Joint Parameter-and-Bandwidth Allocation for Improving the Efficiency of Partitioned Edge Learning

Dingzhu Wen, Mehdi Bennis, *Senior Member, IEEE,* and Kaibin Huang, *Senior Member, IEEE,*

***Abstract*—To leverage data and computation capabilities of mobile devices, machine learning algorithms are deployed at the network edge for training *artificial intelligence* (AI) models, resulting in the new paradigm of edge learning. In this paper, we consider the framework of *partitioned edge learning* for iteratively training a large-scale model using many resource-constrained devices (called workers). To this end, in each iteration, the model is dynamically partitioned into parametric blocks, which are downloaded to worker groups for updating using data subsets. Then, the local updates are uploaded to and cascaded by the server for updating a global model. To reduce resource usage by minimizing the total learning-and-communication latency, this work focuses on the novel joint design of parameter (computation load) allocation and bandwidth allocation (for downloading and uploading). Two design approaches are adopted. First, a practical sequential approach, called partially integrated *parameter-and-bandwidth allocation* (PABA), yields two schemes, namely *bandwidth aware parameter allocation* and *parameter aware bandwidth allocation*. The former minimizes the load for the slowest (in computing) of worker groups, each training a same parametric block. The latter allocates the largest bandwidth to the worker being the latency bottleneck. Second, PABA are jointly optimized. Despite it being a *nonconvex* problem, an efficient and optimal solution algorithm is derived by intelligently nesting a bisection search and solving a *convex* problem. Experimental results using real data demonstrate that integrating PABA can substantially improve the performance of partitioned edge learning in terms of latency (by e.g., 46%) and accuracy (by e.g., 4% given the latency of 100 seconds).**

I. Introduction

An enormous amount of data and computation resources are distributed over a large number of mobile devices [1]. This motivates the deployment of distributed machine learning algorithms at the network edge for fast and scalable model training. As a result, a new paradigm of computing, called edge machine learning, has emerged as an attractive and fast growing research area [2], [3]. In this work, we consider *partitioned edge learning* (PARTEL) for large-scale model training. In the framework, a model is partitioned into parametric blocks that are downloaded onto devices for distributed updating and the computation results are uploaded to a server for updating the global model [4]. This work aims at improving the efficiency of PARTEL by minimizing the total communication-and-learning latency, thereby reducing the resource utilization. Under this objecive, a set of novel schemes are proposed by jointly designing the model-partitioning based load distribution over devices and bandwidth allocation for their downloading and uploading.

The work of K. Huang and D. Wen was supported by Hong Kong Research Grants Council under Grants 17208319 and 17209917, Innovation and Technology Fund under Grant GHP/016/18GD, and Guang-dong Basic and Applied Basic Research Foundation under Grant 2019B1515130003. The work of M. Bennis was supported by EU-CHISTERA projects, CONNECT and LeadingEdge. (Corresponding author: Kaibin Huang).

D. Wen and K. Huang are with The University of Hong Kong, Hong Kong. M. Bennis is with University of Oulu, Finland. (Corresponding email: huangkb@eee.hku.hk).

A. Partitioned Edge Learning

1) Partitioned Learning: A representative framework for partitioned distributed learning is called parameter server, which is proposed in [4], [5] to distribute a large-scale learning task over many resource-constrained machines by *model partitioning*. Implementing the classic method of *block coordinate descent* (BCD)[6], the framework iteratively and distributively solves a large-scale model-optimization problem with a decomposable objective function [e.g., linear regression and *support vector machine* (SVM)]. Specifically, a parameter server divides the model parameters into blocks and update each block in one iteration, called a *communication round*. In each round, the server further divides and distributes the global dataset over workers so that they can locally compute the block gradients for updating the downloaded parametric block under their resource constraints. Then, the local gradients are uploaded to the server for aggregation (e.g., averaging) and updating the particular parametric block of the global model, completing the round. The framework is further developed in a series of work to reduce the learning latency by allowing overlapping of consecutive rounds [7] or workers to use a staled parametric block for computing their updates [8].

