Energy-Efficient Resource Management for Federated Edge Learning with CPU-GPU Heterogeneous Computing

Qunsong Zeng, Graduate Student Member, IEEE, Yuqing Du, Student Member, IEEE, Kaibin Huang, Fellow, IEEE, and Kin K. Leung, Fellow, IEEE

Abstract—Edge machine learning involves the deployment of learning algorithms at the network edge to leverage massive distributed data and computation resources to train artificial intelligence (AI) models. Among others, the framework of federated edge learning (FEEL) is popular for its data-privacy preservation. FEEL coordinates global model training at an edge server and local model training at devices that are connected by wireless links. This work contributes to the energy-efficient implementation of FEEL in wireless networks by designing joint computation-and-communication resource management (C^2RM). The design targets the state-of-the-art heterogeneous mobile architecture where parallel computing using both CPU and GPU, called heterogeneous computing, can significantly improve both the performance and energy efficiency. To minimize the sum energy consumption of devices, we propose a novel C^2RM framework featuring multi-dimensional control including bandwidth allocation, CPU-GPU workload partitioning and speed scaling at each device, and C^2 time division for each link. The key component of the framework is a set of equilibriums in energy rates with respect to different control variables that are proved to exist among devices or between processing units at each device. The results are applied to designing efficient algorithms for computing the optimal C^2RM policies faster than the standard optimization tools. Based on the equilibriums, we further design energy-efficient schemes for device scheduling and greedy spectrum sharing that scavenges “spectrum holes” resulting from heterogeneous C^2 time divisions among devices. Using a real dataset, experiments are conducted to demonstrate the effectiveness of C^2RM on improving the energy efficiency of a FEEL system.

Index Terms—Radio resource management, energy-efficient computation and communication, scheduling, federated learning, heterogeneous computing.

I. INTRODUCTION

Realizing the vision of exploiting the enormous data distributed at edge devices (e.g., smartphones and sensors) to train artificial intelligence (AI) models has been pushing machine learning from the cloud to the network edge, called edge learning [1]. Currently, arguably the most popular edge learning framework is federated learning that preserves users’ privacy by distributed learning over devices [2]–[4]. Each round of the iterative learning process involves the broadcasting of a global model to devices, their uploading of local model updates computed using local datasets, and a server’s aggregation of the received local updates for updating the global model. Executing complex learning tasks on energy and resource constrained devices is a main challenge faced in the implementation of edge learning. To tackle the challenge, the next-generation mobile systems-on-chip (SoC) [5] will feature a heterogeneous architecture comprising a central processing unit (CPU) and a graphics processing unit (GPU) (and even a digital processing unit (DSP) in some designs) [5]. Experiments have demonstrated its advantages in terms of performance and energy efficiency. In this work, we address the issue of energy-efficient implementation of federated edge learning (FEEL) in a wireless system comprising next-generation devices capable of heterogeneous computing. To this end, a novel framework is proposed for energy-efficient joint management of computation-and-communication (C^2) resources at devices.

A. Edge Implementation of Federated Learning

The implementation of FEEL in wireless networks faces two key challenges among others. As mentioned, the distributed learning process requires potentially many edge devices to periodically upload high-dimensional local model updates to an edge server. This includes the first challenge that the excessive communication overhead generated in the learning process can overwhelm the air interface that has finite radio resources but needs to support FEEL as well as other services. Designing techniques for suppressing the overhead forms a vein of active research, called communication-efficient FEEL. Diversified approaches have been proposed such as device scheduling [6], [7], designing customized multi-access technologies [2], [3], and optimizing uploading frequencies [4].

The second challenge faced by FEEL is how to execute power hungry learning tasks (e.g., training AI models each typically comprising millions of parameters) on energy-constrained devices. Tackling the challenge by designing energy-efficient techniques leads to the emergence of an active research theme, called energy-efficient FEEL, which is the topic of current investigation. While there exists a rich literature of energy-efficient techniques for resource management (RM)
in radio access networks (see e.g., [8]), the designs of their counterparts for FEEL, which are the main research focus in energy-efficient FEEL, are different due to the changes on the system objective and operations. In particular, FEEL systems aim at improving the learning performance instead of providing a radio access service and performing update aggregation instead of decoupling multiuser data. In early works [9]–[11], researchers proposed radio-RM techniques (e.g., bandwidth allocation and device scheduling) to improve the tradeoff between devices’ transmission energy consumption and learning performance. More recent research accounts for computation energy consumption which usually constitutes a substantial part of a device’s total consumption in the learning process, motivating the design of C²RM techniques in [12], [13]. In a standard FEEL design, the updates by devices are usually assumed synchronized. It arises from the operation of gradient aggregation and refer to the requirement that all local updates need to be received by the server before the global model can be updated. Consequently, all devices are allowed the same duration (per-round latency). In [12], the learning latency is measured by a weighted sum of individual devices’ latencies and then the learning-energy tradeoffs are optimized. On the other hand, the idea of clock frequency control, or called dynamic voltage and frequency scaling (DVFS) [14], was explored in [13] for energy-efficient FEEL based on the assumption that each device has a CPU featuring DVFS. Then the C²RM was optimized where the communication RM is based on either time-division multiple access (TDMA) or non-orthogonal multiple access (NOMA) and the computation RM is based on dynamic frequency scaling. While prior work assumes single-processor devices, CPU-GPU heterogeneous computing discussed in the sequel is a new paradigm of mobile computing supporting applications with intensive data crunching. The area of energy-efficient FEEL based on heterogeneous computing is uncharted and explored in this work.

B. CPU-GPU Heterogeneous Computing

The mentioned next-generation heterogeneous SoC are capable of supporting diversified workloads such as communication, signal processing, inference, and learning, which arise from a wide range of new mobile applications. Examples of such chips include Snapdragon by Qualcomm, R-series by AMD, and Kirin 970 by Huawei. Via the cooperation of the CPU and GPU on the same chip for executing a single task, heterogeneous computing on such SoC fully utilizes the computation resources of both processors to optimize the computation performance and energy efficiency [15]. In particular, the new paradigm has been demonstrated by experiments to accelerate the running of deep neural networks on smartphones [16], [17]. Existing research on heterogeneous computing focuses on several main design issues including workload partitioning [5], [14], [18], [19], dynamic frequency scaling [20], [21], and memory access scheduling [22]. The first two issues are addressed in this work. Specifically, workload partitioning refers to dividing and allocating workload of a task over the integrated CPU and GPU according to the task requirements and the processors’ states (e.g., temperatures) and characteristics (e.g., speeds and energy efficiencies) so as to optimize the overall performance and efficiency [5], [14], [18], [19]. On the other hand, frequency scaling (or DVFS) previously considered for a single CPU [13] becomes the more sophisticated management in heterogeneous computing due to the joint CPU-GPU control [21]. While prior work on heterogeneous computing focuses on a single device, we study the joint control of workload partitioning and DVFS in the context of a large system comprising multiple devices and furthermore explore their integration with radio-RM.

C. Contributions

The objective of this work is not to contribute any new learning technique but to focus on C²RM to facilitate the implementation of a standard FEEL technique in a wireless network. To this end, we consider a FEEL system consisting of one edge server and multiple edge devices. Its main difference from those in existing work (see e.g., [12]) is that each device is equipped with a CPU-GPU platform enabling heterogeneous computing. The design objective is to minimize the sum energy consumption at devices under a guarantee on learning performance (latency and accuracy) by jointly controlling C²RM in the following four dimensions:

1) **Bandwidth allocation**: Allocating bandwidths to devices for transmission of local model updates under a constraint on the total uplink bandwidth;
2) **C² time division**: Dividing the allowed per-round latency for the computation and communication of each device;
3) **CPU-GPU workload partitioning**: Partitioning and allocating the computation workload for each device to its CPU and GPU;
4) **CPU-GPU frequency scaling**: Controlling the CPU-GPU frequencies/speeds at each device.

While the RM control in the first two dimensions are considered in prior work as discussed, the other two are unique for heterogeneous computing. Their joint control over multiple devices is a challenging and open problem. To the best of the authors’ knowledge, this work represents the first attempt on studying heterogeneous computing in the context of energy-efficient FEEL. The main contributions are described as follows.

- **Equilibrium based C²RM framework**: It is mathematically proved that the control policies are optimal if and only if they achieve the following equilibriums:
  1) The CPU-GPU pair of every device has equal energy-workload rates, defined as the increase in energy consumption per additional workload for each processing unit;
  2) A similar equilibrium exists with respect to the processing units’ computation speeds as their optimal values are proved to be proportional to the corresponding workloads.
  3) All devices have equal energy-time rates, defined as the energy rates with respect to the communication/computation latency;
4) All devices have equal energy-bandwidth rates, defined as the energy rates with respect to the allocated bandwidth.

The equilibriums are applied to obtaining the optimal polices for 1) computation RM, 2) communication RM, and 3) joint C²RM. They are either derived in closed-form or computed using low-complexity algorithms. In particular, for low-complexity joint C²RM, the problem is decomposed into one master problem for achieving the equilibrium in energy-time rates, and two sub-problems for separately achieving the two equilibriums in energy-workload rates and energy-bandwidth rates. The resultant algorithm is shown to have much lower complexity than a standard solution method such as block coordinate descent (BCD).

- C² aware scheduling: Building on the equilibrium framework, an energy-efficient scheduling scheme is designed to select a fixed number of devices for participating in FEEL. The novelty lies in the design of C² aware scheduling metric that balances both the channel state and computation capacity of each device. Specifically, given the objective of sum energy minimization, the metric is designed to be the energy consumption of a device given the derived optimal C² time division and equal bandwidth allocation. In addition, we analyze the effect of the number of scheduled devices on model convergence.

