic energy arrivals at the K WDs, we formulate the online offloading and DNN partitioning control as a Lyapunov drift minimization problem.

A. Transformation of DNN Inferencing Workload Queues

Note that the DNN inferencing workload queue $\{q_n^{\text{ser}}\}$ of the MEC server is deterministic in the sense that the MEC server's CPU rate f^{ser} is fixed and the DNN task workload $\sum_{k=1}^{K}(\lceil\frac{\tau f_{k,n}^{\text{wd}}}{W_k}\rceil W_k - \tau f_{k,n}^{\text{wd}})$ offloaded from the K WDs is fixed under the given WDs' offloading decisions. As such, we establish a virtual local-computing CPU workload queue for each WD k's DNN inferencing computation, denoted by $q_{k,n}^{\text{vir-wd}}$, which evolves according to

$$q_{k,n+1}^{\text{vir-wd}} = \Big[q_{k,n}^{\text{vir-wd}} - \tau f_{k,n}^{\text{wd}} - \frac{1}{K}\tau f^{\text{ser}}\Big]^+ + \beta_{k,n}W_k, \tag{14}$$

where $k \in \mathcal{K}$ and $n \in \mathcal{N}$. As shown in (14), WD k's virtual workload queue $q_{k,n}^{\text{vir-wd}}$ captures both the local-computing workload queue $q_{k,n}^{\text{wd}}$ and the MEC server's workload queue q_n^{ser}, and the MEC server fairly assigns the CPU rate $\frac{1}{K}f^{\text{ser}}$ to execute each WD's offloaded DNN inferencing workload.

B. MEC System State and Control Action

Let $\boldsymbol{s}_n = (\{q_{k,n}^{\text{vir-wd}}, h_{k,n}, E_{k,n}\}_{k=1}^K) \in \mathcal{S}$ denote the overall system state at the n-th slot, where $\mathcal{S}$ is the system state space. Denote by $\mathcal{A}$ the action space of the multiuser MEC system, which collects all the WDs' CPU rates. Let $\pi : \mathcal{S} \mapsto \mathcal{A}$ denote the control policy that makes local-computing CPU rate control decisions based on the state $\boldsymbol{s}_n$. Specifically, we let $\pi(\boldsymbol{s}_n) = \boldsymbol{f}_n^{\text{wd}} \in \mathcal{A}$ denote the control action taken when the system is in state $\boldsymbol{s}_n$ under a control policy π, where $\boldsymbol{f}_n^{\text{wd}} \triangleq [f_{1,n}^{\text{wd}}, ..., f_{K,n}^{\text{wd}}]^T$, and $f_{k,n}^{\text{wd}}$ denotes WD k's local-computing CPU rate. This action $\boldsymbol{f}_n^{\text{wd}}$ causes the system to transition to a new state $\boldsymbol{s}_{n+1}$. Note that both the state space and action space are continuous and dimensionally large.

The random state process $\{\boldsymbol{s}_n\}$ is a controlled Markov process [46]–[48] with the transition kernel given by

$$\Pr(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) = \prod_{k=1}^{K}\Pr(E_{k,n+1}|E_{k,n}, h_{k,n}, f_{k,n}^{\text{wd}}) \cdot \prod_{k=1}^{K}\Pr(h_{k,n+1})\prod_{k=1}^{K}\Pr(q_{k,n+1}^{\text{vir-wd}}|q_{k,n}^{\text{vir-wd}}, f_{k,n}^{\text{wd}}). \tag{15}$$

For WD $k \in \mathcal{K}$, the local-computing CPU workload queue transition probability $\Pr(q_{k,n+1}^{\text{wd}}|q_{k,n}^{\text{wd}}, f_{k,n}^{\text{wd}})$ is given by

$$\Pr(q_{k,n+1}^{\text{vir-wd}} = q_k^{\text{vir-wd}\prime}|q_{k,n}^{\text{vir-wd}} = q_k^{\text{vir-wd}}, f_{k,n}^{\text{wd}} = f_k^{\text{wd}}) = \begin{cases}\Pr(\beta_{k,n}), & \text{if } q_k^{\text{vir-wd}\prime} = [q_k^{\text{vir-wd}} - \tau f_k^{\text{wd}} - \frac{\tau}{K}f^{\text{ser}}]^+ + \beta_{k,n}W_k\\ 0, & \text{otherwise}\end{cases}$$

and the renewable energy queue transition probability $\Pr(E_{k,n+1}|h_{k,n}, E_{k,n}, f_{k,n}^{\text{wd}})$ is given by

$$\Pr(E_{k,n+1} = E_k'|E_{k,n} = E_k, h_{k,n} = h_k, f_{k,n}^{\text{wd}} = f_k^{\text{wd}}) = \begin{cases}\Pr(\alpha_{k,n}), & \text{if } E_k' = \min\Big\{E_k^{\max}, [E_k - \tau p_k \\ & \quad -\tau\xi_k^{\text{wd}}(f_k^{\text{wd}})^3 - \frac{\tau}{h_k}\mathcal{G}(f_k^{\text{wd}})]^+ + \tau\alpha_{k,n}\Big\}\\ 0, & \text{otherwise.}\end{cases}$$

C. Infinite Time Horizon DNN Task Execution Latency Cost

Denoted by $C: \boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd} \in \mathcal{S}\times\mathcal{A} \mapsto c_n = C(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) \in \mathbb{R}^+$ the bounded deterministic cost function. For each transition $(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}, \boldsymbol{s}_{n+1})$ at the n-th decisional slot, we define the current immediate cost function as

$$c_n = C(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) \triangleq \sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}}. \tag{16}$$

Note that the cost function c_n calculates the sum of the virtual local-computing CPU workload queue lengths of the K WDs' DNN inferencing computation at the n-th slot. Therefore, the cost function c_n can be used to characterize the multilayer DNN inferencing computation latency performance of the multiuser MEC system under consideration.

Denote by $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ the infinite time horizon discounted cumulative DNN tasks' execution delay, which starts in state $\boldsymbol{s}_n$, adopts the action $\boldsymbol{f}_n^{\rm wd}$, and thereafter follows the DNN task partitioning and offloading control policy π. Specifically, the infinite time horizon function $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ is expressed as the expected discounted cumulative cost. Hence, we have

$$Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) \triangleq \lim_{N\to\infty} \mathbb{E}^{\pi}\left\{ \sum_{i=0}^{N-1} \gamma^i c_{n+i} \Big| \boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd} \right\}, \tag{17}$$

where $0 < \gamma < 1$ denotes the discount factor weighting the long-term cost, and the expectation $\mathbb{E}\{\cdot\}$ in (17) is taken over different decision makings under different system states following the DNN task partitioning and offloading control policy π across the decisional slots.

D. Lyapunov Function and Lyapunov Drift for $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{wd})$

For each slot $n \in \mathcal{N}$, we choose the maximum battery size $E_k^{\max}$ as a *perturbation* parameter for each WD k's energy availability. Consequently, we define a Lyapunov function L_n for function $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ at the n-th slot as [20], [21], [39]

$$L_n \triangleq Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) + \frac{1}{2}\sum_{k=1}^{K} \underbrace{(E_{k,n} - E_k^{\max})^2}_{\text{energy perturbation of WD } k}, \tag{18}$$

where $\frac{1}{2}(E_{k,n} - E_k^{\max})^2$ denotes the deviation of each WD k's renewable energy queue length $E_{k,n}$ from the battery capacity $E_k^{\max}$. From (18), it follows that the renewable energy queue value $E_{k,n}$ is guaranteed to be pushed toward $E_k^{\max}$ by rendering the Lyapunov function value L_n to be small. As a result, we can ensure that each WD $k \in \mathcal{K}$ always has a sufficiently large amount of renewable energy at the battery for its local computing and offloading operations.

For (18), the associated *Lyapunov drift* conditioned on the current state-action pair $(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ at the n-th slot is given by

$$\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) = \mathbb{E}\Big\{ L_{n+1} - L_n \Big| \boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd} \Big\}, \tag{19}$$

where the expectation $\mathbb{E}\{\cdot\}$ in (19) is taken over the randomness of the DNN inferencing task arrivals, the offloading channel state, and the energy state conditioned on the state-action pair $(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ under the DNN task partitioning and offloading control policy π. Furthermore, we establish the following lemma on the Lyapunov drift (19).

Lemma 1 (Lyapunov Drift): Note that the Lyapunov drift $\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd})$ in (19) can be upper bounded as follows:

$$\begin{aligned}\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) \le \mathbb{E}\Big\{ \Big(\frac{1}{\gamma} - 1\Big) Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) - \frac{1}{\gamma}\sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}} \\ + \sum_{k=1}^{K} (E_k^{\max} - E_{k,n})\big(\tau \xi_k^{\rm wd} (f_{k,n}^{\rm wd})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\rm wd})\big) \\ + \sum_{k=1}^{K} (E_k^{\max} - \tau\alpha_{k,n})^2 \Big| \boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd} \Big\},\end{aligned} \tag{20}$$

where $\bar{\mathcal{G}}_k(f_{k,n}^{\rm wd}) \triangleq \Gamma\sigma^2(2^{\frac{\mathcal{D}_k(W_k - \tau f_{k,n}^{\rm wd})}{\tau B_k}} - 1)$, $\forall k \in \mathcal{K}$.