Recent research on partitioned learning focus on *convolutional neural network* (CNN) models. Such a model comprising nested layers does not have a decomposable (learning) loss function, making direct model partitioning sub-optimal. Overcoming the limitation has driven researchers to extend the BCD method to CNN [9], [10]. Specifically, by introducing and conditioning on auxiliary variables, the layers in a CNN become conditionally independent and thus can be trained separately in different rounds. The cost due to a complex model architecture is that updating each layer (also a parametric block) requires multiple rounds instead of one as in the parameter-server framework with a decomposable function.

2) Edge Implementation: While communication channels are abstracted as bit pipes in prior work, PARTEL concerns the design of new communication techniques for efficient implementation of partitioned learning in wireless networks. This is an uncharted area and the theme of this work. Connecting parameter servers with workers (edge devices) using wireless links gives rise to two challenges: 1) overcoming channel impairments (fading and noise) and scarcity of radio resources and 2) leveraging a massive number of resource-constrained

devices for performing a single large-scale learning task. The direct application of traditional communication techniques may not be sufficient given the excessive communication overhead caused by large-scale model (with millions to billons of parameters) and large-scale dataset (typically comprising millions of high-dimensional multimedia samples). In this work, we adopt the new approach of integrated computation-and-communication design. Specifically, the model-and-data partitioning, computation load allocation, and radio resource allocation are jointly designed so as to reduce the learning-and-communication latency, thereby minimizing the resource utilization.

B. Federated Edge Learning

1) Federated Learning: Another mainstream framework for distributed learning, called *federated learning*, was developed for the purpose of leveraging local data generated at edge devices while preserving their data privacy by avoiding direct data uploading [11]. The framework similar to parameter server but simpler as it involves no model or dataset partitioning. Implementing *stochastic gradient descent* (SGD), federated learning requires each device to download and locally update the *whole* model (or compute the needed gradient) and all devices to upload the computed models/gradients to update a global model after model/gradient aggregation at an edge server; the procedure iterates till the model converges. Communication efficiency is a main research theme in the area as excessive communication overhead is incurred by the repeated uploading of high-dimensional local models/gradients by many devices over many rounds. One approach is to reduce the number of uploading devices by allowing infrequent uploading by devices slow in computation [12], or selecting those whose results are relatively more important for learning (see e.g. [13], [14]). Another approach is to directly compress local gradients exploiting their sparsity [15].

2) Edge Implementation: Driven by the vision of edge intelligence, the new area of *federated edge learning* (FEEL) has emerged, focusing on efficient implementations of federated learning in wireless networks. Based on the approach of communication-and-learning integration, many techniques are designed for efficient transportation of high-dimensional data over wireless channels. New multi-access schemes for FEEL, called "over-the-air computing", are proposed in [16]–[18] to support fast "over-the-air" model/gradient aggregation using the waveform superposition property of a multi-access channel. Another vein of research addresses the issue of *radio resource management* (RRM) in FEEL systems such as bandwidth allocation [19], multiuser scheduling [20], and their joint design [21], [22]. Joint RRM and training batch-size selection is further investigated in [23] to accelerate the learning speed in FEEL systems. From the perspective of FEEL system performance, there exists a fundamental tradeoff between device energy consumption and learning speed, which is quantified in [24]. In addition, in view of the varying communication-and-computation capacities of different nodes, researchers have also developed a hierarchical network architecture for implementing large-scale FEEL [25].

3) Federated vs. Partitioned Edge Learning: The main objective of FEEL is to exploit users' data without violating their privacy. The framework does not involve model partitioning and requires each edge device to update a whole model. Since the devices are resource constrained, FEEL is suitable for *small-to-medium* learning tasks. In contrast, the objective of PARTEL is to train a *large-scale* model using many edge devices as workers via model partitioning. Therefore, the design of efficient PAETEL requires the integration of the partitioning for load allocation with radio resource allocation, which is a new challenge not faced in the area of FEEL.