- Greedy spectrum sharing: For the preceding designs, we assume fixed bandwidth allocation in each round of the iterative learning process. Consequently, the heterogeneous computation latencies of devices generates "spectrum holes" (unused spectrum-time blocks) that can be scavenged for further improving the energy efficiency. To this end, the scheme of greedy spectrum sharing is designed featuring a novel metric, called energy-bandwidth acceleration rate and defined as the derivative of the energy-bandwidth rate for selecting an available device to transmit using the spectrum hole. A larger value of the metric, implies steeper energy reduction for a device when it is allocated additional bandwidth.

II. MODELS AND METRICS

Consider the FEEL system in Fig. 1 comprising a single edge server and K edge devices, denoted by an index set \( K = \{1, \cdots, K\} \). Each iteration between local gradient uploading and global model updating is called a communication round, or round for short. It is assumed that the server has perfect knowledge of the multiuser channel gains and local computation characteristics, which can be obtained by feedback. Using this information, the server determines the energy-efficient strategies for device scheduling and C²RM.

A. Federated Learning Model

A standard federated learning technique (see e.g., [23]) is considered as follows. A global model, represented by the parameter set \( w \), is trained collaboratively across the edge devices by leveraging local labelled datasets. For device \( k \), let \( D_k \) denote the local dataset and define the local loss function as:

\[
F_k(w) = \frac{1}{|D_k|} \sum_{(x_j,y_j) \in D_k} \ell(w; x_j, y_j), \quad \text{where } \ell(w; x_j, y_j) \text{ is the sample-wise loss function quantifying the prediction error of the model } w \text{ on the training sample } x_j \text{ with regard to its label } y_j.
\]

For ease of exposition, we focus on the gradient-averaging implementation while the current designs also apply to the model-averaging implementation [23]. In each round, say the \( i \)-th round, the server broadcasts the current model \( w^{(i)} \) as well as selection indicators \( \{p_k\} \) to all edge devices, where the indicator \( p_k = 1 \) if device \( k \) is scheduled, or 0 otherwise. Suppose \( M \) devices are scheduled for participation in each round, denoted by an index set \( M_i \) for the \( i \)-th round. Based on the received model \( w^{(i)} \), each scheduled device calculates the gradient \( \nabla F_k(w^{(i)}) \) using its local dataset. Upon completion, the local gradients are transmitted to the server for aggregation:

\[
g^{(i)} = \frac{1}{M} \sum_{k \in M_i} \nabla F_k(w^{(i)})
\]

Synchronized updates by devices are assumed. The global model is then updated by stochastic gradient descent (SGD) as \( w^{(i+1)} = w^{(i)} - \eta g^{(i)} \), where \( \eta \) is the learning rate. The process iterates until the model converges.

B. Model of Heterogeneous Computing

1) Workload model: Adopting a standard model (see e.g., [24]), the total workload \( W \) for a computing task is given as \( W = N_{\text{FLOP}} \times D \), where \( D \) is the local dataset size and \( N_{\text{FLOP}} \) denotes the number of floating point operations (FLOPs) needed for processing each sample. Furthermore, we define \( f^{(c)}_k \) and \( f^{(l)}_k \) (in cycle/s) as the clock frequency of the CPU and GPU at device \( k \), respectively. It follows that the computing speeds of CPU and GPU can be defined as \( f_k = f^{(c)}_k \times n_k \) and \( f'_k = f^{(l)}_k \times n'_k \), where \( n_k \) and \( n'_k \) denote the number of CPU and GPU FLOPs per cycle, respectively.\(^1\)

2) Workload partitioning model: For workload partitioning in local gradient computation, we consider input-sample partitioning, also known as data parallelism [27]–[29]. As a result, each processor (CPU or GPU) processes a fraction of input samples. As specified in [27], the local dataset \( D_k \) at device \( k \) is first partitioned into two sub-datasets, and then the CPU and GPU are assigned with the sub-datasets \( D_{k,\text{CPU}} \) and \( D_{k,\text{GPU}} \) satisfying \( D_k = D_{k,\text{CPU}} \cup D_{k,\text{GPU}} = D_k \). The partitioned workloads for CPU and GPU at device \( k \) are denoted

\(^1\)The architecture of a GPU comprises multiple streaming multiprocessors (SMs), each of which has many streaming processors (SPs) used in a single-instruction multiple data execution style [25]. The throughput of GPU satisfies \( n'_k = N^\text{SM} \times n'_k \) where \( N^\text{SM} \) and \( n'_k \) denote the number of SMs and the throughput per SM, respectively [26].
as $W_k$ and $W'_k$ respectively, where $W_k = N_{\text{FLOP}} \times |D_k^{\text{CPU}}|$ and $W'_k = N_{\text{FLOP}} \times |D_k^{\text{GPU}}|$, with

\[(\text{Workload constraint}) \quad W_k + W'_k = W, \quad \forall k \in \mathcal{M}. \quad (3)\]

In the training process, CPU and GPU independently calculate their gradients based on their assigned datasets in parallel, and then obtain $g_k^{\text{CPU}} = \frac{1}{|D_k^{\text{CPU}}|} \sum (x_j, y_j) \in D_k^{\text{CPU}} \nabla \ell (w; x_j, y_j)$ and $g_k^{\text{GPU}} = \frac{1}{|D_k^{\text{GPU}}|} \sum (x_j, y_j) \in D_k^{\text{GPU}} \nabla \ell (w; x_j, y_j)$. The local gradient of device $k$ can be calculated by $g_k = \frac{1}{T} \left( |D_k^{\text{CPU}}| g_k^{\text{CPU}} + |D_k^{\text{GPU}}| g_k^{\text{GPU}} \right)$, which gives the gradient $\nabla F_k(w)$. Given the partitioned workloads and CPU/GPU frequencies, the local computation time for the total workload $W$, denoted as $t'_k$, is determined by the processor that needs more time to finish completing the assigned workload and given as

\[
(\text{Local computation time}) \quad t'_k = \max \left\{ \frac{W_k}{f_k}, \frac{W'_k}{f'_k} \right\}, \quad \forall k \in \mathcal{M}. \quad (4)
\]

3) DVFS model: For a CMOS circuit, the power consumption of a processor can be modeled as a function of clock frequency: $P = \Psi (f^{(c)})^3$ with the coefficient $\Psi$ [in Watt/(cycle/s)³] depending on the chip architecture and $f^{(c)}$ being the clock frequency [30], [31]. Using this model[2], the power consumption of CPU and GPU at device $k$ can be written as³

\[
P_k^{\text{CPU}} = \Psi_k^{\text{CPU}} \left( f_k^{(c)} \right)^3 = C_k f_k^3, \quad (5)
\]

\[
P_k^{\text{GPU}} = \Psi_k^{\text{GPU}} \left( f_k^{(c)} \right)^3 = G_k f_k^3, \quad (6)
\]

where $C_k = \Psi_k^{\text{CPU}} / n_k^3$ and $G_k = \Psi_k^{\text{GPU}} / n_k^3$. Using these functions, frequency scaling refers to controlling the power of CPU and GPU by adjusting their speeds $f_k$ and $f'_k$, respectively.

Remark 1. (CPU/GPU Computation Efficiency). The coefficient $C_k$ (or $G_k$) in (5) (or (6)) characterizes the computation efficiency of a CPU (or GPU), defined as the rate of power growth in response to the increase of the cubed computing speeds. In practical data processing based on heterogeneous computing, the GPU tends to play a main role (with $G_k < C_k$) while the CPU contributes supplementary computing resources.

4) Energy consumption model: Given the time durations $W_k / f_k$ and $W'_k / f'_k$ for CPU and GPU to complete their tasks with the workloads $W_k$ and $W'_k$, the resultant energy consumption at device $k$ can be written as $E_k^{\text{CPU}} = C_k W_k f_k^2$ and $E_k^{\text{GPU}} = G_k W'_k f_k^2$. Then the total computation energy consumption of device $k$ per round is

\[
E_k^{\text{emp}} = C_k W_k f_k^2 + G_k W'_k f'_k^2, \quad \forall k \in \mathcal{M}. \quad (7)
\]

C. Multiple-Access Model

Consider the frequency-division multiple access (FDMA) for gradient uploading with the total bandwidth $B$. Let $B_k$ denote the allocated bandwidth for device $k$ in an arbitrary round, which is fixed throughout the round. This assumption is relaxed in Section V. Then we have the constraint:

\[
(\text{Bandwidth constraint}) \quad \sum_{k \in \mathcal{M}} B_k = B. \quad (8)
\]

Channels are assumed to be frequency non-selective. The edge server usually recruits idling devices as workers that either static or at most moving at pedestrian speeds [23]. For this reason, we adopt the model of slow block fading. To be specific, the channel gain of device $k$, denoted as $h_k$, is assumed to remain unchanged within one round but varies independently and identically (i.i.d.) over rounds. Given synchronous updates, a time constraint is set for each round:

\[
(\text{Latency constraint}) \quad t'_k + t_k \leq T, \quad \forall k \in \mathcal{M}, \quad (9)
\]

where $t'_k$ and $t_k$ denote the time for local training/computation and gradient uploading of device $k$, respectively; $T$ is the maximum total time for one round. A prerequisite for a scheduled device is that it can meet the above time constraint. Let $P_{\text{TX}}^k$ denote the transmission power [in Watt/Hz]
of device \( k \). The achievable rate can be written as \( r_k = B_k \log_2 \left( 1 + P_k T_k x_k \right), \forall k \in \mathcal{M} \), where \( N_0 \) is the spectrum density of the complex white Gaussian channel noise. Let \( L = |g| \alpha \) denote the gradient size (in bit) with \( \alpha \) denoting a sufficient large number of bits for quantizing each parameter with negligible distortion. Then the data rate is \( r_k = L/T_k, \forall k \in \mathcal{M} \). By combining the results, the communication energy consumption of device \( k \) per round is

\[
E_{k}^\text{comm} = B_k P_k T_k x_k = B_k t_k x_k \left( 2 \pi \frac{L}{T_k} - 1 \right), \forall k \in \mathcal{M}. \tag{10}
\]

### D. Performance Metric

It is assumed that the data distributed at devices satisfy the condition for model convergence within a finite number of rounds, denoted as \( N \) (see e.g., [35]). As implied by (10), capacity-achieving codes are deployed to ensure reliable transmissions. Consequently, wireless channels have no effects on \( N \) though they affect per-round latencies and energy consumption. The objective of energy-efficient RM is to minimize the total energy consumption of all active devices in the \( N \)-round learning process, namely

\[
\sum_{i=1}^{N} \sum_{k=1}^{K} \rho_k \left[ E_{k}^\text{comp} + E_{k}^\text{comm} \right] \text{ with } E_{k}^\text{comp} \text{ and } E_{k}^\text{comm} \text{ given in (7) and (10), respectively. Given block fading and the per-round latency constraint in (9), the objective can be straightforwardly proved to be equivalent to minimize the total energy consumption for each round. This is aligned with a standard approach in adaptive transmission and its proof is similar to those in the literature (see Lemma 1 in [36]) and thus omitted for brevity. It follows that for subsequent designs, it is sufficient to consider an arbitrary round and the corresponding total energy consumption, i.e., \( \sum_{k=1}^{K} \rho_k \left[ E_{k}^\text{comp} + E_{k}^\text{comm} \right] \), as the performance metric, is called sum energy hereafter.