Proof: See Appendix A. ∎

E. Online DNN Task Partitioning and Offloading Control Problem

Note that the negative Lyapunov drift terms in (20) play a central role when applying the Lyapunov drift theory to analyze the stability of dynamic systems [39]. Intuitively, the negative Lyapunov drift is a stabilizing force that pulls the energy harvesting based multiuser MEC system back to the equilibrium point $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) = 0$ and $E_{k,n} = E_k^{\max}$, $\forall k \in \mathcal{K}$, which means that the DNN task execution delay is zero and the battery of each WD has stored maximum energy. As a result, the online DNN task partitioning and offloading control design can be formulated as a Lyapunov drift minimization problem, which is expressed as[2]

$$\begin{aligned}(\text{P1}):\ \min_{\boldsymbol{f}_n^{\rm wd}}\ & \Bigg[\Big(\frac{1}{\gamma} - 1\Big) Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\rm wd}) - \frac{1}{\gamma}\sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}} \\ & + \sum_{k=1}^{K} (E_k^{\max} - E_{k,n})\big(\tau \xi_k^{\rm wd} (f_{k,n}^{\rm wd})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\rm wd})\big) \\ & + \sum_{k=1}^{K} (E_k^{\max} - \tau\alpha_{k,n})^2 \Bigg] \qquad (21a) \\ \text{s.t.}\ & \tau \xi_k^{\rm wd} (f_{k,n}^{\rm wd})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\rm wd}) \le E_{k,n},\ \forall k \in \mathcal{K} \qquad (21b) \\ & 0 \le f_{k,n}^{\rm wd} \le \min \frac{1}{\tau}\Big(W_k, \Big\lfloor \frac{q_{k,n}^{\rm wd}}{W_k} \Big\rfloor W_k\Big),\ \forall k \in \mathcal{K} \qquad (21c) \\ & Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_k^{\rm wd}) \le \bar{D}_k,\ \forall k \in \mathcal{K}, \qquad (21d)\end{aligned}$$

where $Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_k^{\rm wd})$ denotes the infinite time horizon DNN task execution latency function associated with WD $k \in \mathcal{K}$, which starts in state $\boldsymbol{s}_{k,n}$, takes action $f_{k,n}^{\rm wd}$, and thereafter follows a local DNN task partitioning and offloading control policy π, and $\bar{D}_k$ denotes the prescribed DNN task execution delay requirement for WD k. Specifically, the per-WD DNN

[2]Note that by introducing extra optimization variables, the total MEC server's CPU rate constraint, and the total bandwidth constraints for offloading, our current work can be extended to the case of dynamic MEC server's CPU rate allocation and dynamic offloading bandwidth allocation, which is left for future investigation work.

task execution latency function $Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_k^{\text{wd}})$ is defined as

$$Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) \triangleq \lim_{N\to\infty} \mathbb{E}^{\pi}\Big\{ \sum_{i=0}^{N-1} \gamma^i c_{k,n+i} \Big| \boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}} \Big\}, \quad (22)$$

where $k \in \mathcal{K}$ and $n \in \mathcal{N}$.

Remark 2: For the Lyapunov drift minimization problem (P1), it is still difficult to obtain the optimal local-computing CPU rate vector $\boldsymbol{f}_n^{\text{wd}}$. The reasons are stated as follows.

- First, due to the lack of closed-form expression for the infinite time horizon DNN tasks' execution latency cost $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$, it is infeasible to directly evaluate the objective function of (P1). The MEC system state space $\mathcal{S}$ and control action $\{f_{k,n}^{\text{wd}}\}_{k=1}^K = \pi(\boldsymbol{s}_n)$ are both continuous, which further renders $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ difficult to be numerically computed.
- Second, as problem (P1) involves a total of K workload queues, the K WDs' channel conditions for task offloading, K task arrival flows, and K renewable energy queues), it is usually of high time-complexity to realize a centralized solution, especially in the case with a large number of WDs. Also, solving (P1) in a centralized manner involves a significant signaling overhead by collecting the global CSI $\{h_{k,n}\}$, the global EQSI $\{E_{k,n}\}$, and the global TQSI $\{q_{k,n}^{\text{wd}}\}$ of the K WDs at each slot $n \in \mathcal{N}$. This is not appropriate for timely decision making in various DNN interfering applications.
- Third, the WDs' local-computing action $\{f_{k,n}^{\text{wd}}\}_{k=1}^K$ at each slot must always satisfy the energy availability constraints (21b), so that the current local-computing action and the future actions are coupled, in the sense that a current local-computing decision may cause the outage of the renewable energy queue and hence affect some local-computing actions in the future. As such, the solution of (P1) needs to achieve a desirable balance between the WDs' energy demand-supply relationships, the task offloading opportunity induced by the wireless fading channel, and the urgency of completing the DNN inferencing computation.

To address the above challenges in Remark 2, we next pursue to obtain a decentralized solution for (P1) based on the equivalent decomposition and parametric online learning approach, such that each WD $k \in \mathcal{K}$ can make its own decision only based on the local TQSI/EQSI/CSI under the limited coordination of the MEC server.

IV. Decentralized Decomposition for (P1)

In this section, for facilitating the DNN task partitioning and offloading control designs, we proceed to decentralize problem (P1) into a total of K subproblems by decoupling the global state and action among the K WDs.

To start with, for each WD $k \in \mathcal{K}$, we define $\boldsymbol{s}_{k,n} \triangleq [q_{k,n}^{\text{vir-wd}}, h_{k,n}, E_{k,n}]^T \in \mathcal{S}_k$ and $c_{k,n} = C_k(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) \triangleq q_{k,n}^{\text{vir-wd}}$ as the state and immediate cost function of WD k, respectively, at the n-th decision slot, where $f_{k,n}^{\text{wd}} = \pi(s_{k,n}) \in \mathcal{A}_k$ denotes WD k's current action at the n-th slot under the DNN task partitioning and offloading control policy π, and $\mathcal{A}_k$ denotes the action space of WD k. The random state process $\{\boldsymbol{s}_{k,n} \in \mathcal{S}_k\}$ of each WD k is shown to be a controlled Markov process, where the transition kernel is given by

$$\Pr(\boldsymbol{s}_{k,n+1}|\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) = \Pr(h_{k,n+1})\Pr(q_{k,n+1}^{\text{vir-wd}}|q_{k,n}^{\text{vir-wd}}, f_{k,n}^{\text{wd}}) \cdot \Pr(E_{k,n+1}|E_{k,n}, h_{k,n}, f_{k,n}^{\text{wd}}). \quad (23)$$

Recalling the expressions for the transition kernel $\Pr(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ and the cost function $c_n = \sum_{k=1}^K q_{k,n}^{\text{vir-wd}}$ in (15) and (16), respectively, it follows that

$$c_n = \sum_{k=1}^K c_{k,n} \quad (24a)$$

$$\Pr(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) = \prod_{k=1}^K \Pr(\boldsymbol{s}_{k,n+1}|\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}). \quad (24b)$$

From (24), it yields that the controlled Markov process $\boldsymbol{s}_n$ can be equivalently characterized by the K controlled Markov subprocesses $\boldsymbol{s}_{1,n}, ..., \boldsymbol{s}_{K,n}$. As a result, the infinite time horizon DNN task execution latency function $Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ associated with $\boldsymbol{s}_n$ is decomposed as

$$Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) = \sum_{k=1}^K Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}), \quad (25)$$

where $Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ is only associated with WD k (cf. (22)).

By substituting (25) into the right-hand-side of (20), the Lyapunov drift $\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ can be further expressed as

$$\begin{aligned}\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) \le \sum_{k=1}^K \mathbb{E}\Big\{ \Big(\frac{1}{\gamma} - 1\Big) Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) - \frac{1}{\gamma} q_{k,n}^{\text{vir-wd}} \\ + (E_k^{\max} - E_{k,n})\Big(\tau \xi_k^{\text{wd}} (f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\text{wd}})\Big) \\ + (E_k^{\max} - \tau\alpha_{k,n})^2 \Big| \boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}} \Big\}. \quad (26)\end{aligned}$$

Building on (26) and the independency of renewable energy processes $\{E_{k,n}\}_{k=1}^K$ among the K WDs, we present the problem decentralization of (P1) in the following proposition.

Proposition 1 (Problem Decentralization of (P1)): For any given realization $(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$, the Lyapunov drift minimization problem (P1) is equivalently decentralized into a number of K subproblems, each of which is associated with one WD. Specifically, the subproblem associated with each WD $k \in \mathcal{K}$ is formulated as

$$\begin{aligned}(\text{P2}.k): \\ \min_{0 \le f_{k,n}^{\text{wd}} \le \bar{W}_k/\tau} \quad & \Big[\Big(\frac{1}{\gamma} - 1\Big) Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) \\ & + (E_k^{\max} - E_{k,n})\big(\tau \xi_k^{\text{wd}} (f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\text{wd}})\big) \Big] \\ \text{s.t.} \quad & \tau \xi_k^{\text{wd}} (f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\text{wd}}) \le E_{k,n} \\ & Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_k^{\text{wd}}) \le \bar{D}_k, \ \forall k \in \mathcal{K},\end{aligned}$$

where $\bar{W}_k \triangleq \min(W_k, \Big\lfloor \frac{q_{k,n}^{\text{wd}}}{W_k} \Big\rfloor W_k)$, and the irrelevant terms with respect to $f_{k,n}^{\text{wd}}$ in the right-hand-side of (26) is removed.

Remark 3: The problem decomposition of (P1) into a total of K subproblems is illustrated in Fig. 2. Each subproblem

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

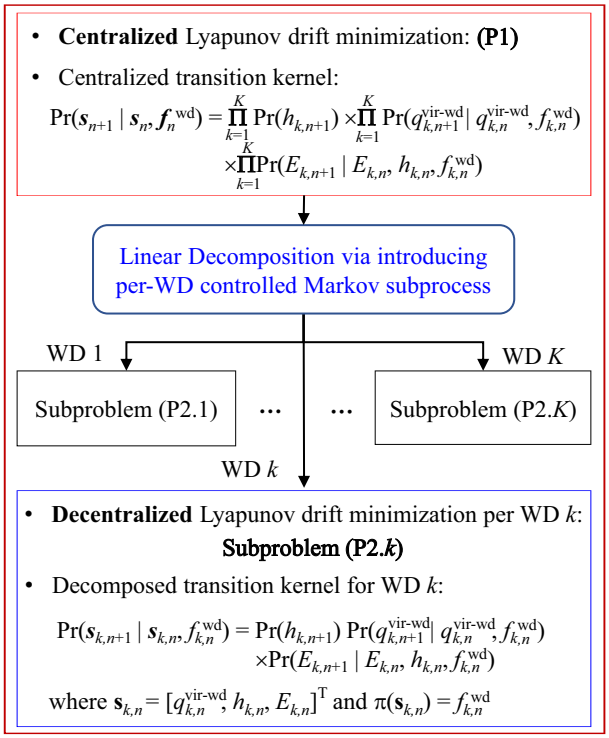


Fig. 2. The linear decomposition of problem (P1).