C. Contributions and Organization

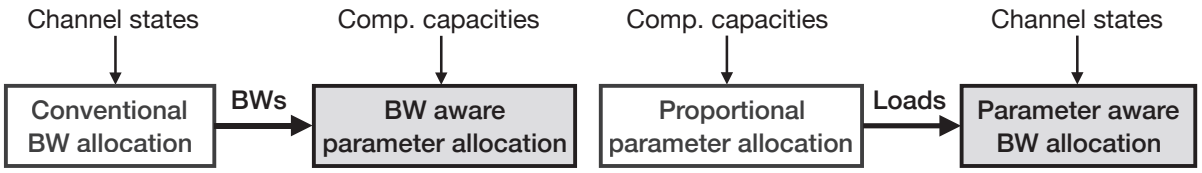
In this paper, we consider a single-cell wireless system supporting PARTEL. In the system, the workers are grouped and each group is responsible for updating an assigned parametric block. Within one group, the global dataset is distributed over workers so that each resource-constrained worker need compute the block update using only a data subset. In each communication round, an edge server coordinates the learning process by performing the following operations:

1) *Parameter allocation*, referring to partitioning the model into parametric blocks for load allocation;
2) *Bandwidth allocation*, namely partitioning the bandwidth for downloading latest values of parameters to workers and uploading their updates on parametric blocks;
3) Cascading the uploaded block updates to update the global model.

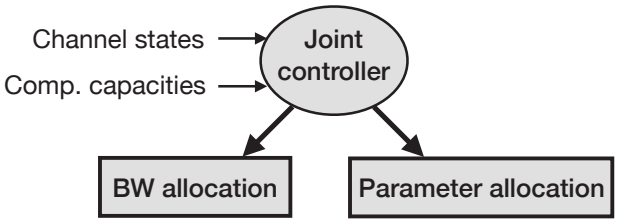
Such coordinated distributed learning introduces the constraint of *synchronized updates* by devices. The rounds are repeated till the model converges.

One way for improving the efficiency of PARTEL is to minimize the total (communication plus computation) latency so as to minimize the utilization of radio and device-computation resources. Under the constraint of synchronized updates, the total latency depends on both the communication and computation latency of all devices, which can be controlled by bandwidth and parameter allocation, respectively. This motivates the current work to make the first attempt on jointly designing *parameter-and-bandwidth allocation* (PABA) for PARTEL systems. Our approach is to formulate and solve latency minimization problems for optimizing the PABA policy for given heterogeneous channel states and device-computation capacities. The specific contributions and findings are summarized as follows.

First, practical PABA schemes are proposed for the scenario of large-scale network with fast varying channels. In this scenario, the direct optimization of joint PABA is a challenging problem, which is non-convex with many variables and requiring an iterative solution method (see the second contribution). Moreover, the task of solving the problem has to be repeated whenever the channels change. To overcome the difficulty, we propose the practical *partially integrated PABA* where the designs of the two functional blocks are sequential: the first block (either parameter or bandwidth allocation) is designed independently of the other and the second block is



(a) Partially integrated PABA



(b) Joint PABA

Fig. 1. Two approaches of designing parameter-allocation and bandwidth-allocation blocks: (a) partially integrated design and (b) joint design.

designed conditioned on the first. This results in two simple PABA schemes, summarized as follows.

- **Bandwidth aware parameter allocation**: Consider the optimization of the parameter allocation conditioned on bandwidth allocation using a conventional scheme [see the left of Fig. 1(a)]. The optimization is shown be a linear program, allowing the optimal policy to be derived in closed form. The policy is found to minimize the load for the worker group slowest in computation. To be precise, the parametric-block length assigned to one group is inversely proportional to its slowest worker's total latency.
- **Parameter aware bandwidth allocation**: Next, reversing the design order [see the right of Fig. 1(a)] yields the current scheme. By analyzing and exploiting the problem structure, solving the latency optimization problem is reduced to a simple bisection search. The resultant optimal policy for parameter aware bandwidth allocation is found to allocate the largest bandwidth to the worker being the latency bottleneck to alleviate the bottleneck.

Next, targeting slowly varying channels, we develop an efficient iterative algorithm for solving the problem of joint PABA optimization, called *fully integrated PABA*, as illustrated in Fig. 1(b). To this end, a useful property is derived that the optimal policy equalizes the *group bandwidth allocation rates*, defined as the additional bandwidth required by assigning one additional parameter to the group for updating. Leveraging the property, an efficient solution method is derived that intelligently nests a bisection search and solving a *convex* problem. To gain further insights, two special cases with single-worker groups or intra-group uniform computation capacities are considered. The optimal polices are derived in simple form and aligned with intuition (e.g., allowing more load to a group with better computation capacity).