### III. ENERGY-EFFICIENT C² RESOURCE MANAGEMENT

In this section, considering the case that all devices are scheduled for uploading (i.e., \( \rho_k = 1, \forall k \in \mathcal{K} \)), we study: 1) computation RM, 2) communication RM, and 3) joint C²RM by analyzing and computing the optimal policies. Then, the results are applied to establishing an energy-learning tradeoff.

#### A. Computation Resource Management

1) **Problem Formulation:** The computation resources distributed at devices can be managed by controlling their workload partitioning and speed scaling to minimize the sum computation energy. The design is formulated as the following optimization problem.

\[
\min_{\{W_k, f_k, f_k\}} \sum_{k=1}^{K} \left( C_k W_k f_k^3 + G_k W_k f_k^2 \right)
\]

\[\text{s.t. } W_k + W_k' = W, W_k \geq 0, W_k' \geq 0, \forall k \in \mathcal{K}, \]

\[t_k' = \max \left\{ \frac{W_k}{f_k}, \frac{W_k'}{f_k} \right\}, \forall k \in \mathcal{K}, \]

\[0 \leq t_k' \leq T_k', \forall k \in \mathcal{K}, \tag{P1}\]

where the three sets of constraints follow from (3), (4), and the time constraint with \( T_k' \) being the allowed maximum computation time for device \( k \).

2) **Optimal Policy:** The optimal policy for computation RM is derived in closed-form below.

**Lemma 1. (Speed Scaling Rule).** To minimize the sum computation energy, the computation speeds of CPU and GPU, \( f_k \) and \( f_k' \), should be scaled by following

\[
\frac{W_k}{f_k} = \frac{W_k'}{f_k'}, \forall k \in \mathcal{K}. \tag{11}
\]

given an arbitrary workload pair \( (W_k, W_k') \) with \( W = W_k + W_k' \).

**Proof:** See Appendix A.

The optimal strategy of workload-proportional speed scaling in Lemma 1 is intuitive and results from equalizing CPU and GPU computation time to avoid the slower one becoming the bottleneck of local gradient computation. Then, with Lemma 1, the original Problem P1 can be rewritten as follows:

\[
\min_{\{W_k, t_k'\}} \sum_{k=1}^{K} \left( C_k W_k^3 + G_k W_k^2 t_k^2 \right)
\]

\[\text{s.t. } W_k + W_k' = W, W_k \geq 0, W_k' \geq 0, \forall k \in \mathcal{K}, \]

\[0 \leq t_k' \leq T_k', \forall k \in \mathcal{K}. \tag{P2}\]

Next, it is found that the optimal policy achieves an equilibrium between the CPU and GPU on each device, in terms of energy-workload rate.

**Lemma 2. (Energy-Workload Rate Equilibrium).** Consider the energy-workload rates of CPU and GPU given as

\[
\frac{\partial E_{CPU}^k}{\partial W_k} = \frac{3C_k W_k^2}{t_k^3} \text{ and } \frac{\partial E_{GPU}^k}{\partial W_k} = \frac{3G_k W_k^2}{t_k^3}, \forall k \in \mathcal{K}, \tag{12}\]

where \( E_{CPU}^k = \frac{C_k W_k^3}{t_k^3} \) and \( E_{GPU}^k = \frac{G_k W_k^3}{t_k^3} \) denote the computation energy of CPU and GPU, respectively. Then, the necessary and sufficient condition for optimal workload allocation is

\[
\frac{\partial E_{CPU}^k}{\partial W_k} = \frac{\partial E_{GPU}^k}{\partial W_k}, \forall k \in \mathcal{K}, \tag{13}\]

with \( W_k + W_k' = W \).

**Proof:** See Appendix B.

Furthermore, one can observe that the objective function in P2 is a non-increasing function in \( t_k', \forall k \in \mathcal{K} \). Therefore, the optimality requires to maximize the computation time of each device, resulting in \( t_k' = T_k', \forall k \in \mathcal{K} \), which is independent of the workload partitioning. Using the result as well as Lemmas 1 and 2, the optimal computation RM policy is obtained as follows.

**Proposition 1. (Optimal Computation RM).** The optimal workload allocation is

\[
W_k^* = \sqrt{C_k W}, W_k'^* = \sqrt{G_k W}, \quad k \in \mathcal{K}, \tag{14}\]

(Original Workload Allocation)
where \( C_k \) and \( G_k \) are computation coefficients for CPU and GPU, respectively. Moreover, the optimal speed scaling for CPU and GPU is

\[
(f_k^* = \frac{W_k^*}{T_k^*}, \quad f_k^* = \frac{W_k^*/e}{T_k^*}, \quad k \in K,)
\]

where \( T_k^* \) is the allowed maximum computation time for device \( k \).

**Remark 2.** (Energy-Efficient CPU-GPU Heterogeneous Computing). According to Proposition 1, more workload tends to be allocated to the processor with a smaller computation coefficient indicating a higher computation efficiency (see Remark 1), thereby reducing the devices’ energy consumption. In addition, we emphasize on another fact that the proposed CPU-GPU heterogeneous computing scheme with policy in Proposition 1 is more energy-efficient than using the GPU (or CPU) only. Mathematical proof is presented in Appendix C.

**B. Communication Resource Management**

1) Problem Formulation: The total radio resources are managed by controlling bandwidth allocation and transmission time to minimize the sum transmission energy. The corresponding optimization problem can be formulated as

\[
\min_{\{B_k, t_k\}} \sum_{k=1}^{K} B_k t_k N_0 \left( 2 \frac{h_k^2}{T_k} - 1 \right),
\]

s.t. \( \sum_{k=1}^{K} B_k = B, \quad B_k \geq 0, \quad \forall k \in K, \)

\( 0 \leq t_k \leq T_k, \quad \forall k \in K. \)

The problem has a standard structure in the literature of energy-efficient communication (see e.g., [37]). It is convex and can be solved using a standard algorithm such as BCD. In the sequel, we present an alternative solution method yielding an equilibrium property that is useful for low-complexity policy computation faster than conventional methods.

2) Properties of Optimal Policies: As before, the objective of Problem \( \text{P3} \) is observed to be a non-increasing function in \( t_k \). Therefore, it is optimal to maximize the transmission time of each device, resulting in \( t_k^* = T_k, \quad \forall k \in K \). Next, to derive the optimal bandwidth allocation strategy, a necessary and sufficient condition is given as follows.

**Lemma 3.** (Energy-Bandwidth Rate Equilibrium). The energy-bandwidth rate as defined earlier is mathematically obtained as

\[
\frac{\partial E_k^{\text{cmm}}}{\partial B_k} = \frac{t_k N_0}{h_k^2} \left( 2 \frac{h_k^2}{B_k t_k} - 2 \ln 2 \frac{1}{B_k t_k} - 1 \right) < 0, \quad k \in K.
\]

The optimal bandwidth allocation equals the energy-bandwidth rates as

\[
\frac{\partial E_1^{\text{cmm}}}{\partial B_1} = \frac{\partial E_2^{\text{cmm}}}{\partial B_2} = \cdots = \frac{\partial E_K^{\text{cmm}}}{\partial B_K} = -\nu^*,
\]

where \( \nu^* \) is a constant and \( \sum_{k=1}^{K} B_k = B \).

**Proof:** See Appendix D.

It can be observed from the above lemma that the increase of allocated bandwidth reduces the communication energy consumption of the device. Then, the optimal communication RM policy directly follows from the energy-bandwidth rate equilibrium as stated below.

**Proposition 2.** (Optimal Bandwidth Allocation). The optimal policy for bandwidth allocation is

\[
B_k^* = \frac{L \ln 2}{T_k \left[ 1 + \mathcal{W}_0 \left( \frac{h_k^2 \nu^*-T_k N_0}{T_k N_0 e} \right) \right]}, \quad k \in K,
\]

where \( T_k \) is the allowed maximum transmission time for device \( k; \mathcal{W}_0(\cdot) \) is the Lambert \( W \) function (principal branch) and \( e \) is the Euler’s number.

Proposition 2 suggests that \( B_k^* \) is a non-increasing function with respect to \( h_k^2 \). It means that more bandwidths should be allocated to devices with weaker channels for the benefit of sum communication energy reduction.

Obtaining the optimal bandwidths \( \{B_k^*\} \) via (18) requires computing the optimal energy-bandwidth rate \( \nu^* \) by solving the following equation:

\[
\sum_{k=1}^{K} \left[ \frac{L \ln 2}{T_k \left[ 1 + \mathcal{W}_0 \left( \frac{h_k^2 \nu^*-T_k N_0}{T_k N_0 e} \right) \right]} \right] = B.
\]

Due to the intractability of the Lambert function \( \mathcal{W}_0 \) and the unknown range of \( \nu^* \), the solution cannot be found efficiently using standard methods such as Newton-Raphson method and bi-section search. An alternative method, derivative-free optimization (DFO), which requires no derivative, has too high complexity. To address this issue, a fast algorithm for policy computation is designed below.