(P2.k) can be independently solved by WD $k \in \mathcal{K}$ without additional message exchanging, and without other WDs or the MEC server. Therefore, based on the proposed problem decomposition, the original Lyapunov drift minimization problem (P1) can be solved in a decentralized fashion by independently solving these K subproblems (P2.1),...,(P2.K). On the other hand, due to the continuity of the local state space $\mathcal{S}_k$ and action space $\mathcal{A}_k$ of WD k, as well as the lack of closed-form expression for the infinite time horizon DNN tasks' execution latency cost $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$, problem (P2.$k$) is highly challenging to solve. Alternatively, we next linearly approximate $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ by using a lower dimensional feature representation, and then we apply a parametric online learning method to solve (P2.k) for each WD $k \in \mathcal{K}$.

V. Parametric Online Learning Solution for (P1) with Function Approximation

In this section, we first explicitly approximate the objective function of each subproblem (P2.k) in a linear parametric form, and then present a parametric online learning solution for (P1) by individually solving subproblems (P2.1),...,(P2.K).

A. Function Approximation for Each Subproblem (P2.k)

For each WD $k \in \mathcal{K}$, we employ a policy set parameterized by $\boldsymbol{\mu}_{k,n} \triangleq [\mu_{k,1,n}, ..., \mu_{k,M,n}]^T \in \mathbb{R}^{M\times 1}$ to linearly approximate the infinite time horizon DNN task execution latency cost function $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ at the n-th slot [41]–[43]. For the purpose of characterizing the state-action pair $(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$, we need to define an M-dimensional feature vector $\boldsymbol{\phi}_{k,n} = [\phi_{k,1,n}, ..., \phi_{k,M,n}]^T \in \mathbb{R}^{M\times 1}$ for each WD $k \in \mathcal{K}$, where each element $\phi_{k,m,n}$ is a sigmoid function of $(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ for $m = 1, ..., M$. In particular, we have [41]–[43]

$$\phi_{k,m,n} \triangleq \frac{1}{1+\exp(-b_{m,1}^{k,n} q_{k,n}^{\text{vir-wd}} - b_{m,2}^{k,n} h_{k,n} - b_{m,3}^{k,n} E_{k,n} - b_{m,4}^{k,n} f_{k,n}^{\text{wd}})},$$

where $\{b_{m,1}^{k,n}, b_{m,2}^{k,n}, b_{m,3}^{k,n}, b_{m,4}^{k,n}\}$ are the adaptive real-valued weight parameters associated with the CPU workload queue length $q_{k,n}^{\text{vir-wd}}$, the offloading channel power gain $h_{k,n}$, the renewable energy queue length $E_{k,n}$, and the control action $f_{k,n}^{\text{wd}}$ of WD k at the n-th slot, respectively.

With the feature vector $\boldsymbol{\phi}_{k,n}$ and parameter vector $\boldsymbol{\mu}_{k,n}$, the parametric approximation of $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ is given by [42]

$$Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) \approx \boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n}, \tag{28}$$

where $\boldsymbol{\mu}_{k,n}$ is the bounded parameter vector that satisfies $\|\boldsymbol{\mu}_{k,n}\| \le J_k$, $\forall k \in \mathcal{K}$, and J_k is a positive constant. For the parametric approximation, the per-WD DNN task execution latency constraint $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}}) \le \bar{D}_k$ in (21d) is reduced to

$$\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n} \le \bar{D}_k, \tag{29}$$

where $k \in \mathcal{K}$. Based on Cauchy-Schwartz inequality, it follows that $\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n} \le \|\boldsymbol{\mu}_{k,n}^T\| \|\boldsymbol{\phi}_{k,n}\| \le M\|\boldsymbol{\mu}_{k,n}\|$, where the second inequality holds from the fact that each element of $\boldsymbol{\phi}_{k,n}$ is smaller than one (i.e., $\phi_{k,m,n} \le 1$, $\forall m = 1, ..., M$). By letting the upper bound of $\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n}$ satisfy the latency constraint, we have $M\|\boldsymbol{\mu}_{k,n}\| \le \bar{D}_k$, $\forall k \in \mathcal{K}$. By combining $M\|\boldsymbol{\mu}_{k,n}\| \le \bar{D}_k$ with $\|\boldsymbol{\mu}_{k,n}\| \le J_k$, the parameter vector $\boldsymbol{\mu}_{k,n}$ is thus bounded as $\|\boldsymbol{\mu}_{k,n}\| \le \bar{D}_k/M$, which equivalently characterizes WD k's DNN task execution latency.

The parametric approximation performance of $\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n}$ with respect to $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ is numerically evaluated in Fig. 3, where the simulation parameters are set as those in Section VI. It is observed that the parametric approximation (28) achieves increasingly close to the original performance of $Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})$ as M increases. Theoretically, based on the universal approximation property of sigmoidal functions, the approximation error of (28) is of order $1/\sqrt{M}$ [44].

B. Online Learning Solution of Subproblem (P2.k)

By replacing $Q_k^\pi(\boldsymbol{s}_n, f_{k,n}^{\text{wd}})$ with $\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n}$, the objective function of problem (P2.k) can be represented as

$$\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) \triangleq \left(\frac{1}{\gamma} - 1\right)\boldsymbol{\mu}_{k,n}^T \boldsymbol{\phi}_{k,n} + (E_k^{\max} - E_{k,n})\left(\tau \xi_k^{\text{wd}} (f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}} \bar{\mathcal{G}}_k(f_{k,n}^{\text{wd}})\right), \tag{30}$$

where $\boldsymbol{\eta}_{k,n} \triangleq [f_{k,n}^{\text{wd}}, \boldsymbol{\mu}_{k,n}^T]^T \in \mathbb{R}^{(M+1)\times 1}$.

Based on (30), problem (P2.k) can be approximately solved by minimizing the parameterized Lyapunov drift $\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$ subject to the WD k's energy availability constraint and the norm-bounded constraint for the parameter vector, in which we jointly optimize the local-computing CPU rate $f_{k,n}^{\text{wd}}$ and

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

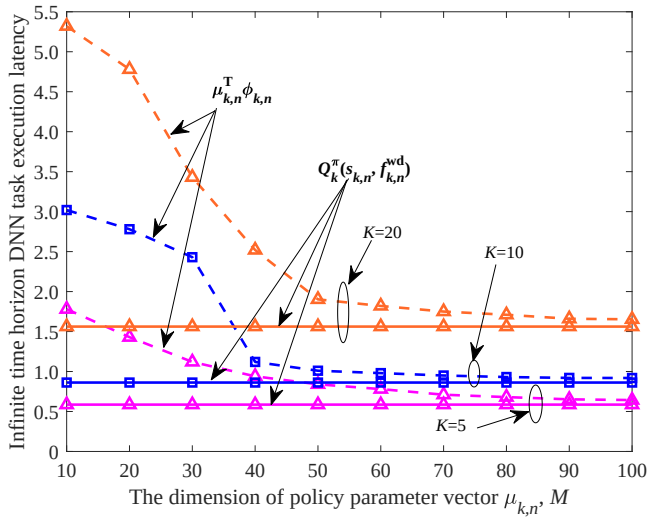


Fig. 3. The performance gap between the linear parametric approximation $\boldsymbol{\mu}_{k,n}^T\boldsymbol{\phi}_{k,n}$ and the original $Q_k^{\pi}(\boldsymbol{s}_{k,n}, f_{k,n}^{\mathrm{wd}})$ versus the dimension M of the policy parameter vector $\boldsymbol{\mu}_{k,n}$, where $k \in \mathcal{K}$.

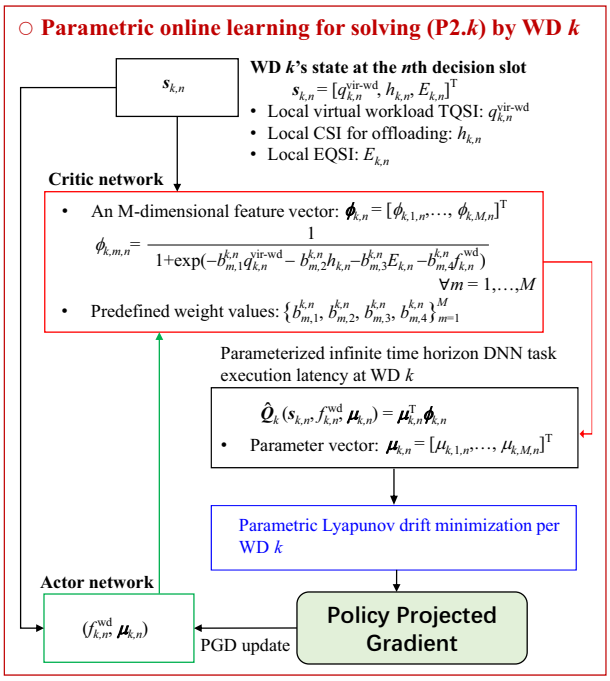


Fig. 4. The algorithm flow of the proposed parametric online learning for solving (P2.k) by WD k, where $k \in \mathcal{K}$.

the parameter vector $\boldsymbol{\mu}_{k,n}$. This is formally summarized as follows:

$$\boldsymbol{\eta}_{k,n}^* = \arg\min_{\boldsymbol{\eta}_{k,n}} \quad \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) \tag{31a}$$

$$\text{s.t. } 0 \le f_{k,n}^{\mathrm{wd}} \le \bar{W}_k/\tau \tag{31b}$$

$$\tau\xi_k^{\mathrm{wd}}(f_{k,n}^{\mathrm{wd}})^3 + \frac{\tau}{h_{k,n}}\bar{\mathcal{G}}_k(f_{k,n}^{\mathrm{wd}}) \le E_{k,n} \tag{31c}$$

$$\|\boldsymbol{\mu}_{k,n}\| \le \bar{D}_k/M. \tag{31d}$$

It is worth noting that due to the coupling of $\boldsymbol{\mu}_{k,n}$ and $f_{k,n}^{\mathrm{wd}}$ in the term $\boldsymbol{\mu}_{k,n}^T\boldsymbol{\phi}_{k,n}$, problem (31) is non-convex [45], and it is thus challenging to obtain the global optimal solution. Alternatively, by leveraging the projected gradient descent (PGD) method [46], we pursue to obtain a local optimal solution (i.e., a stationary point) of (31) with low complexity.