The remainder of the paper is organized as follows. In Section II, the system model is introduced. In Section III, the total-latency minimization problem is formulated and simplified. In Section IV and V, two schemes of partially integrated PABA are designed while the scheme of fully integrated PABA is derived in Section VI. Section VII presents the experimental results followed by concluding remarks in Section VIII.

II. Models and Metrics

A. System Model

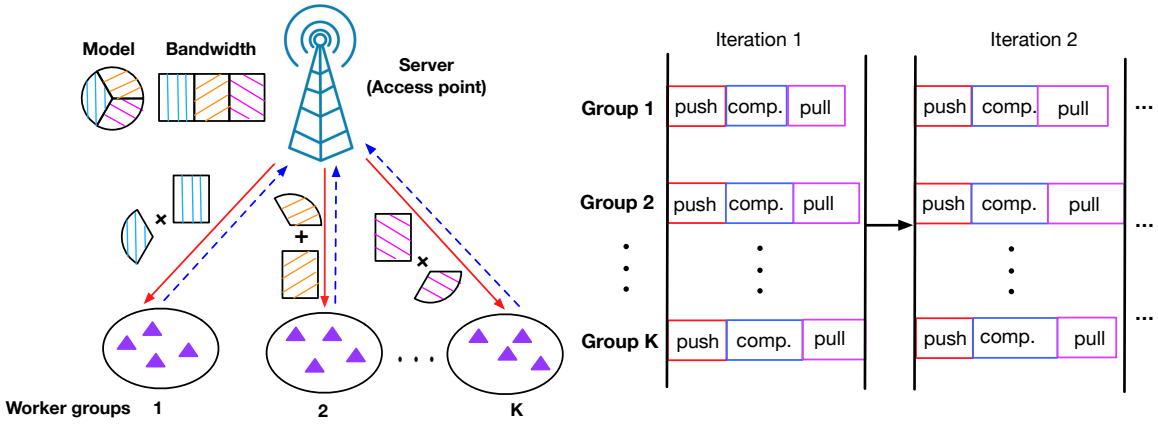
A single-cell system is considered, as illustrated in Fig. 2(a). In the cell, there are a server equipped with a single-antenna *access point* (AP) and multiple single-antenna edge devices, serving as workers. The workers are divided into K groups, identified by the index set $\mathcal{G}_1$, $\mathcal{G}_2$, ..., $\mathcal{G}_K$, each of which collaboratively performs one task. The n-th worker in group $\mathcal{G}_k$ is denoted as (k, n). The server is connected to workers via wireless links. For simplicity, the channels are assumed to be static in one iteration of model training and vary over different iterations. We assume that the AP has the *channel state information* (CSI) of all links that are useful for bandwidth allocation. The uplink/downlink spectrum is divided into orthogonal frequency non-selective channels, each of which is assigned to one worker. The downlink and uplink channel gains of worker (k, n) are denoted as $H_{\mathrm{d},k,n}$ and $H_{\mathrm{u},k,n}$, respectively.

B. Learning Model

The PARTEL framework is designed for a large-scale learning task with a decomposable objective function. As mentioned, this is natural for algorithms such as SVM and logistic regression [6] and can be made feasible for CNN models using the method of auxiliary variables [9], [10]. Following the literature, a decomposable objective function has the following form (see e.g., [6]):

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{F}(\boldsymbol{\theta}) + \mathcal{R}(\boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta} = \{\theta_1, ..., \theta_{n_\mathrm{p}}, ...\theta_{N_\mathrm{p}}\}^T$ is the parameter vector of the learning model, $\mathcal{F}(\boldsymbol{\theta})$ is the loss function, and $\mathcal{R}(\boldsymbol{\theta})$ is the block-separable regularized function (e.g., ℓ_1 and ℓ_2 regularizations) to reduce the overfitting or increase the sparsity of



(a) PARTEL Network Architecture (b) Iterative Learning Process

Fig. 2. System model and operations of the PARTEL framework.

the trained learning model, respectively. Specifically, the loss function can be written as $\mathcal{F}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\phi(\boldsymbol{\theta};\mathbf{x}_m)$, where $\mathcal{X}$ = $\{\mathbf{x}_m\}$ is the dataset, M is the size of the dataset, and $\phi(\cdot)$ is a smooth function. And the block-separable regularized function can be written as $\mathcal{R}(\boldsymbol{\theta}) = \sum_{n_\mathrm{p}=1}^{N_\mathrm{p}}\psi(\theta_{n_\mathrm{p}})$, where θ_{n_p} is the n_p-th element of $\boldsymbol{\theta}$ and N_p is the total number of parameters. During training, if the regularization function $\mathcal{R}(\cdot)$ is smooth, gradient descent can be used for updating the learning model [26]. Otherwise, the learning model is updated by another method, called proximal gradient descent [27].