3) Optimal Policy Computation: Instead of solving \( \nu^* \) and \( \{B_k^*\} \) in a sequential order, the fast algorithm calculates the optimal values iteratively, comprising the following two phases.

- **Phase I:** (Bandwidth Optimization). Given the energy-bandwidth rate \( \nu \), compute the bandwidth allocation \( \{B_k\} \) using (18). As \( \nu \) is not optimal, the computed \( \{B_k\} \) may not satisfy the bandwidth constraint in (8). Thus it is necessary to normalize each \( B_k \) as \( \tilde{B}_k = \frac{B}{\sum_{k=1}^{K} B_k} B_k \).

- **Phase II:** (Energy-Bandwidth Rate Updating). Calculate the respective energy-bandwidth rates \( \nu_k = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k} \), \( \forall k \in K \), by substituting \( \{\tilde{B}_k\} \) and \( \{t_k = T_k\} \) into (16). Then, update the current energy-bandwidth rate \( \nu \) using \( \{\nu_k\} \) as elaborated in the sequel.

The two phases are iterated until convergence as indicated by the energy-bandwidth rate equilibrium in (17).

A key step in the algorithm is the updating rule for \( \nu \) in Phase II. To derive the rule, a useful property is given as follows.

**Lemma 4.** Given \( \nu_k(B_k) = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k} \) in (16) and \( B_k(\nu) \) in (18), \( \forall k \in K \), for the \( i \)-th iteration between Phase I and II, the following holds:

- **Phase I:** (Bandwidth Optimization). Given the energy-bandwidth rate \( \nu \), compute the bandwidth allocation \( \{B_k\} \) using (18). As \( \nu \) is not optimal, the computed \( \{B_k\} \) may not satisfy the bandwidth constraint in (8). Thus it is necessary to normalize each \( B_k \) as \( \tilde{B}_k = \frac{B}{\sum_{k=1}^{K} B_k} B_k \).

- **Phase II:** (Energy-Bandwidth Rate Updating). Calculate the respective energy-bandwidth rates \( \nu_k = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k} \), \( \forall k \in K \), by substituting \( \{\tilde{B}_k\} \) and \( \{t_k = T_k\} \) into (16). Then, update the current energy-bandwidth rate \( \nu \) using \( \{\nu_k\} \) as elaborated in the sequel.

The two phases are iterated until convergence as indicated by the energy-bandwidth rate equilibrium in (17).

A key step in the algorithm is the updating rule for \( \nu \) in Phase II. To derive the rule, a useful property is given as follows.

**Lemma 4.** Given \( \nu_k(B_k) = -\frac{\partial E_k^{\text{cmm}}}{\partial B_k} \) in (16) and \( B_k(\nu) \) in (18), \( \forall k \in K \), for the \( i \)-th iteration between Phase I and II, the following holds:

- If \( \nu^{(i-1)} > \nu^* \), then 1) \( \nu^* < \max_k \{\nu_k^{(i)}\} < \nu^{(i-1)} \); and 2) \( \text{sgn} \left( \sum_{k=1}^{K} B_k^{(i)} - B \right) = -1 \);
where \( \nu \) is summarized in Algorithm 1.

2) \( \text{sgn} \left( \sum_{k=1}^{K} B_k^{(i)} - B \right) = 1; \)

where \( \nu_k^{(i)} = \nu_k(B_k^{(i)}) \) with \( \tilde{B}_k^{(i)} = \frac{B}{\sum_{k=1}^{K} B_k^{(i)}} B_k^{(i)} \) given \( B_k^{(i)} = B_k(\nu^{(i-1)}), \forall k \in \mathcal{K}. \)

Proof: See Appendix E. \( \square \)

Essentially, by induction, the above Lemma 4 simply implies that if the initial point \( \nu^{(0)} \) is greater than \( \nu^* \), and the updating rule adopted for \( \nu \) is \( \nu^{(i)} = \max \{ \nu_k^{(i)} \} \), the convergence of \( \nu \) to the global optimal \( \nu^* \) is guaranteed. This is also true for the case of \( \nu^{(0)} < \nu^* \) by involving the updating rule as \( \nu^{(i)} = \min \{ \nu_k^{(i)} \} \). It is further noted that the condition of \( \nu^{(0)} > \nu^* \) or \( \nu^{(0)} < \nu^* \) can be determined by calculating \( \text{sgn} \left( \sum_{k=1}^{K} B_k^{(1)} - B \right) \), where \( B_k^{(1)} = B_k(\nu^{(0)}) \) follows from (18). In summary, we have the following updating rule for \( \nu \) in Phase II:

\[
\text{(Update rule)}: \quad \nu^{(i)} = \begin{cases} 
\max_k \{ \nu_k^{(i)} \}, & \text{sgn} \left( \sum_{k=1}^{K} B_k^{(1)} - B \right) = -1; \\
\min_k \{ \nu_k^{(i)} \}, & \text{otherwise,}
\end{cases}
\]

where \( \{ \nu_k^{(i)} \} \) and \( \{ B_k^{(1)} \} \) are specified in Lemma 4. Based on the above rule, the algorithm for optimal bandwidth allocation is summarized in Algorithm 1.

**Algorithm 1 Optimal Bandwidth Allocation**

**Input:** initial value of \( \nu \).

**Output:** optimal \( \nu^* \) and \( \{ B_k^* \} \).

**Calculate:** indicator \( y = \text{sgn} \left( \sum_{k=1}^{K} B_k(\nu) - B \right) \).

**Repeat:**

- Calculate \( \{ B_k \} \) by substituting \( \nu \) into Proposition 2;
- Normalize \( \tilde{B}_k = \frac{B}{\sum_{k=1}^{K} B_k}, k \in \mathcal{K} \);
- Calculate \( \{ \nu_k \} \) by substituting \( \{ \tilde{B}_k \} \) into (16), \( k \in \mathcal{K} \);
- Update \( \nu = \frac{1}{\sum_{k=1}^{K} B_k^{(1)}} \max_k \{ \nu_k \} + \frac{1}{\sum_{k=1}^{K} B_k^{(1)}} \min_k \{ \nu_k \}; \)

**Until** \( \max_k \{ \nu_k \} < \varepsilon \) (to equalize the energy-bandwidth rates).

**Remark 3. (Low-Complexity and Optimality).** The complexity of our proposed Algorithm 1 is \( O(\log \frac{1}{\varepsilon}) \) with \( \varepsilon \) denoting the target accuracy. For comparability, the computation of the DFO method for solving \( \nu^* \) and \( \{ B_k^* \} \) in a sequential order is \( O(\frac{K}{\varepsilon}) \), and that of the BCD algorithm for directly solving Problem \( P3 \) is \( O(\frac{K}{\varepsilon} \log \frac{1}{\varepsilon}) \), which are much higher than that of Algorithm 1. Furthermore, the optimality is guaranteed by Algorithm 1 as indicated by Lemma 4.

**C. Joint \( C^2 \) Resource Management**

1) **Problem Formulation:** Building on the preceding results, an energy-efficient \( C^2 \)RM framework is designed by joint control of workload partitioning, \( C^2 \) time division, and bandwidth allocation to minimize sum energy. The optimization problem is formulated as

\[
\min_{W_k, t_k^*, B_k, t_k} \sum_{k=1}^{K} \left[ C_k W_k^3 + G_k W_k^3 + B_k t_k N_0 \right. \\
\left. \frac{B_k}{h_k^2} \left( 2 \frac{p_k}{h_k} - 1 \right) \right]
\]

\[\text{s.t.} \quad W_k + t_k^* = W, \quad W_k \geq 0, \quad W_k' \geq 0, \quad \forall k \in \mathcal{K}, \]

\[\sum_{k=1}^{K} B_k = B, \quad B_k \geq 0, \quad \forall k \in \mathcal{K}, \]

\[t_k^* + t_k \leq T, \quad t_k^* \geq 0, \quad t_k \geq 0, \quad \forall k \in \mathcal{K}. \]

**Remark 4. (Feasibility Checking).** The constraints on maximum frequency and transmit power are omitted to simplify the problem and the solution, because the current design focuses on the regime of energy-efficient system operation where such constraints are usually inactive. The current design can be modified to ensure the feasibility under such constraints by translating them into the requirements on the minimum computation time and bandwidth, which can be included as guidelines for deploying the current design. In particular, under the constraints: \( f_k \leq f_{k, \max}, f_k' \leq f_{k, \max}', p_{k, \max} \leq P_{k, \max}, \forall k \in \mathcal{K} \), the required bandwidth for \( P4 \) is given as

\[
B \geq \sum_{k=1}^{K} \left( T - \frac{W_k}{f_{k, \max} + f_{k, \max}'} \log_2 \left( 1 + P_{k, \max} h_k^2 \right) \right). \quad (20)
\]

2) **Properties of Optimal Policy:** It is shown in the sequel that a set of equilibriums exist among devices in terms of energy rates with respect to different types of control variables when \( C^2 \)RM is optimally energy-efficient. The insights facilitate designing low-complexity policy for solving Problem \( P4 \). First, by the same argument as in Subsections A and B, the latency constraint in Problem \( P4 \) should be active for energy minimization: \( t_k^* + t_k = T, \forall k \in \mathcal{K} \). The optimal \( C^2 \) time division of \( T \) has the following property.