For notational convenience, we denote the feasible solution set of (31) as set $\mathcal{X}_{k,n}$ for each slot n, which is defined as

$$\mathcal{X}_{k,n} \triangleq \Big\{\boldsymbol{\eta}_{k,n} = [f_{k,n}^{\mathrm{wd}}, \boldsymbol{\mu}_{k,n}^T]^T \;\Big|\; (31\text{b}), (31\text{c}), (31\text{d})\Big\}. \tag{32}$$

It is shown that $\mathcal{X}_{k,n}$ is a convex set of $\boldsymbol{\eta}_{k,n}$ [45]. Based on the PGD method [46], the design vector $\boldsymbol{\eta}_{k,n}$ is updated over slots according to

$$\boldsymbol{\eta}_{k,n+1} = \mathcal{P}_{\mathcal{X}_{k,n+1}}\Big\{\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})\Big\}, \tag{33}$$

where $\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) \in \mathbb{R}^{(M+1)\times 1}$ denotes the gradient vector of function $\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$ with respect to $\boldsymbol{\eta}_{k,n}$, $\lambda_k > 0$ denotes the constant learning rate of WD k at each slot n, and $\mathcal{P}_{\mathcal{X}_{k,n+1}}\{\boldsymbol{x}\}$ denotes the projection operation of projecting $\boldsymbol{x}$ into the nearest point in set $\mathcal{X}_{k,n+1}$. The gradient vector $\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$ is explicitly obtained as $\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) =$

$$\Big[\frac{\partial\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial f_{k,n}^{\mathrm{wd}}}, \frac{\partial\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial\phi_{k,1,n}}, ..., \frac{\partial\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial\phi_{k,M,n}}\Big]^T, \text{ where}$$

$$\frac{\partial\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial\mu_{k,m,n}} = \Big(\frac{1}{\gamma} - 1\Big)\phi_{k,m,n}, \ \forall m = 1, ..., M \tag{34a}$$

$$\frac{\partial\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial f_{k,n}^{\mathrm{wd}}} = (E_k^{\max} - E_{k,n})\Big(3\tau(f_{k,n}^{\mathrm{wd}})^2 + \frac{\tau}{h_{k,n}}\bar{\mathcal{G}}_k'(f_{k,n}^{\mathrm{wd}})\Big) + (\frac{1}{\gamma} - 1)\sum_{m=1}^{M}\mu_{k,m,n}\phi_{k,m,n}(1-\phi_{k,m,n})b_{m,4}^{k,n}, \tag{34b}$$

with $\bar{\mathcal{G}}_{k,n}'(f_{k,n}^{\mathrm{wd}}) = -\frac{\Gamma\sigma^2(\ln 2)\mathcal{D}_k'(W_k - \tau f_{k,n})}{B_k}2^{\frac{\mathcal{D}_k(W_k-\tau f_{k,n}^{\mathrm{wd}})}{\tau B_k}}$ being the first-order derivation of $\bar{\mathcal{G}}_{k,n}'(f_{k,n}^{\mathrm{wd}})$ with respect to $f_{k,n}^{\mathrm{wd}}$, and $\mathcal{D}_k'(x)$ denotes the first-order derivation of function $\mathcal{D}_k(x)$ with respect to x. The algorithm flow of the proposed online learning scheme is illustrated in Fig. 4.

C. Proposed Online Learning Algorithm for (P1) and Convergence Analysis

We summarize the decentralized parametric actor-critic online learning method for obtaining a stationary point of problem (P1) in Algorithm 1, where each WD $k \in \mathcal{K}$ is responsible for solving subproblem (P2.k), and these subproblems (P2.1),...,(P2.K) are solved in a parallel manner.

Note that the standard gradient descent methods without projection usually converge to a stationary point, where the corresponding gradient must be zero [45], [46]. Due to the projection operator (33) employed in our case, the gradient at the convergent point of our Algorithm 1 may not be zero. However, the convergence point should be invariant to the projection with one-step gradient descent. The convergence of the proposed online learning scheme is characterized in the following proposition.

TABLE II
ALGORITHM 1: PARAMETRIC LEARNING APPROACH FOR SOLVING (P1)

a) **Initialization:** Set $n = 0$, $\epsilon_0 > 0$; start the system with initial state $\boldsymbol{s}_0 = (\{q_{k,0}^{\text{vir-wd}}, h_{k,0}, E_{k,0}\}_{k=1}^{K})$; for each WD $k \in \mathcal{K}$, the local-computing CPU rate $f_{k,0}^{\text{wd}}$, the policy parameter vector $\boldsymbol{\mu}_k^0$, and the learning rate $\lambda_{k,0}$ are initiated; we set $n = 0$.
b) **for** each WD $k = 1, ..., K$ **do**
c) **while** $\frac{|\mathcal{F}_{k,n+1}(\boldsymbol{\eta}_{k,n+1}) - \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})|}{\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})} > \epsilon_0$ **do**
- Compute the partial gradient $\frac{\partial \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial \mu_{k,m,n}}$ by (34a) for $i = 1, ..., M$;
- Compute the partial gradient $\frac{\partial \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})}{\partial f_{k,n}^{\text{wd}}}$ by (34b);
- Update $\boldsymbol{\eta}_{k,n}$ by the projected gradient descent process (33);
- Set $n \leftarrow n + 1$;

d) **end while**
e) **end for**
f) **Output:** The local-computing CPU rate solution of problem (P1).

Proposition 2 (Convergence Analysis): For each $k \in \mathcal{K}$, let the constant learning rate λ_k of the project gradient descent process (33) be sufficiently small such that $0 < \lambda_k \leq \frac{2}{\beta_k}$, where β_k is determined by the largest spectrum norm of the Hessian matrix $\nabla^2 \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) \in \mathbb{R}^{(M+1)\times(M+1)}$ of function $\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$ for $\boldsymbol{\eta}_{k,n} \in \mathcal{X}_{k,n+1}$, i.e.,

$$\beta_k = \max_{\boldsymbol{\eta}_{k,n} \in \mathcal{X}_{k,n+1}} \|\nabla^2 \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})\|_2, \tag{35}$$

where $\|\boldsymbol{A}\|_2$ denotes the spectrum norm of matrix $\boldsymbol{A}$. Then, by letting the discount factor $\gamma \to 1$ and the slot index $n \to \infty$, the solution $\boldsymbol{\eta}_{k,n}$ obtained by the proposed Algorithm 1 is guaranteed to converge almost surely to a stationary point $\boldsymbol{\eta}_k^*$ of (P2.k), where $\boldsymbol{\eta}_k^*$ must satisfy the fixed-point equation given by

$$\boldsymbol{\eta}_k^* = \lim_{n \to \infty} \mathcal{P}_{\mathcal{X}_{k,n+1}} \Big\{ \boldsymbol{\eta}_k^* - \lambda_k \nabla g_{k,n}(\boldsymbol{\eta}_k^*) \Big\}. \tag{36}$$

Proof: See Appendix B. ∎

D. Computational Complexity Analysis

In this subsection, we briefly discuss the computational complexity analysis for the proposed online learning Algorithm 1 for (P1). Specifically, for each iteration of Algorithm 1, the computational complexity of computing the partial gradient (34a), the partial gradient (34b), and the Euclidean projection (33) is about $\mathcal{O}(1)$, $\mathcal{O}(M)$, and $\mathcal{O}(M \log(M))$, respectively, where M is the number of features of each WD's DNN task execution latency cost [43]. Therefore, the total computational complexity for one iteration of the proposed Algorithm 1 is about $\mathcal{O}(M \log(M) + M + 1)$. Furthermore, note that the approximation error of each WD's DNN task execution latency function (28) is of order $1/\sqrt{M}$ [44]. As a result, we can choose an appropriate parameter M to strike a balance between the computational complexity of the proposed Algorithm 1 and the approximation accuracy of DNN task execution latency, such that the proposed Algorithm 1 can be implemented in a (near) real-time manner.