C. PARTEL Architecture

Consider the network architecture in Fig. 2(a). The global dataset is partitioned at the server and downloaded by workers such that each worker loads a data subset and each worker group has the whole dataset. The model-parameter vector is partitioned into K disjoint parametric blocks, as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k, ..., \boldsymbol{\theta}_K\}$, where $\boldsymbol{\theta}_k$ is assigned to group $\mathcal{G}_k$ for updating. One main benefit of PARTEL is that each resource-constrained worker only needs to calculate and transmit the gradient or proximal gradient of a parametric block over a data subset instead of the whole parameter vector during each iteration. Taking the case of smooth regularization function as an example, only the following block of gradient elements is required for computation and transmission by worker (k, n),

$$\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_{n_\mathrm{p}}} = \frac{1}{M}\sum_{m=1}^{M}\frac{\partial\phi(\boldsymbol{\theta};\mathbf{x}_m)}{\partial\theta_{n_\mathrm{p}}} + \frac{1}{|\mathcal{G}_k|}\frac{\mathrm{d}\psi(\theta_{n_\mathrm{p}})}{\mathrm{d}\theta_{n_\mathrm{p}}},\ \forall\theta_{n_\mathrm{p}} \in \boldsymbol{\theta}_k, \quad (2)$$

where $\mathcal{X}_{k,n}$ is the data subset at worker (k, n), $|\mathcal{G}_k|$ represents the number of workers in group $\mathcal{G}_k$, $\frac{\psi(\theta_{n_\mathrm{p}})}{|\mathcal{G}_k|}$ represents the regularization function for each worker, and $\boldsymbol{\theta}_k$ with the size of b_k is the parametric block assigned to group $\mathcal{G}_k$, respectively.

In the PARTEL framework, one training iteration of the learning model is called *one (communication) round*. In each round, the server first shares the whole model-parameter vector $\boldsymbol{\theta}$ to all workers. Then, the gradient or proximal gradient of each parametric block is calculated by one worker group. Finally, the gradients or proximal gradients of all groups are uploaded to the server to update all parameters. Wireless links are used for sharing the whole model parameters and uploading the gradients. Thereby, to train the distributed learning algorithms in PARTEL framework, there are three steps in one round, as follows.

- *Push*: The server (AP) broadcasts the whole model parameters $\boldsymbol{\theta}$ to all workers.
- *Computation*: Each worker computes the gradient or the proximal gradient of the its assigned parametric block based on its loaded data subset.
- *Pull*: All workers upload the gradients or proximal gradients of their corresponding parametric blocks to the server. The server aggregates the gradients or proximal gradients from all groups and updates the corresponding parametric block.

The iterative process of the distributed learning algorithms is shown in Fig. 2(b). From the figure, synchronized updates are required in each round. Synchronized updates arise from the operation of gradient aggregation in the pull step 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) for uploading their local gradients and thus synchronized in update transmission. Hence, the latency in the round is decided by the "slowest" worker. In the sequel, the latency of any one round is defined.

Remark 1 (Learning Convergence Speed)**.** With model updating per round, the distributed learning using the PARTEL architecture is optimal in the sense of achieving the same learning performance as the centralized learning within a same number of rounds (see Lemma 1). However, the implementation of PARTEL in wireless network makes it necessary to measure the learning duration/latency in second. The reason is that finite radio bandwidth and resource-constrained workers cause significant latency of communication and computation in each round. Therefore, optimizing parameter allocation and bandwidth allocation is important for ensuring fast model convergence as in achieving targeted learning performance within a given duration measured in second.

Remark 2 (Relation with FEEL)**.** In the case of only one worker group and hence no model partitioning, the PARTEL architecture reduces to that of FEEL (with uploading per round), as all workers calculate the gradient of the whole parameter vector over a data subset.