**Lemma 5. (Energy-Time Rate Equilibrium).** The energy-time rates as defined earlier can be mathematically obtained as

\[
\xi_k := \frac{\partial E_{k, \text{cm}}}{\partial t_k} = -\frac{2(C_k W_k^3 + G_k W_k^3)}{t_k^*}, \quad k \in \mathcal{K}, \quad (21)
\]

and

\[
\xi_k := \frac{\partial E_{\text{cmp}}}{\partial t_k} = \frac{B_k N_0}{h_k^2} \left( 2 \frac{p_k}{h_k} - \frac{L\ln 2}{B_k t_k} - 2 \frac{p_k}{h_k} - 1 \right), \quad k \in \mathcal{K}. \quad (22)
\]

The optimal \( C^2 \) time division for each device requires the energy-time rate equilibrium:

\[
\frac{\partial E_{\text{cmp}}}{\partial t_k'} = \frac{\partial E_{\text{cmp}}}{\partial t_k}, \quad \forall k \in \mathcal{K}, \quad (23)
\]

with \( t_k^* + t_k = T \).

The proof is similar to that for Lemma 2 and thus omitted for brevity. Next, one can observe that Problem \( P4 \) integrates \( P2 \) and \( P3 \) by summing their objectives and combining their constraints. Therefore, \( P4 \) is also convex and furthermore the results in Lemmas 2 and 3 hold for the current case. Then, combining Lemmas 1, 2, 3 and 5 yields the following main result.
Theorem 1. (Equilibrium Based C²RM). The optimal C²RM policy achieves the following equilibriums:

1) For each device, the optimal C² time division equals the energy-computation-time and energy-communication-time rates:
\[ \frac{\partial E_{k}^{\text{comp}}}{\partial T_{k}} = \frac{\partial E_{k}^{\text{comm}}}{\partial t_{k}}, \forall k \in \mathcal{K}. \tag{24} \]

2) The optimal bandwidth allocation equals the energy-bandwidth rates:
\[ \frac{\partial E_{k}^{\text{comm}}}{\partial B_{1}} = \frac{\partial E_{k}^{\text{comm}}}{\partial B_{2}} = \cdots = \frac{\partial E_{k}^{\text{comm}}}{\partial B_{K}}, \tag{25} \]

3) The optimal workload allocation at each device equals the energy-workload rates:
\[ \frac{\partial E_{k}^{\text{CPU}}}{\partial W_{k}} = \frac{\partial E_{k}^{\text{GPU}}}{\partial W_{k}}, \forall k \in \mathcal{K}. \tag{26} \]

4) The optimal speed scaling at each device equals the energy-speed rates:
\[ \frac{\partial E_{k}^{\text{CPU}}}{\partial f_{k}} = \frac{\partial E_{k}^{\text{GPU}}}{\partial f_{k}}, \forall k \in \mathcal{K}. \tag{27} \]

Remark 5. (Equalizing C² Heterogeneity). In the process of synchronized updates, Theorem 1 suggests that energy-efficient C²RM should equalize the heterogeneity in communication channels and computation efficiencies using the multi-dimensional control variables. First, the heterogeneity in multiuser channel states and CPU/GPU computation efficiencies at each device is equalized by bandwidth allocation and workload partitioning (with speed scaling) as reflected in the second and third (with fourth) equilibriums. Second, the heterogeneity in C² speeds is equalized by adjusting the C² time division according to the first equilibrium. It is worth mentioning that C² time division represents a C² tradeoff that more communication resources can compensate for the lack of computation resources and vice versa.

3) Optimal Policy Computation: Though the optimal solution for Problem P4 can be obtained by finding the equilibriums in Theorem 1 numerically using the standard method of BCD, it has high complexity. To tackle this challenge, we design a more efficient algorithm as follows. First, Problem P4 can be decomposed into one master problem and two sub-problems as follows:

- (Master Problem): The master problem is the optimization of C² time division. Denote \( E^{\text{comp}}(\{\tilde{T}_{k}\}) \) and \( E^{\text{comm}}(\{\tilde{T}_{k}\}) \) as the computation and communication energy, which are functions of the allowed maximum communication time \( \{\tilde{T}_{k}\} \) given optimized resource management strategies output by sub-problem 1 and sub-problem 2. Then, the master problem is cast as
\[
\min_{\{\tilde{T}_{k}\}} E^{\text{comp}}(\{\tilde{T}_{k}\}) + E^{\text{comm}}(\{\tilde{T}_{k}\}) \tag{MP}
\]

such that \( 0 < \tilde{T}_{k} < T, \forall k \in \mathcal{K}. \)

- (Two Sub-problems):

- (Computation RM): Problem P1 with \( \{T'_{k} = T - \tilde{T}_{k}\}; \)

- (Communication RM): Problem P3 with \( \{T_{k} = \tilde{T}_{k}\}. \)

As Problem P4 is convex, the optimal solutions can be obtained via iterations between the master problem and two sub-problems. Each iteration comprises two steps: 1) given \( \{\tilde{T}_{k}\} \), compute the optimal RM strategies by solving two sub-problems; 2) given allocated bandwidths and partitioned workloads, calculate the sub-gradient of the master problem and apply it to updating \( \{\tilde{T}_{k}\} \) via the gradient descent method. The two steps are iterated until convergence.

Note that the solution of one sub-problem, namely Problem P1, can be calculated directly via (14) and (15) while numerically calculation is required for solving the other sub-problem, namely Problem P3. Given the efficient Algorithm 1 developed for computing \( \{B_{k}^{\ast}\} \), in the sequel, we aim at further improving the efficiency by proposing a novel initialization method of \( \nu \). To this end, another useful property is introduced in the following lemma.

Lemma 6. The optimal \( \nu^{\ast} \) under the arbitrary given C² time division, namely \( T'_{k} + T_{k} = T \) with \( t'_{k} \leq T'_{k} \) and \( t_{k} \leq T_{k}, \forall k \in \mathcal{K}, \) can be calculated as
\[
\nu^{\ast} = -\frac{1}{B} \sum_{k=1}^{K} T_{k} \left. \frac{\partial E_{k}^{\text{comm}}}{\partial t_{k}} \right|_{T_{k}} = -\frac{1}{B} \sum_{k=1}^{K} T_{k} \xi_{k}(T_{k}), \tag{28}
\]

where \( \xi_{k}(T_{k}) = \left. \frac{\partial E_{k}^{\text{comm}}}{\partial t_{k}} \right|_{T_{k}} \) is the energy-communication time rate under current time division.

The above result can be proved by noting the relation \( T_{k} \frac{\partial E_{k}^{\text{comm}}}{\partial t_{k}} |_{T_{k}} = B_{k} \frac{\partial E_{k}^{\text{comm}}}{\partial B_{k}} |_{B_{k}} = -\nu^{\ast} B_{k}^{\ast} \) given the bandwidth constraint, namely \( \sum_{k=1}^{K} B_{k}^{\ast} = B \). As the values of \( \{\xi_{k}(T_{k})\} \) are not attainable, \( \nu^{\ast} \) can not be solved directly. Nevertheless, we can get a good initial point for \( \nu \) by taking advantage of (28). To be specific, due to the fact that \( \xi_{k}'(T_{k}) \) and \( \xi_{k}(T_{k}) \) converge with iterations and \( \xi_{k}'(T_{k}) \) can be calculated by solving P1 in closed-form, we propose to initialize \( \nu \) as
\[
\nu_{0} := -\frac{1}{B} \sum_{k=1}^{K} T_{k} \xi_{k}'(T_{k}), \tag{29}
\]
Such initialization provides an increasingly better initial point that is closer to the optimal solution as the outer iteration proceeds. The algorithm for computing the optimal C²RM policy is summarized in Algorithm 2.

Remark 6. (Low-Complexity Algorithm). The proposed Algorithm 2 is of low-complexity, which has complexity up to \( O(K \log^{2} \frac{1}{\varepsilon}) \) compared to that of the conventional BCD method for solving P4, which has complexity \( O(K \log \frac{1}{\varepsilon}) \).

D. Discussion on Energy-Learning Tradeoff

In the literature, the model convergence of federated learning is characterized by either the averaged gradient norm (see e.g., [35]) or the loss function (see e.g., [23]), as a monotone decreasing function of the number of required rounds, \( N \), and participating devices \( K \). For example, it is reported in [35] that
with a properly chosen learning rate (step size), the averaged gradient norm over rounds is shown to be proportional to $1/\sqrt{N}$ and $1/\sqrt{K}$. In existing work, the per-round latency, $T$, is assumed constant and thereby can be omitted in the convergence analysis. By considering $T$ or $N$ as a function of $C^2$ sum energy, denoted as $E_{\Sigma}$, we can discuss a learning-energy tradeoff as follows.

**Definition 1. (Learning Latency).** The learning latency is defined as $T_{\text{total}} = N \times T$ (in second).

1) **Fixed number of participating devices, $K$:** This implies a fixed distributed dataset and thus fixed $N$. On the other hand, it can be inferred from Lemma 5 that $T$ is a monotone decreasing function of $E_{\Sigma}$. Thus, the learning latency can be written as $T_{\text{total}} = N \times T(E_{\Sigma})$, which establishes the tradeoff between $T_{\text{total}}$ and $E_{\Sigma}$. Specifically, reducing $E_{\Sigma}$ can result in increased learning latency, and vice versa. Based on Lemma 5, one can see that the reduction in the learning latency leads to the increment in sum energy at a rate faster then linear.

2) **Fixed per-round latency, $T$:** For this case, an energy-learning tradeoff also exists. To be specific, it can be easily shown that the number of devices that can satisfy the per-round latency constraint is a monotone increasing function of $E_{\Sigma}$. Thereby, reducing $E_{\Sigma}$ can result in a reduced number of devices participating in learning. This gives rise to a larger required number of rounds for convergence due to the shrinking of distributed dataset. Thus, the learning latency can be written as $T_{\text{total}} = N(E_{\Sigma}) \times T$ with $N(E_{\Sigma})$ being a monotone decreasing function. Then the energy-learning tradeoff is one that shortening learning latency needs more energy and vice versa. Particularly, the intermediate parameter, namely the number of participating devices, controls the said tradeoff in this case. This further motivates us to explore $C^2$ aware device scheduling in the following section.

In general, the functions $T(E_{\Sigma})$ in the first case and $N(E_{\Sigma})$ in the second case have no closed form. The analysis of their properties (e.g., scaling laws) is an interesting topic but outside the scope of this work.