VI. NUMERICAL RESULTS

In this section, we evaluate the proposed distributed parametric online learning solution for DNN task partitioning and offloading control in energy harvesting MEC systems. In the simulations, we consider a practical multilayer DNN inferencing task of image classification [3], where the DNN has $(10^3 + 1)$ hidden layers and each hidden layer contains a number of 3×10^3 neurons. The input layer of the DNN contains 10^4 nodes, and the DNN has 10 output nodes to classify the input images into 10 categories. Each image consists of 100×100 pixels, and each pixel is represented by 24 bits. Based on the Inter 64 IA-32 architectures, we assume each addition operation costs 1 CPU cycles, while each multiplication operation needs 4 CPU cycles. The computational cost between the input layer and the first hidden layer of the DNN is around 1.5×10^8 CPU cycles, the computational cost in the DNN hidden layers is about 4.5×10^{10} CPU cycles, and the computational cost between the last hidden layer and the output layer of the DNN is about 5×10^4 CPU cycles. Hence, the workload of each WD k's DNN task is set to consist of $W_k = W = 10^{11}$ CPU cycles, and the raw-data size of each WD k's DNN task is set as $\widetilde{W}_k = \widetilde{W} = 10^8$ bits. We consider that the renewable energy harvesting rate $\alpha_{k,n}$ of WD k is uniformly distributed between 0 and $2\bar{\alpha}_k$, and the layer number of the DNN model deployed at WD k is set as $\Theta_k = 32$, $\forall k \in \mathcal{K}$. For WDs' task offloading channel in the uplink, the channel power gain $h_{k,n}$ is exponentially distributed with mean $G_0 d_k^{-4}$, where $G_0 = -43$ dB is the pathloss constant at a reference distance of 1 m, and d_k is the distance from WD k to the BS [40]. The K WDs are randomly and uniformly distributed in a cell with a radius of 500 m. The noise power at the BS receiver is set to be 10^{-13} W. The time slot length is set to be $\tau = 0.2$ seconds unless otherwise specified. The effective capacitance of the CPU architecture for WD $k \in \mathcal{K}$ is set as $\xi_k^{\text{wd}} = 10^{-28}$ Joules, and the MEC server's CPU rate is set as $f^{\text{ser}} = 5$ GHz. For each WD $k \in \mathcal{K}$, the spectrum bandwidth for task offloading is set as $B_k = 1.5$ MHz and the SNR gap is set as $\Gamma = 1.4$ [40]. The K WDs are considered to have the same average energy arrival rate and DNN task arrival probability, as well as the same battery size, i.e., $\bar{\alpha}_k = \bar{\alpha}$, $b_k = b$, $E_k^{\max} = E^{\max}$, $\forall k \in \mathcal{K}$. The error tolerance is set as $\epsilon_0 = 10^{-3}$ and the parameter dimension is set as $M = 50$ for linearly approximating the action-value function in Algorithm 1. The simulation is based on Monte Carlo runs, where a total of 5000 randomized realizations are averaged for the given average energy arrival $\bar{\alpha}$, the task arrival probability b, and the offloading channel configurations.

The proposed parametric online learning scheme is compared with the following five baseline schemes.

- *Baseline 1 (Random Scheme):* In this scheme, each WD $k \in \mathcal{K}$ offloads a random amount of the partitioned task workload to the MEC server, which corresponds to a random DNN task partitioning policy.
- *Baseline 2 (Binary Offloading Scheme without DNN Task Partitioning):* In this scheme, the DNN inferencing task can only be locally executed by the WD or be fully offloaded to the MEC server.
- *Baseline 3 (Myopic Optimization Scheme):* In this scheme, the temporal correlation between the system states and the decisions is ignored, and the cost function

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

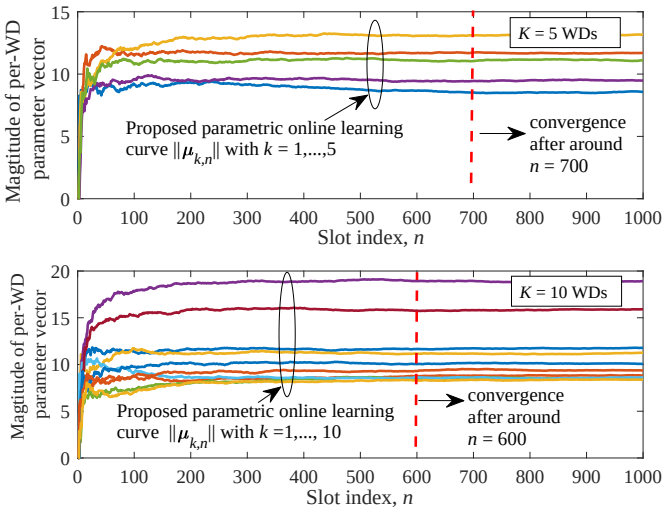


Fig. 5. The convergence performance of the proposed parametric online learning Algorithm 1.

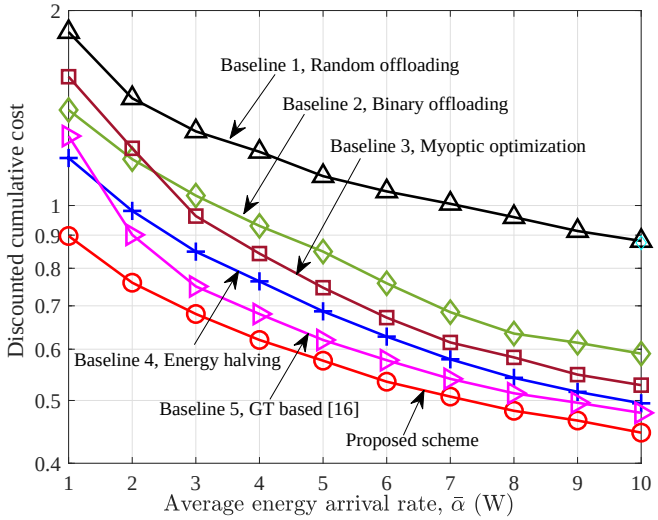


Fig. 6. The discounted cumulative cost versus average energy arrival rate $\bar{\alpha}$.

at the current slot is minimized by utilizing all the available battery energy.

- *Baseline 4 (Energy Halving Scheme):* In this scheme, each WD $k \in \mathcal{K}$ utilizes half of the energy determined by the myopic optimization scheme. This heuristic scheme intends to conserve the renewable energy in the battery.
- *Baseline 5 (Dynamic Scheme based on Aggregative Game Theory (GT) [16]):* In this scheme, a dynamic pricing strategy is used by the MEC server to motivate WDs towards task offloading, and an aggregative game theory (GT) based algorithm is employed to jointly optimize DNN task partitioning and offloading designs [16].

A. Online Learning Convergence Performance

The convergence performance of the proposed parametric online learning Algorithm 1 for the DNN task partitioning and offloading control is shown in Fig. 5, where the average energy arrival rate is $\alpha = 4$ W, the DNN task arrival probability is $b = 0.3$, and the battery size is $E^{\max} = 35$ J for all the WDs. It is observed that Algorithm 1 requires around 700 and 600 iterations to converge into a desirable solution under the cases with $K = 5$ and $K = 10$ WDs, respectively. This is because a larger number of WDs implies a larger number of degrees of freedom in adjusting the WDs' local computing and offloading policies, so as to better adapting to the fluctuation of dynamic task/energy arrivals and time-varying wireless offloading channel gains.

B. DNN Task Execution Latency versus Energy Arrival Rate $\bar{\alpha}$

Fig. 6 shows the long-term discounted cumulative cost versus the average energy arrival rate $\bar{\alpha}$, where $K = 20$, $b = 0.4$, and $E^{\max} = 30$ J. It is observed that the discounted cumulative cost of all the schemes decreases as the average energy arrival rate $\bar{\alpha}$ increases. This is expected, since a larger value of $\bar{\alpha}$ implies more energy is available for each WD to execute the DNN inferencing computation via local computing and/or offloading. The proposed scheme is observed to achieve a significant performance gain over all the baseline schemes, due to the ability of being fully dynamic to adapt to the TQSI/EQSI and offloading channel conditions. In Fig. 6, the baseline energy halving scheme is observed to outperform the baseline myopic optimization scheme, and the performance gain becomes substantial when the average energy arrival rate $\bar{\alpha}$ is small (e.g., $\bar{\alpha} < 4$). This shows the importance of reserving the battery energy for reducing the long-term cumulative cost. The baseline binary offloading scheme is observed to perform inferiorly to the baseline myopic optimization scheme in the case with a large $\bar{\alpha}$ value (e.g., $\bar{\alpha} \geq 3$). Due to lack of optimization for offloading designs, the baseline random offloading scheme performs inferiorly to other schemes. The baseline GT based scheme is observed to outperform the other baseline schemes, which implies the importance of dynamic DNN partitioning and offloading designs in reducing DNN execution latency.

C. DNN Task Execution Latency versus Battery Capacity $E^{\max}$

Fig. 7 illustrates the discounted cumulative cost performance versus the WD battery capacity $E^{\max}$, where $K = 20$, $\bar{\alpha} = 4$ W, and $b = 0.6$. Observe that the cumulative cost of all the schemes decreases with the increasing of the WD battery capacity. This is expected, since a larger battery capacity can better adapts to the dynamic state of TQSI/EQSI and the fluctuations of offloading channel power gain. Compared to the five baseline schemes, the proposed scheme has a large performance gain. As the battery capacity increases, the performance gain achieved by the baseline energy halving scheme over the baseline myopic optimization scheme decreases. This is because a larger battery capacity can better help alleviate the fluctuation range of future task/energy arrivals in the myopic optimization scheme for local computing and offloading designs. Interestingly, with the ability of adapting

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

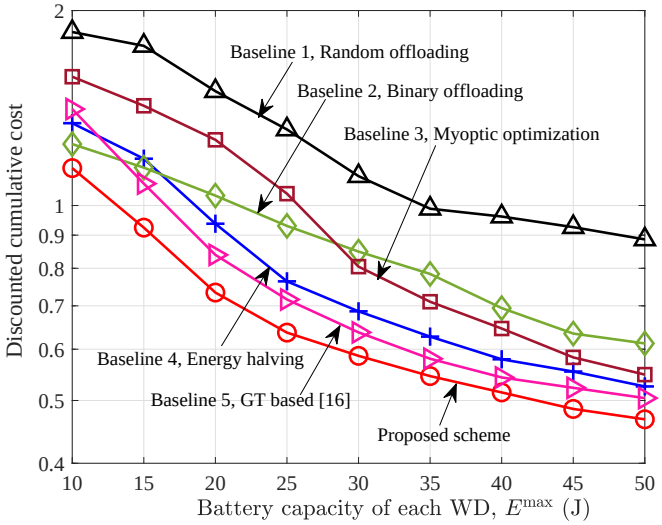


Fig. 7. The discounted cumulative cost versus WD battery capacity $E^{\max}$.