IV. $C^2$ AWARE SCHEDULING

When there are many devices providing more than sufficient data, it is desirable from the energy-efficiency perspective to select only a subset of devices for participating in model training. In this section, the scheduler design is presented and the effect of scheduling on model convergence is quantified.

A. Scheduler Design

The mathematical formulation of the design is similar to P4 by modifying the objective as

$$\sum_{k=1}^{K} \rho_k \left[ C_k \frac{W_k^2 + G_k W_k^3}{W_k^3} + B_k t_k \right]$$

under the additional constraints, namely

$$\sum_{k=1}^{K} \rho_k = M, \rho_k \in \{0, 1\}, \forall k \in K,$$

where $\{\rho_k\}$ are the indicator variables for selecting devices. This mixed integer non-linear problem is NP-hard and classical optimal algorithms (e.g., branch and bound) have too high complexity to be practical when $K$ is large. To tackle this challenge, we propose a novel $C^2$ aware scheduler design of low-complexity to minimize sum energy.

We consider the problem of selecting $M$ out of $K$ devices for FEEL. The data distribution over devices is assumed i.i.d. Then with $M$ fixed, device scheduling has no effect on the number of rounds $N$ required for learning (whose dependence on $M$ is studied in Section B). This allows scheduling decisions to be based on the $C^2$ states of devices, giving the name of $C^2$ aware scheduling. The key element of the scheduler is a scheduling metric designed as follows. Based on the preceding analysis, it is desirable to select devices with good channels and/or good computation efficiencies for energy reduction. To identify such devices, we propose to first perform equal bandwidth allocation over all devices (i.e., $B_k = \bar{B} = B/K, \forall k \in K$) and then evaluate the resulting energy consumption of each device. Note that equal bandwidth allocation enables the high energy consumption of a device to indicate: 1) a poor channel, or 2) a low computation efficiency, or 3) both. Therefore, the devices’ energy consumption with equal bandwidth allocation is a suitable scheduling metric given as:

$$E_k = \frac{B \bar{N}_0}{h^2} \left( 2N_k t_k^3 - \frac{2N_k L L_{\text{opt}}}{2 h^3} (T - t_k^3) \right) + \frac{2N_k W^3}{(T - t_k^3)^3}$$

where

$$a_k = \frac{C_k G_k}{\sqrt{C_k + \sqrt{G_k}}}, \forall k \in K.$$  

To compute its value, the optimal transmission time $t_k^*$ is needed given $B_k = \bar{B} = B/K$. Based on the energy-time rate equilibrium developed in Lemma 5, $t_k^*$ solves the equation below:

$$f(t_k^*) = \frac{\bar{B} \bar{N}_0}{h^2} \left( 2N_k t_k^3 - \frac{2N_k L L_{\text{opt}}}{2 h^3} (T - t_k^3) \right) + \frac{2N_k W^3}{(T - t_k^3)^3} = 0.$$  

A closed-form expression for $t_k^*$ is hard to derive but can be numerically computed by a bi-section search since $f(t_k^*)$ is a monotonically increasing function. Given the values of $\{E_k\}$, the $C^2$ aware scheduler selects $M$ devices with the smallest

**Algorithm 2 Optimal $C^2$RM**

**Input:** initial values of $\{\bar{T}_k\}$.

**Output:** optimal $\{B_k^*, \{t'_k = T - \bar{T}_k\}, \{t_k^* = \bar{T}_k\}$.

**Repeat:**

- Solve the two subproblems with $T_k = \bar{T}_k$ and $T_k' = T - \bar{T}_k$ as input:

  - **Solve Problem (P1):**
    - calculate $\{W_k\}$ and $\{W'_k\}$ using (14);
    - obtain $\{\xi'_k\}$ by substituting $\{t'_k = T_k'\}$ and $\{W_k, W'_k\}$ into (21);

  - **Solve Problem (P3):**
    - calculate $\{B_k\}$ and solve $\nu$ using Algorithm 1 by involving the initialization method (29);
    - obtain $\{\xi_k\}$ by substituting $\{B_k\}$ and $\{t_k = T_k\}$ into (22);

- Update the time division: $\bar{T}_k = \bar{T}_k - \eta (\xi_k - \xi_k')$, $\forall k \in K$;

**Until** $\|\xi' - \xi\|_2 \leq \varepsilon$ (to equalize the energy-time rates), where

$$\xi' = [\xi'_1, \ldots, \xi'_K]^T$$

and

$$\xi = [\xi_1, \ldots, \xi_K]^T.$$
values. The scheduling scheme is summarized in Algorithm 3.

Algorithm 3 $C^2$ Aware Scheduling

Initialization: \( \forall k \in K, \rho_k = 0 \), \( B = B/K \).
Output: Subset \( M = \{ k \in K \mid \rho_k = 1 \} \).

- Solve for the optimal time divisions \( \{ t_k^* \} \) using (31) and a bi-section search;
- Calculate the scheduling metrics \( \{ E_k \} \) by substituting \( \{ t_k^* \} \) into (30);
- Select \( M \) devices with the smallest \( E_k \) and set their \( \rho_k = 1 \).

B. Effect of Scheduling on Convergence

In this subsection, we quantify the effect of $C^2$ aware scheduling (without considering data importance) on the convergence rate of FEEL (in round) due to the reduced size of the dataset for model updating. For tractable analysis, we follow the standard assumptions on the loss function as made in the literature (see e.g., [38]).

Assumption 1. It is assumed that the loss functions has the following two properties:

- (Convexity and Smoothness). All functions \( \{ F_k \} \) are convex and \( \beta \)-smooth, that is, for all \( u, v \), we have
  \[
  F_k(u) \geq F_k(v) + \langle \nabla F_k(v), u - v \rangle, \quad (32)
  \]
  \[
  F_k(u) \leq F_k(v) + \langle \nabla F_k(v), u - v \rangle + \frac{\beta}{2} \| u - v \|^2. \quad (33)
  \]

- (Variance Bound). Stochastic gradients \( \{ \nabla F_k(w) \} \) are unbiased and variance bounded by \( \sigma^2 \), that is,
  \[
  E[\nabla F_k(w)] = \nabla F(w), \quad (34)
  \]
  \[
  E[\|\nabla F_k(w) - \nabla F(w)\|^2] \leq \sigma^2, \quad (35)
  \]

where \( \nabla F_k(w) \) and \( \nabla F(w) \) denote the gradients of a local loss function and the global loss function, respectively, and the expectations are taken over all devices.

Under the above assumptions, the upper bound on the convergence rate of FEEL algorithm with device scheduling is given in the following theorem.

Theorem 2. (Convergence Rate with Scheduling). Given the learning rate as \( \eta = \frac{1}{\sqrt{N}} \), the convergence rate of the FEEL algorithm with device-scheduling can be upper-bounded as

\[
\frac{1}{N} \sum_{i=0}^{N-1} E_M \left[ F(w^{(i)}) - F(w^*) \right] \leq \frac{1}{\sqrt{N}} \left[ \| w^{(0)} - w^* \|^2 + \frac{\sigma^2(K - M)}{(K - 1)M} \right]. \quad (36)
\]

where \( N \) denotes the number of communication rounds.

For brevity, the detailed proof is omitted here and presented in the extended version [39]. Compared with the existing results without scheduling (see e.g., [40]), the last term in (36) is new that characterizes the effect of scheduling on learning. One can observe from the term that increasing the number of scheduled devices \( M \) leads to the vanishment of the biased term at a rate of \( O(\frac{1}{M}) \), giving rise to a faster convergence rate, however, at the cost of more sum energy consumption scales faster than linearly with \( M \).

V. EXTENSION AND DISCUSSION

A. Greedy Spectrum Sharing

In the preceding sections, the bandwidths are allocated at the beginning of each round and then fixed throughout the round. However, as mentioned, the heterogeneity in the computation durations of devices may result in spectrum holes as illustrated in Fig. 2. Scavenging them by spectrum sharing among devices can reduce sum energy, which is the theme of this section. While optimizing the spectrum sharing is intractable, we design a practical scheme based on the greedy principle. The scheme allocates the spectrum hole upon its arrival to one of available devices for transmission (in addition to the pre-allocated bandwidth) so as to reduce its energy consumption, called greedy spectrum sharing. To design the scheme, we divide the round into multiple time slots. Furthermore, let \( S \) and \( S_i \) denote the sets of scheduled devices and a subset of devices that have completed computation at the beginning of a particular time slot, respectively. Then, the resulting spectrum holes have the total bandwidth of \( B - \sum_{k \in S} B_k \) with \( B_k \) denoting the pre-allocated bandwidth to device \( k \), which is determined at the beginning of each round. Intuitively, all the unoccupied bandwidths should be allocated to the device \( k \in S_i \) that has the largest energy reduction with respect to bandwidth growth. Mathematically, this selection metric is to select the device, denoted as \( k^* \), with the minimum energy-bandwidth rate at \( B_k^* \) provided \( \frac{\partial E_{\text{sum}}}{\partial B_k} < 0 \), \( \forall k \in K \), that is

\[
k^* = \arg \min_{k \in S_i} \left| \frac{\partial E_{\text{sum}}}{\partial B_k} \right| B_k^* . \quad (37)
\]

However, according to Lemma 3, the energy-bandwidth rates of all scheduled devices are equalized at equilibrium, making this criterion ineffective. To address this issue, we design an alternative metric that can also effectively reduce sum energy. To this end, we introduce the following notion.
Definition 2. (Acceleration Rate). The acceleration rate is defined as the partial derivative of the energy-bandwidth rate:
\[
\varphi_k(B_k) := \frac{\partial^2 E_{\text{mm}}^k}{\partial B_k^2} = \frac{2\pi^2 L^2 N_0 (\ln 2)^2}{B_k^3 h_k^2} > 0, \forall k \in \mathcal{K}. \tag{38}
\]

A device with a small acceleration rate implies a lower energy-bandwidth rate upon being allocated with extra bandwidths, leading to more energy reduction. Therefore, we propose to adopt the criterion of minimum acceleration rate at \(\{B_k = B_k^*\}\), namely \(\{\varphi_k(B_k^*)\}\), as follows

(Device Selection) \(k^* = \arg \min_{k \in S_t} \varphi_k(B_k^*)\). \tag{39}

Based on the above criterion, the scheme is summarized in Algorithm 4.

Algorithm 4. Greedy Spectrum Sharing

**Initialization:** Apply Algorithm 3 and 2 in sequential order to obtain \(\{\rho_k^*, B_k^*\}\).

Denote \(\Delta t\) as the time slot duration and let \(t_{\text{count}} = 0\).

For the subset of devices \(S = \{k \in \mathcal{K} \mid \rho_k = 1\}\):

**While** \(t_{\text{count}} < T\):

- Denote \(S_r\) as the set of devices that have completed local computation at time \(t_{\text{count}}\):
  - For \(k \notin S_r\), no bandwidth will be occupied by them:
  - For \(k \in S_r\), each device will first be allocated with bandwidth \(B_k^*\):
    - Calculate \(\varphi_k\) for each \(k \in S_r\) using (38);
    - Select the device via (39) and allocate all the accessible bandwidth \(B - \sum_{k \in S_r} B_k^*\) to it;
  - \(t_{\text{count}} = t_{\text{count}} + \Delta t\).

B. Extension to Frequency-Selective Channels

The joint C²RM and device scheduling algorithms in the preceding sections are designed assuming frequency non-selective channels. In this subsection, we discuss the deployment of the algorithms in the case of frequency selective channels in an orthogonal frequency division multiple access (OFDM/MA) system. Essentially, the optimization of the joint design needs to account for one more dimension, i.e., channel variation in frequency for each device. One practical approach for applying the current design is to consider the average channel gain for each device as in \([41], [42]\). To be specific, for each device, say device \(k\), divide its frequency-selective channel into \(|N_k|\) frequency non-selective sub-channels with \(N_k\) being the sub-channel index set and \(h_k^{(i)}\) denoting the channel gain of the \(i\)-th sub-channel. Then its average channel gain is given by

\[
\bar{h}_k = \sqrt{\frac{\sum_{i \in N_k} (h_k^{(i)})^2}{|N_k|}}, \quad \forall k \in \mathcal{K}. \tag{40}
\]

By exploiting the average channel gain \(\bar{h}_k\) as a surrogate for \(h_k\) of device \(k\) defined for the frequency non-selective counterpart in preceding sections, all the resource allocation strategies and algorithms developed for the frequency non-selective case can be applied to the current scenario.

VI. EXPERIMENTAL RESULTS

A. Experiment Setup

The simulation settings are as follows unless specified otherwise. In the FEEL system, there are \(K = 50\) devices and each of them is capable of CPU-GPU heterogeneous computing. The devices’ CPU and GPU coefficients, \(\{C_k\}\) and \(\{G_k\}\), are uniformly selected from the set \(\{0.020, 0.021, \ldots, 0.040\}\) and \(\{0.001, 0.002, \ldots, 0.010\}\) [in Watt-(MFLOPs/s)\(^{-1}\)], respectively\(^5\). Consider an FDMA system with the uplink bandwidth \(B = 5\, \text{MHz}\). The channel gains \(\{h_k\}\) are modeled as i.i.d Rayleigh fading with average path loss set as \(10^{-3}\). The noise variance is \(N_0 = 10^{-9}\) W/\(\text{Hz}\). The classification task aims at classifying handwritten digits from the well-known MNIST dataset. The classifier model is implemented using a 6-layer convolutional neural network (CNN) which consists of two \(5 \times 5\) convolution layers with ReLU activation, each followed by \(2 \times 2\) max pooling, a fully connected layer with 50 units and ReLU activation, and a final softmax output layer. The total number of parameters is \(21,840\) and the computation workload is \(W = 9.75\) MFLOPs. Furthermore, we suppose that each parameter of the training model gradient is quantized into 16 bits, and as a result, the transmission overhead is \(L = 3.49 \times 10^5\) bits.

B. C² Resource Management

1) Energy-efficient RM: The performance of the proposed computation RM, communication RM and joint C²RM policies are evaluated by simulations. To be specific, we consider two settings: a) uniform time division [see Fig. 3(a)]; and b) optimal time divisions [see Fig. 3(b)]. Under each setting, we consider four schemes: 1) C²RM, 2) only computation RM, 3) only communication RM, and 4) without RM. In particular, to distinguish our proposed optimal C²RM policy, we name the policy with C²RM under uniform time division setting as “separate C²RM” while the proposed optimal one is called “joint C²RM”. The curves of the sum energy versus the per-round latency \(T\) are illustrated in Fig. 3. Several observations can be made. First, the sum energy reduces as \(T\) grows for all cases. This coincides with the results in Lemma 5 that the energy consumption is a monotonically decreasing function in computation and communication time. Second, for either setting a) or b), it can be found that the two schemes 2) and 3), namely only computation RM and only communication RM, outperform the scheme without RM but underperform the C²RM. For example, under setting a), they reduce the sum energy of the policy without RM by 13.8% and 43.5%, respectively, for per-round latency equal to 1.0 s. Meanwhile, the separate C²RM scheme reduces the sum energy of the policy without RM by 57.3%. These results demonstrate the effectiveness of our proposed workload and bandwidth allocation policies for C²RM. Third, comparing the results in setting a) with b), we find that the strategy of C² time division plays a significant role in energy efficiency. For example, the four schemes 1)-4) in b) with optimal time divisions

\(^5\)The range of \(\{C_k\}\) and \(\{G_k\}\) are set according to the fact that local computation energy consumption per round should be within the range based on the real test, e.g., the test on the Samsung Exynos 5422 SoC [14].
reduce the sum energy of those in a) by 17.2%, 47.5%, 9.8%, and 43.5%, respectively, for per-round latency equal to 1.0 s. This coincides with Remark 4 that the optimal C^2 time division achieves the best energy efficiency by balancing C^2 heterogeneity. Fourth, our proposed joint C^2RM policy outperforms all other schemes, showing its effectiveness. For example, it reduces the sum energy of the schemes 1)-4) in a) by 17.2%, 59.0%, 37.4%, and 64.6%, as well as the schemes 2)-4) in b) by 21.9%, 31.3%, and 37.4%, respectively, for per-round latency equal to 1.0 s.

2) Greedy spectrum sharing: The performance of the proposed greedy spectrum sharing algorithm in Algorithm 4 is benchmarked against the optimal C^2RM without spectrum sharing in Algorithm 2. Note that the number of devices are set to be K = 20 with high heterogeneity regarding their computation efficiencies and channels. The curves of sum energy versus the bandwidth B are plotted in Fig. 4. Two observations can be made. First, the sum energy reduces as B grows as more bandwidths can be traded for lower transmission power. Second, it can be found that the proposed greedy spectrum sharing policy improves the energy efficiency of the C^2RM without spectrum sharing by scavenging unused radio resources. For example, it reduces sum energy of the baseline scheme, namely optimal C^2RM without spectrum sharing by 10.6% for uplink bandwidth equal to 2.5 MHz.

C. Device Scheduling

Consider the scenario that the edge server schedules a subset of devices for learning. The number of rounds is fixed as 10 with constrained latency T = 1 s for each round. The performance of the proposed C^2-aware scheduling is benchmarked against two baselines: 1) best-M channels selection and 2) the random selection. The effects of the number of scheduled devices M on the average learning accuracy of the FEEL algorithm and sum energy consumption are shown in Fig. 5(a). Several observations can be made as follows. First, the average learning accuracy is an increasing function of M as it increases the training data per round. Second, it can be observed that the increase on M leads to the growth of sum energy at a rate faster than linear, which agrees with the preceding discussion. Furthermore, one can observe that the proposed scheduling scheme outperforms the baseline which randomly selects M devices, e.g., achieving 45.1% sum energy reduction for M = 35. To present the results more clearly, we define the optimality gap for the baseline (either best-M channels selection or random selection) as \( \frac{E_{\text{baseline}} - E_{\text{proposed}}}{E_{\text{proposed}}} \times 100\% \), where \( E_{\text{proposed}} \) and \( E_{\text{baseline}} \) denote the sum energy consumption with the solution from C^2 aware scheduling and the baseline, respectively. It can be observed from Fig. 5(b) that the proposed C^2 aware scheduling also outperforms the baseline which selects M devices with best channels, e.g., the resultant optimality gap is up to 135% for M = 5. It is also worth mentioning that when the number of participants becomes large, the communication energy becomes the bottleneck due to the constrained total bandwidth. Therefore, both the proposed C^2 aware scheduling and best-M channels method tend to select the same subset of devices in this scenario.

VII. CONCLUDING REMARKS

In this paper, we have proposed a set of schemes for joint C^2RM to enhance the energy-efficiency of a FEEL system using CPU-GPU heterogeneous computing. This work paves the way for followup works to address several interesting
Then, it is also true for the case \( \frac{W_k}{T_k} < \frac{W'_{k'}}{T_{k'}} \).