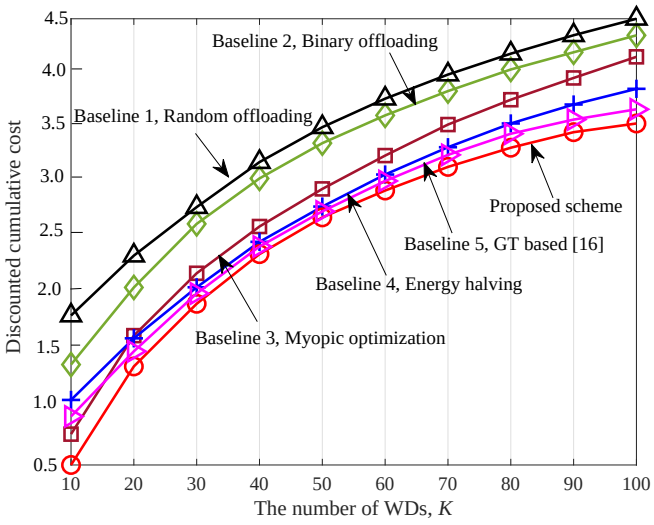


Fig. 9. The discounted cumulative cost versus the number of WDs K.

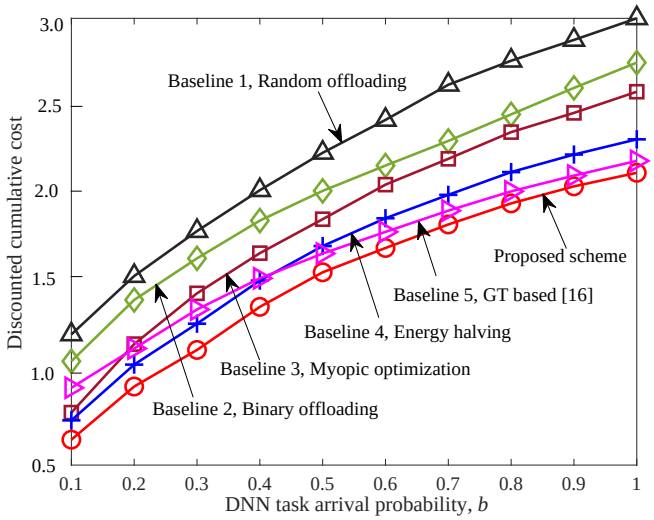


Fig. 8. The discounted cumulative cost versus the DNN task arrival probability b at the WDs.

to the system state, the baseline binary offloading scheme is observed to outperform the other three baseline schemes in the case with a small WD battery capacity, but this is not so as the battery capacity increases. This indicates the importance of simultaneously exploiting both local and remote (i.e., MEC server) computational resources in the case of a large WD battery capacity. Again, the baseline GT based scheme is observed to outperform the other baseline schemes, and the baseline random offloading scheme performs inferiorly to the other schemes.

D. DNN Task Execution Latency versus DNN Task Arrival Probability b

Fig. 8 shows the discounted cumulative cost performance versus the DNN task arrival probability b, where $K = 30$, $\bar{\alpha} = 4$ W, and $b = 0.6$. As the task arrival probability b increases, the cumulative cost of all the schemes increases. The proposed scheme is observed to achieve a large performance gain over the five baseline schemes. The baseline GT based scheme is observed to outperform the other four baseline schemes as b increases. The baseline energy halving scheme outperforms the baseline myopic optimization performance, and the performance gain becomes larger as b increases. This indicates the necessity of reserving a certain amount of energy in the battery in order to handle the bursty task arrivals in the future, especially in the case with a large task arrival probability. Also, due to lack of simultaneously leveraging the local computing and offloading capabilities, it is observed that the baseline binary offloading scheme performs inferiorly to the baseline myopic optimization scheme. Again, the baseline random offloading scheme is observed to perform inferiorly to the other schemes in this setup.

E. DNN Task Execution Latency versus Number of WDs K

Fig. 9 indicates the discounted cumulative cost versus the number of WDs K, where $\bar{\alpha} = 5$ W, $b = 0.6$, and $E^{\max} = 30$ W. As K increases, the discounted sum cost of all the schemes is observed to increase. It is observed that the proposed scheme outperforms the five baseline schemes. The GT based baseline outperforms the other four baseline schemes as K increases. Additionally, the baseline energy halving scheme performs inferiorly to the baseline myopic optimization scheme with a small K value (e.g., $K \leq 20$), but it is observed to outperform the baseline myopic optimization scheme when K becomes larger in this setup. This further indicates the necessity to reserve energy at the WD battery in sequential computing and offloading designs with a large number of WDs. As in Fig. 8, the baseline binary offloading scheme outperforms the baseline random offloading scheme, but it performs inferiorly to the other four schemes.

VII. Conclusion

In this paper, we investigated the decentralized DNN task partitioning and offloading control designs for a multiuser MEC system with renewable energy. We introduced TQSI and EQSI to capture the dynamic urgency of the DNN inferencing tasks and the dynamic energy availability of the battery at each WD, respectively. We first proposed a novel Lyapunov function that captures the DNN task execution delay and energy cost of the entire MEC system. The online optimization of joint DNN task segmentation and offloading control of the WDs was then formulated as a Lyapunov drift minimization problem. To reduce the computational complexity and facilitate a fully decentralized solution, we decomposed the original Lyapunov drift minimization problem into multiple low-dimensional subproblems. Each subproblem aims to achieve a balance between minimizing the long-term discounted cumulative cost and stabilizing the battery energy for one associated WD. Furthermore, we developed the parametric online learning algorithm to individually solve each subproblem by the associated WD, where the convergence is theoretically guaranteed. Numerical results show that the proposed online decentralized solution has a substantial performance gain over the various existing design schemes. As a potential research direction to capture the coupling of multiuser DNN partitioning and offloading in MEC systems, by allowing multiple WDs to dynamically share the total communication/computation resource, the adaptive offloading bandwidth allocation and dynamic MEC server's computing power allocation strategies are worthy of further investigation.

Appendix

A. Proof of Lemma 1

Note that the considered Lyapunov drift $\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ in (19) can be re-expressed as

$$\Delta(\boldsymbol{s}_n, \boldsymbol{f}_{n+1}^{\text{wd}}) = \mathbb{E}\Big\{Q^{\pi}(\boldsymbol{s}_{n+1}, \boldsymbol{f}_{n+1}^{\text{wd}})\Big|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}\Big\} - Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) + \sum_{k=1}^{K}\frac{1}{2}\Big((E_{k,n+1}-E_k^{\max})^2 - (E_{k,n}-E_k^{\max})^2\Big). \quad (37)$$

In the following, we discuss the upper bounds for the two terms in (37).

First, for the infinite-time horizon task execution delay cost function, we have [48]

$$\begin{aligned}&\mathbb{E}\Big\{Q^{\pi}(\boldsymbol{s}_{n+1}, \boldsymbol{f}_{n+1}^{\text{wd}})|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}\Big\}\\ &= \frac{1}{\gamma}Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) - \frac{1}{\gamma}\mathbb{E}\Big\{C(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}, \boldsymbol{s}_{n+1})|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}\Big\} \quad (38a)\\ &= \frac{1}{\gamma}Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) - \frac{1}{\gamma}\sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}}. \quad (38b)\end{aligned}$$

Accordingly, it follows from (38) that

$$\begin{aligned}&\mathbb{E}\Big\{Q^{\pi}(\boldsymbol{s}_{n+1}, \boldsymbol{f}_{n+1}^{\text{wd}})|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}\Big\} - Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})\\ &\quad= \Big(\frac{1}{\gamma}-1\Big)Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) - \frac{1}{\gamma}\sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}}. \quad (39)\end{aligned}$$

Next, we define $\varphi_{k,n} \triangleq E_{k,n} - E_k^{\max}$, $\forall k \in \mathcal{K}$. Subtracting $E_k^{\max}$ on both sides of the renewable queue dynamics in (11), it follows that

$$\begin{aligned}\varphi_{k,n+1} &= \min\Big\{0, [E_{k,n} - z(f_{k,n}^{\text{wd}})]^+ + \tau\alpha_{k,n} - E_k^{\max}\Big\}\\ &= \min\Big\{0, \max\{\varphi_{k,n} + \tau a_{k,n} - z(f_{k,n}^{\text{wd}}), \tau\alpha_{k,n} - E_k^{\max}\}\Big\}, \quad (40)\end{aligned}$$

where $z(f_{k,n}^{\text{wd}}) \triangleq \tau\xi_k^{\text{wd}}(f_{k,n}^{\text{wd}})^3 + \frac{\tau}{h_{k,n}}\bar{\mathcal{G}}_k(f_{k,n}^{\text{wd}})$. By squaring both sides of (40), we obtain the inequality chain as

$$\begin{aligned}&\varphi_{k,n+1}^2\\ &\le (\max\{\varphi_{k,n} - (z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n}), \tau\alpha_{k,n} - E_k^{\max}\})^2 \quad (41a)\\ &\le (\varphi_{k,n} - (z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n}))^2 + (\tau\alpha_{k,n} - E_k^{\max}\})^2 \quad (41b)\\ &= \varphi_{k,n}^2 - 2\varphi_{k,n}(z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})\\ &\quad + (z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})^2 + (E_k^{\max} - \tau\alpha_{k,n}\})^2, \quad (41c)\end{aligned}$$

where the inequality of (41a) holds from the fact that $(\min\{0, x\})^2 \le x^2$, and the inequality of (41b) holds from the fact that $(\max\{x, y\})^2 \le x^2 + y^2$. Based on (41), it further follows that

$$\begin{aligned}&\varphi_{k,n+1}^2 - \varphi_{k,n}^2 \le -2\varphi_{k,n}(\mathcal{F}_{k,n}(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})^2\\ &\qquad + (\mathcal{F}_{k,n}(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})^2 + (E_k^{\max} - \tau\alpha_{k,n}\})^2 \quad (42a)\\ &= 2(E_k^{\max} - E_{k,n})(z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})\\ &\qquad + (z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n})^2 + (E_k^{\max} - \tau\alpha_{k,n}\})^2 \quad (42b)\\ &\le 2(E_k^{\max} - E_{k,n})z(f_{k,n}^{\text{wd}}) + 2(E_k^{\max} - \tau\alpha_{k,n})^2, \quad (42c)\end{aligned}$$

where the right-hand-side of (42c) is obtained by replacing $z(f_{k,n}^{\text{wd}}) - \tau\alpha_{k,n}$ with $z(f_{k,n}^{\text{wd}})$ in the first term and by replacing $z(f_{k,n}^{\text{wd}})$ with $\tilde{E}_{k,n}$ in the second and third terms for (42b). The equality of (42c) holds from the inequalities $E_{k,n} \le E_k^{\max}$ and $\tau\alpha_{k,n} \ge 0$, as well as the energy availability constraint that $0 \le z(f_{k,n}^{\text{wd}}) \le E_{k,n}$ at each slot n.

Therefore, based on (42), for each $k \in \mathcal{K}$, we are ready to obtain the following upper bound:

$$\begin{aligned}&\frac{1}{2}(E_{k,n+1} - E_k^{\max})^2 - \frac{1}{2}(E_{k,n} - E_k^{\max})^2\\ &\qquad \le (E_k^{\max} - E_{k,n})z(f_{k,n}^{\text{wd}}) + (E_k^{\max} - \tau a_{k,n})^2. \quad (43)\end{aligned}$$

Now, by combining (39), (43), and the definition of $z(f_{k,n}^{\text{wd}})$, the Lyapunov drift $\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}})$ is upper bounded by

$$\begin{aligned}\Delta(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) \le &\mathbb{E}\Big\{\Big(\frac{1}{\gamma}-1\Big)Q^{\pi}(\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}) - \frac{1}{\gamma}\sum_{k=1}^{K} q_{k,n}^{\text{vir-wd}}\\ &+ \sum_{k=1}^{K}(E_k^{\max} - E_{k,n})z(f_{k,n}^{\text{wd}})\\ &+ \sum_{k=1}^{K}(E_k^{\max} - \tau a_{k,n})^2\Big|\boldsymbol{s}_n, \boldsymbol{f}_n^{\text{wd}}\Big\}, \quad (44)\end{aligned}$$

which completes the proof of Lemma 1.

This article has been accepted for publication in IEEE Journal of Selected Topics in Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3221850

B. Proof of Proposition 2

First, we present an upper bound for the function difference $\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n+1}) - \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$. Using the Taylor expansion around point $\boldsymbol{\eta}_{k,n}$ [46], we have

$$\begin{aligned}
&\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n+1}) - \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) = \nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \\
&\quad + \frac{1}{2}(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n})^T\nabla^2\mathcal{F}_{k,n}(\tilde{\boldsymbol{\eta}}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \quad (45a)\\
&\leq \nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \\
&\quad + \frac{1}{2}\|\nabla^2\mathcal{F}_{k,n}(\tilde{\boldsymbol{\eta}}_{k,n})\|_2 \cdot \|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2 \quad (45b)\\
&\leq \nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) + \frac{\beta_k}{2}\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2, \quad (45c)
\end{aligned}$$

where $\tilde{\boldsymbol{\eta}}_{k,n} \in \mathcal{X}_{k,n+1}$. The first inequality (45b) holds from the Cauchy-Schwartz inequality, and (45c) holds from the fact that $\nabla^2\mathcal{F}(\tilde{\boldsymbol{\eta}}_{k,n}) \preceq \beta_k\boldsymbol{I}$, where β_k is the largest spectrum norm of the Hessian matrix $\nabla^2\mathcal{F}(\boldsymbol{\eta}_{k,n})$ for $\boldsymbol{\eta}_{k,n} \in \mathcal{X}_{k,n+1}$ (c.f. (35)).

Based on the PGD process (33), we have

$$\begin{aligned}
\boldsymbol{\eta}_{k,n+1} &= \mathcal{P}_{\mathcal{X}_{k,n+1}}\Big\{\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})\Big\} \\
&= \underset{\boldsymbol{\eta}\in\mathcal{X}_{k,n+1}}{\arg\min}\ \|\boldsymbol{\eta} - (\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}))\|^2, \quad (46)
\end{aligned}$$

which implies that

$$(\boldsymbol{\eta}_{k,n+1} - (\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})))^T(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \leq 0. \quad (47)$$

Note that by substituting $\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n} = \boldsymbol{\eta}_{k,n+1} - (\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})) - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$, we have

$$\begin{aligned}
&\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2 = -\lambda_k\nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \\
&+ \Big(\boldsymbol{\eta}_{k,n+1} - \big(\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})\big)\Big)^T(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}). \quad (48)
\end{aligned}$$

As a result of (47) and (48), we are ready to have

$$\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2 \leq -\lambda_k\nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}),$$

which implies the inner-product $\nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n})$ is upper bounded as

$$\nabla\mathcal{F}_{k,n}^T(\boldsymbol{\eta}_{k,n})(\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}) \leq -\frac{1}{\lambda_k}\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2. \quad (49)$$

By substituting (49) into (45b), we have

$$\begin{aligned}
&\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n+1}) - \mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n}) \\
&\qquad \leq \Big(\frac{\beta_k}{2} - \frac{1}{\lambda_k}\Big)\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2. \quad (50)
\end{aligned}$$

Furthermore, from (50) and the definition of function $\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})$, it follows that

$$\begin{aligned}
&\mathbb{E}[Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n+1}^{\text{wd}}) - Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})] \\
&\qquad \leq \frac{\gamma}{1-\gamma}\Big(\frac{\beta_k}{2} - \frac{1}{\lambda}\Big)\mathbb{E}\Big[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2\Big] + \epsilon, \quad (51)
\end{aligned}$$

where $\epsilon > 0$ is a bounded positive constant. Based on the Bellman equation [48], we have

$$\mathbb{E}\Big[Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n+1}^{\text{wd}}) - \gamma Q_k^\pi(\boldsymbol{s}_{k,n+1}, f_{k,n+1}^{\text{wd}})\Big] = \mathbb{E}[c_n]. \quad (52)$$

By letting (51) subtract (52) and leveraging the fact that $\mathbb{E}[c_n] \geq 0$, we arrive at

$$\begin{aligned}
&\mathbb{E}\Big[\gamma Q_k^\pi(\boldsymbol{s}_{k,n+1}, f_{k,n+1}^{\text{wd}}) - Q_k^\pi(\boldsymbol{s}_{k,n}, f_{k,n}^{\text{wd}})\Big] \\
&\qquad \leq \frac{\gamma}{1-\gamma}\Big(\frac{\beta_k}{2} - \frac{1}{\lambda}\Big)\mathbb{E}\Big[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2\Big] + \epsilon. \quad (53)
\end{aligned}$$

Similarly, based on (53), we have a series of inequalities for $i = 1, ..., N-1$, where

$$\begin{aligned}
&\mathbb{E}[\gamma^i Q_k^\pi(\boldsymbol{s}_{k,n+i}, f_{k,n+i}^{\text{wd}}) - \gamma^{i-1}Q_k^\pi(\boldsymbol{s}_{k,n+i-1}, f_{k,n+i-1}^{\text{wd}})] \\
&\leq \frac{\gamma^i}{1-\gamma}(\frac{\beta_k}{2} - \frac{1}{\lambda})\mathbb{E}\big[\|\boldsymbol{\eta}_{k,n+i} - \boldsymbol{\eta}_{k,n+i-1}\|^2\big] + \gamma^{i-1}\epsilon. \quad (54)
\end{aligned}$$

By summing the equalities (54) of both sides from $i = 1$ to $i = N-1$ and setting $n = 1$, it follows that

$$\begin{aligned}
-Q_k^\pi(\boldsymbol{s}_{k,1}, f_{k,1}^{\text{wd}}) \leq &\sum_{i=1}^{N-1}\frac{\gamma^i}{1-\gamma}(\frac{\beta_k}{2} - \frac{1}{\lambda_k})\mathbb{E}\big[\|\boldsymbol{\eta}_{k,i+1} - \boldsymbol{\eta}_{k,i}\|^2\big] \\
&+ \sum_{i=1}^{N-1}\gamma^{i-1}\epsilon. \quad (55)
\end{aligned}$$

With some algebraic manipulations for (55), we have

$$\begin{aligned}
&\sum_{n=1}^{N-1}\frac{\gamma^n}{1-\gamma}\Big(\frac{1}{\lambda_k} - \frac{\beta_k}{2}\Big)\mathbb{E}\Big[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2\Big] \\
&\qquad \leq \frac{1}{1-\gamma}\epsilon + Q_k^\pi(\boldsymbol{s}_{k,1}, f_{k,1}^{\text{wd}}). \quad (56)
\end{aligned}$$

Next, we prove $\lim_{\gamma\to1,n\to\infty}\mathbb{E}[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2] = 0$ by contradiction. Suppose that $\mathbb{E}[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2]$ is bounded away from zero, i.e., there exists a constant $\rho > 0$ satisfying $\mathbb{E}[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2] > \rho$, $\forall n$. We have

$$\sum_{n=1}^{N-1}\frac{\gamma^n}{1-\gamma}\Big(\frac{1}{\lambda_k} - \frac{\beta_k}{2}\Big)\rho \leq \frac{\epsilon}{1-\gamma} + Q_k^\pi(\boldsymbol{s}_{k,1}, f_{k,1}^{\text{wd}}). \quad (57)$$