### B. Proof of Lemma 2

The computation energy of device \( k \) can be expressed as

\[
E_{k,\text{CPU}}^{\text{comp}} = E_{k,\text{CPU}}^{\text{CPU}}(W_k, t_k') + E_{k,\text{GPU}}^{\text{GPU}}(W'_{k'}, t'_{k'})
\]

where \( \{\lambda_k \geq 0\} \) and \( \{\theta_k \geq 0\} \) are the Lagrange multipliers. Then we have the conditions:

\[
\frac{\partial \mathcal{L}}{\partial W_k} = \frac{\partial E_{k,\text{CPU}}}{\partial W_k} + \gamma_k - \lambda_k = 0, \quad \forall k \in \mathcal{K},
\]

\[
\frac{\partial \mathcal{L}}{\partial W'_{k'}} = \frac{\partial E_{k,\text{GPU}}}{\partial W'_{k'}} + \gamma_k - \theta_k = 0, \quad \forall k \in \mathcal{K}.
\]

The complementary slackness conditions give that

\[
\lambda_k W_k = 0, \quad \theta_k W'_{k'} = 0, \quad \forall k \in \mathcal{K}.
\]

Since at least one processor should be active. Assume \( W'_{k'} > 0 \) and then we have \( \theta_k = 0 \) from the second condition in (44). Since the multiplier \( \lambda_k \) is required to be \( \lambda_k \geq 0 \), we know

\[
\lambda_k = \frac{\partial E_{k,\text{CPU}}}{\partial W_k} - \frac{\partial E_{k,\text{GPU}}}{\partial W'_{k'}} \geq 0, \quad \forall k \in \mathcal{K}.
\]

Substituting results (12) of Lemma 2 into (45), we have

\[
W_k \geq \sqrt{\frac{C_k}{G_k}} W'_{k'} > 0, \quad \forall k \in \mathcal{K}.
\]

Accordingly, we have \( W_k > 0 \) so that \( \lambda_k = 0 \). Therefore, it is optimal for both processors to be active with the energy-workload rate equilibrium (13) as stated in Lemma 2.

### C. Proof of the claim in Remark 2

We prove the fact that the proposed CPU-GPU heterogeneous computing scheme with policy in Proposition 1 is more energy-efficient than just using GPU to complete workload \( W \) within the allowed computation time \( T_k' \). Based on (14) and (15), the optimal computation energy consumption of CPU-GPU heterogeneous computing is

\[
E_{k,\text{CPU-GPU}}^{\text{comp}} = C_k W_k^* f_k^* W'_{k'} f'_{k'} = \frac{C_k G_k W^3}{(\sqrt{C_k} + \sqrt{G_k})^2 T_k'^2}.
\]

As for the baseline that the GPU takes the total workload \( W \), it is not hard to derive the optimal speed scaling rule as \( f_k^* = \frac{W_k}{T_k'} \) and the optimal computation energy is

\[
E_{k,\text{GPU}}^{\text{comp}} = G_k W_k^* W'_{k'} f_k^* = \frac{G_k W^3}{T_k'^2}.
\]

Then, we can derive the difference between them as follows:

\[
E_{k,\text{CPU-GPU}}^{\text{comp}} - E_{k,\text{GPU}}^{\text{comp}} = -\frac{(\sqrt{G_k} W * W^3)}{(\sqrt{C_k} + \sqrt{G_k})^2 T_k'^2} < 0.
\]

Therefore, we can conclude that \( E_{k,\text{CPU-GPU}}^{\text{comp}} < E_{k,\text{GPU}}^{\text{comp}} \), which means the considered CPU-GPU heterogeneous computing is a more energy-efficient scheme.
D. Proof of Lemma 3

Substituting $t_k^* = T_k$ into (10), the communication energy of device $k$ can thus be expressed as the function of allocated bandwidths, i.e., $E^{mm}_k(B_k)$. By introducing Lagrange multipliers $\{\mu_k\}$ for the inequality constraints $\{B_k \geq 0\}$ and a scalar multiplier $\nu^*$ for the equality constraint $\sum_{k=1}^K B_k = B$, the Lagrangian function is defined as
\[
\mathcal{L}(\{B_k\}, \{\mu_k\}, \nu) = \sum_{k=1}^K [E^{mm}_k(B_k) + \mu_k B_k] + \nu \left( \sum_{k=1}^K B_k - B \right),
\]
(49)

Then we have the following conditions:
\[
B_k^* \geq 0, \quad \sum_{k=1}^K B_k^* = B, \quad \mu_k \geq 0, \quad \mu_k B_k^* = 0, \quad \forall k \in \mathcal{K},
\]
\[
\left. \frac{\partial E^{mm}_k}{\partial B_k} \right|_{B_k^*} - \mu_k + \nu^* = 0, \quad \forall k \in \mathcal{K}.
\]
(50)

The feasible requirement gives $B_k^* > 0, \forall k \in \mathcal{K}$, so that $\mu_k^* = 0$. Therefore, $\left. \frac{\partial E^{mm}_k}{\partial B_k} \right|_{B_k^*} = -\nu^*, \quad \forall k \in \mathcal{K}$.

E. Proof of Lemma 4

To begin with, we introduce several basic properties: 1) $B_k$ is a decreasing convex function of $\nu$ with the form in Lemma 2; 2) $\nu_k$ is a decreasing and strictly convex function of $B_k$ with the form in (16); 3) $\nu_k(B_k) > 0$ for all $k$ and all value of $B_k > 0$. The proofs are straightforward and omitted for brevity.

Assume the point $\nu^{(i-1)} > \nu^*$, it follows from property 1) that $B_k(\nu^{(i-1)}) < B_k^*, \forall k \in \mathcal{K}$. Denote $B_k^{(i)} \triangleq B_k(\nu^{(i-1)})$, and it holds that $\sum_{k=1}^K B_k^{(i)} < \sum_{k=1}^K B_k^* = B$. Therefore, $\text{sgn} \left( \sum_{k=1}^K B_k^{(i)} - B \right) = -1$ and $B_k^{(i)} = \frac{B}{\sum_{k=1}^K B_k^{(i)}}$. By property 2), we know that $\nu_k(B_k^{(i)}) < \nu_k(B_k) = \nu^{(i-1)}, \forall k \in \mathcal{K}$. Then, we have $\max_k \{\nu_k(B_k^{(i)})\} < \nu^{(i-1)}$.

Next, we prove by contradiction that $\max_k \{\nu_k(B_k^{(i)})\} < \nu^*$ as follows. First, we assume that $\max_k \{\nu_k(B_k^{(i)})\} < \nu^*$ and one can have $\nu_k(B_k^{(i)}) < \nu^*, \forall k \in \mathcal{K}$. Accordingly, we have $B_k^{(i)} > B_k^*, \forall k \in \mathcal{K}$, which implies that $\sum_{k=1}^K B_k^{(i)} > \sum_{k=1}^K B_k^* = B$. However, it is invalid as $\sum_{k=1}^K B_k^{(i)} = \sum_{k=1}^K B_k^* = B$. Thus the earlier assumption is false. Thereby, we have $\max_k \{\nu_k(B_k^{(i)})\} > \nu^*$.

Following the same procedure, we can derive for the case of $\nu^{(i-1)} < \nu^*$, that $y = \text{sgn} \left( \sum_{k=1}^K B_k^{(i)} - B \right) = 1$ and $\nu^{(i-1)} < \min_k \{\nu_k(B_k^{(i)})\} < \nu^*$.

REFERENCES

Yuqing Du received the B.Eng. degree from Harbin Engineering University (HEU) in 2016 and the Ph.D. degree from The University of Hong Kong (HKU) in 2020, both in electrical engineering. His research interests include edge learning, distributed machine learning.

Kaibin Huang (Fellow, IEEE) received the B.Eng. and M.Eng. degrees from the National University of Singapore, and the Ph.D. degree from The University of Texas at Austin, all in electrical engineering. Presently, he is an associate professor in the Dept. of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong. He received the IEEE Communication Society’s 2021 Best Survey Paper, 2019 Best Tutorial Paper, 2019 Asia Pacific Outstanding Paper, 2015 Asia Pacific Best Paper Awards as well as Best Paper Awards at IEEE GLOBECOM 2006 and IEEE/CIC ICC 2018. Moreover, he received an Outstanding Teaching Award from Yonsei University in S. Korea in 2011. He has served as the lead chairs for the Wireless Comm. Symp. of IEEE GLOBecom 2017 and the Comm. Theory Symp. of IEEE GLOBECOM 2014 and the TPC Co-chairs for IEEE PMRC 2017 and IEEE CTW 2013. He is an Executive Editor for IEEE Transactions on Wireless Communications, an Associate Editor for Journal on Selected Areas in Communications (JSAC), and an Area Editor for IEEE Transactions on Green Communications and Networking. Previously, he has also served on the editorial board of IEEE Wireless Communications Letters. Moreover, he has guest edited special issues for IEEE JSAC, IEEE Journal on Selected Topics on Signal Processing, and IEEE Communications Magazine. He is an IEEE Fellow and an IEEE Distinguished Lecturer of both the IEEE Communications and Vehicular Technology Societies. He has been named a Highly Cited Researcher by Clarivate Analytics in 2019 and 2020.

Kin K. Leung received his B.S. degree from the Chinese University of Hong Kong in 1980, and his M.S. and Ph.D. degrees from University of California, Los Angeles, in 1982 and 1985, respectively. He worked at AT&T Bell Labs and its successor companies in New Jersey from 1986 to 2004. Since then, he has been the Tanaka Chair Professor in the Electrical and Electronic Engineering (EEE), and Computing Departments at Imperial College in London. He is the Head of Communications and Signal Processing Group in the EEE Department. His current research focuses on optimization and machine learning for design and control of large-scale communication, computer and sensor infrastructures. He also works on multi-antenna and cross-layer designs for wireless networks. He received the Distinguished Member of Technical Staff Award from AT&T Bell Labs (1994), and was a co-recipient of the Lanchester Prize Honorable Mention Award (1997). He was elected an IEEE Fellow (2001), received the Royal Society Wolfson Research Merits Award (2004-09), and became a member of Academia Europaea (2012) and an IET Fellow (2021). Jointly with his co-authors, he received the IEEE ComSoc Leonard G. Abraham Prize (2021) and best paper awards at the IEEE PMRC 2012, ICDCS 2013 and ICC 2017. In 2019, he served as a member (2009-11) and the chairman (2012-15) of the IEEE Fellow Evaluation Committee for ComSoc. He served as an editor for the IEEE JSAC: Wireless Series, IEEE Trans. on Wireless Communications and IEEE Trans. on Communications. Currently, he chairs the Steering Committee for the IEEE Trans. on Mobile Computing, and is an editor for the ACM Computing Survey and International Journal on Sensor Networks.