From (57), it follows that

$$\rho \leq \frac{\epsilon + (1-\gamma)Q_k^\pi(\boldsymbol{s}_{k,1}, f_{k,1}^{\text{wd}})}{\sum_{n=1}^{N-1}\gamma^n(\frac{1}{\lambda_k} - \frac{\beta_k}{2})}. \quad (58)$$

Based on (58), by letting $\gamma \to 1$, we have

$$\begin{aligned}
\rho &\leq \lim_{\gamma\to1,N\to\infty}\ \frac{\epsilon + (1-\gamma)Q_k^\pi(\boldsymbol{s}_{k,1}, f_{k,1}^{\text{wd}})}{\sum_{n=1}^{N-1}\gamma^n(\frac{1}{\lambda_k} - \frac{\beta_k}{2})} \quad (59a)\\
&= \lim_{\gamma\to1,N\to\infty}\frac{\epsilon}{\sum_{n=1}^{N-1}\gamma^n(\frac{1}{\lambda_k} - \frac{\beta_k}{2})} \quad (59b)\\
&= \lim_{\gamma\to1}\frac{\epsilon(1-\gamma)}{\gamma(\frac{1}{\lambda_k} - \frac{\beta_k}{2})} = 0, \quad (59c)
\end{aligned}$$

which contradicts the assumption $\rho > 0$. Therefore, it must hold that

$$\lim_{\gamma\to1,n\to\infty}\mathbb{E}\Big[\|\boldsymbol{\eta}_{k,n+1} - \boldsymbol{\eta}_{k,n}\|^2\Big] = 0. \quad (60)$$

As a result of (60), by considering the projection $\boldsymbol{\eta}_{k,n+1} = \mathcal{P}_{\mathcal{X}_{k,n+1}}\big\{\boldsymbol{\eta}_{k,n} - \lambda_k\nabla\mathcal{F}_{k,n}(\boldsymbol{\eta}_{k,n})\big\}$, it follows that the sequence of $\{\boldsymbol{\eta}_{k,n}\}$ obtained by the proposed Algorithm 1 is guaranteed to converge almost surely to a certain $\boldsymbol{\eta}_k^*$, which satisfies the fixed-point equation of $\boldsymbol{\eta}_k^* = \lim_{n\to\infty}\mathcal{P}_{\mathcal{X}_{k,n+1}}\big\{\boldsymbol{\eta}_k^* - \lambda_k\nabla\mathcal{F}_{k,n+1}(\boldsymbol{\eta}_{k,n})\big\}$. Now, the proof of Proposition 2 is completed.

REFERENCES

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[2] D. Roberts, S. Yaida, and B. Hanin, *Principles of Deep Learning Theory*, 2021. [Online]. Available: https://arxiv.org/abs/2106.10165

[3] S. Dorner, S. Cammerer, J. Hoydis, and S. Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[4] L. Qian, P. Yang, M. Xiao, O. Dobre, M. Renzo, J. Li, Z. Han, Q. Yi, and J. Zhao, "Distributed learning for wireless communications: Methods, applications and challenges," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 326–342, Apr, 2022.

[5] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, pp. 487–500, Apr. 2022.

[6] "MEC in 5G networks," ETSI, Sophia Antipolis, France, White Paper, No. 28, Jun. 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[7] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud archtecture and orchestation," *IEEE Commun. Surveys Tuts.,* vol. 19, no. 3, pp. 1657–1681, 3rd Quar. 2017.

[8] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.,* vol. 19, no. 3, pp. 2322–2358, 3rd Quar. 2017.

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.,* vol. 19, no. 4, pp. 2322–2358, 4th Quar. 2017.

[10] F. Wang, J. Xu, and Z. Ding, "Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[11] F. Wang and V. K. N. Lau, "Multi-level over-the-air aggregation of mobile edge computing over D2D wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8337–8353, Oct. 2022.

[12] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last miles of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[13] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[14] T. Mohammed, C. Joe-Wong, R. Babbar, and M. D. Francesco, "Distributed inference acceleration with adaptive DNN partitioning and offloading," in *Proc. IEEE INFOCOM*, Toronto, Canada, Jul. 2020.

[15] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint DNN partition deployment and resource allocation for delay-sensitive deep learning inference in IoT," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9241–9254, Oct. 2020.

[16] M. Gao, R. Shen, L. Shi, W. Qi, J. Li, and Y. Li, "Task partitioning and offloading in DNN-task enabled mobile edge computing networks," To appear in *IEEE Trans. Mobile Comp.*, 2021.

[17] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.

[18] F. Wang, S. Cai, and V. K. N. Lau, "Sequential offloading for distributed DNN computation in multiuser MEC systems," 2022. [Online]. Available: https://arxiv.org/abs/2203.01005

[19] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 360–381, Mar. 2015.

[20] L. Huang and M. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1117–1130, Aug. 2013.

[21] S. Cai and V. K. N. Lau, "MIMO precoding for networked control systems with energy harvesting sensors," *IEEE Trans. Signal Process.*, vol. 64, no. 17, pp. 4469–4478, Sep. 2016.

[22] Z. Wei, X. Yu, D. W. K. Ng, and R. Schober, "Resource allocation for simultaneous wireless information and power transfer systems: A tutorial overview," *Proc. IEEE*, vol. 110, no. 1, pp. 127–149, Jan. 2022.

[23] B. Guler and A. Yener, "Energy-harvesting distributed machine learning," in *Proc. IEEE ISIT*, Melbourne, Australia, 2021.

[24] 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.

[25] G. Zhang, Y. Chen, Z. Shen, and L. Wang, "Distributed energy management for multiuser mobile-edge computing systems with energy harvesting devices and QoS constraints," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4035–4048, Jun. 2019.

[26] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cog. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[27] M. Min, X. Wan, L. Xiao, T. Chen, M. Xia, D. Wu, and H. Dai, "Learning-based privacy-aware offloading for healthcare IoT with energy harvesting," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4307–4316, Jun. 2019.

[28] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: A learning method," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4436–4447, Jun. 2019.

[29] X. Chen, H. Wen, W. Ni, S. Zhang, X. Wang, S. Xu, and Q. Pei, "Distributed online optimization of edge computing with mixed power supply of renewable energy and smart grid," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 389–403, Jan. 2022.

[30] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

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

[32] S. Bi and Y. J. A. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

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

[34] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.

[35] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[36] D. Wu, F. Wang, X. Cao, and J. Xu, "Wireless powered user cooperative computation in mobile edge computing," in *Proc. IEEE Globecom Workshops*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–7.

[37] C. Campolo, G. Lia, M. Amadeo, G. Ruggeri, A. Iera, and A. Molinaro, "Towards named AI networking: Unveiling the potential of NDN for edge AI," in *Proc. ADHOC-NOW*, Bari, Italy, Oct. 2020, pp. 16–22.

[38] S. Kazmi, M. Iqbal, and S. Coleri, "Total transmission time minimization through relay selection for full-duplex wireless powered cooperative communication networks," in *Proc. ADHOC-NOW*, Bari, Italy, Oct. 2020, pp. 257–268.

[39] M. Neely, "Stochastic network optimization with application to communication and queueing system," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[40] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[41] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforecement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, Nov. 1999, pp. 1057–1063.

[42] M. Geist and O. Pietquin, "An algorithmic survey of parametric value function approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 845–867, Jun. 2013.

[43] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, Apr. 2003.

[44] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[45] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, U.K.: Cambridge Univ. Press, 2004.

[46] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vols. I and II*. Boston, MA: Athena Scientific, 2005.

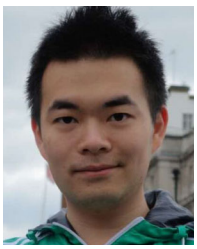
[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[48] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2nd, MIT Press, 2018.

Feng Wang (Member, IEEE) received the Ph.D. degree from Fudan University, Shanghai, China, in 2016.

From 2012 to 2013, he was a Research Fellow with the Sharp Laboratories of China, Shanghai, China. In 2017, he was a Post-Doctoral Research Fellow with the Engineering Systems and Design Pillar, Singapore University of Technology and Design, Singapore. From 2021 to 2022, he was a Hong Kong Scholar Fellow with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. He is currently an Associate Professor with the School of Information Engineering, Guangdong University of Technology, Guangzhou, China. His research interests include signal processing for wireless communications, mobile edge computing and intelligence, and applications of optimization algorithms. He received the Examplary Reviewer for the IEEE Wireless Communications Letters in 2020.

He has currently served as a member of the Technical Program Committees for several IEEE conferences and as a reviewer for several IEEE journals.



Songfu Cai (Member, IEEE) received the Ph.D. degree in electronic and computer engineering (ECE) from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2019.

He is currently a Postdoctoral Research Fellow with the Department of ECE, HKUST. His research interests include wireless communication, Industrial Internet of Things, learning-driven radio resource management, and networked control systems. He was the recipient of the Hong Kong Ph.D. Fellowship in 2013.

Vincent K. N. Lau (Fellow, IEEE) obtained B.Eng (Distinction 1st Hons–ranked 2nd) from the Department of EEE, University of Hong Kong in 1992 and PhD from the University of Cambridge in 1997.

He joined the Bell Labs–Lucent Technologies, New Jersey in 1997 as Member of Technical Staff. He joined the Department of ECE, HKUST in August 2004 and is currently a Chair Professor. Vincent has published over 300 papers and contributed to 50+ US patents on wireless systems. He is a Fellow of IEEE, IET and HKIE, Changjiang Chair Professor and the Croucher Senior Research Fellow. His research interests include Massive MIMO Systems, Federated Learning, Mission Critical IoT Systems, Stochastic Optimization Algorithms.