D. Latency Model

Consider an arbitrary communication round, say the r-th round, and an arbitrary worker, say worker (k,n). The latency, denoted as $t_{k,n}^{(r)}$, is composed of three parts:

$$t_{k,n}^{(r)} = T_{\rm ph}^{(r)} + \hat{t}_{k,n}^{(r)} + t_{{\rm pl},k,n}^{(r)}, \tag{3}$$

where $T_{\rm ph}^{(r)}$, $\hat{t}_{k,n}^{(r)}$, and $t_{{\rm pl},k,n}^{(r)}$ correspond to the three steps, namely push, computation, and pull, respectively.

1) Push latency: The push latency is defined as the time for the server to broadcast the whole parameter vector $\boldsymbol{\theta}$ to all workers, which is given by

$$T_{\rm ph}^{(r)} = \max_{\{(k,n)\}} \frac{A_{\rm p} N_{\rm p}}{BR_{{\rm d},k,n}^{(r)}}, \tag{4}$$

where $A_{\rm p}$ is the number of bits per model parameter, $N_{\rm p}$ is the total number of parameters, B is the system bandwidth, and $R_{{\rm d},k,n}^{(r)}$ is the downlink spectrum efficiency of worker (k,n). The efficiency can be written as $R_{{\rm d},k,n}^{(r)} = \log_2\left(1+P_{\rm b}H_{{\rm d},k,n}^{(r)}/N_0\right)$, where $P_{\rm b}$ is the transmission power of the AP, $H_{{\rm d},k,n}^{(r)}$ is the downlink channel gain, and N_0 is the channel noise variance. Note that the push latency is a constant identical for all workers.

2) Computation latency: Denote the number of computation operations to calculate the gradient with respect to one parameter using one data sample as O, the number of data samples loaded by worker (k,n) as $D_{k,n}$, and the CPU frequency of worker (k,n) as $f_{k,n}^{\rm c}$. Then, the computation latency of worker (k,n) is a function of its assigned load $b_k^{(r)}$, i.e., the length of its assigned parametric block $\boldsymbol{\theta}_k^{(r)}$. It can be written as

$$\hat{t}_{k,n}^{(r)}\left(b_k^{(r)}\right) = \frac{b_k^{(r)} D_{k,n} O}{f_{k,n}^{\rm c}}. \tag{5}$$

3) Pull latency: The pull latency consists of two parts. One is the time for worker (k,n) to upload the gradients to the server, denoted as $\tilde{t}_{k,n}^{(r)}$. The other is the time for the server to update the learning model, denoted as $T_{\rm s}^{(r)}$. Hence, the pull latency is given by

$$t_{{\rm pl},k,n}^{(r)} = \tilde{t}_{k,n}^{(r)} + T_{\rm s}^{(r)}. \tag{6}$$

The time $T_{\rm s}^{(r)}$ is the same for all workers. The uploading time $\tilde{t}_{k,n}^{(r)}$ is given by

$$\tilde{t}_{k,n}^{(r)}\left(b_k^{(r)}, \rho_{k,n}^{(r)}\right) = \frac{A_{\rm g} b_k^{(r)}}{\rho_{k,n}^{(r)} BR_{{\rm u},k,n}^{(r)}}, \tag{7}$$

where $A_{\rm g}$ is the number of bits for each gradient element, $b_k^{(r)}$ is the assigned parametric-block length, $\rho_{k,n}^{(r)}$ is the ratio of uplink bandwidth allocated to worker (k,n), and $R_{{\rm u},k,n}^{(r)}$ is the uplink spectrum efficiency. The efficiency is $R_{{\rm u},k,n}^{(r)} = \log_2\left(1+P_{\rm u}H_{{\rm u},k,n}^{(r)}/N_0\right)$, where $P_{\rm u}$ is the uplink transmission power and $H_{{\rm u},k,n}^{(r)}$ is the uplink channel gain.

We define the group latency in the r-th round as follows. Since all parameters should be updated in the round, the group latency is decided by the "slowest" worker. The latency of group $\mathcal{G}_k$ is thus given as:

$$t_k^{(r)}\left(b_k^{(r)}, \{\rho_{k,n}^{(r)}\}\right) = \max_{n\in\mathcal{G}_k} t_{k,n}^{(r)}\left(b_k^{(r)}, \rho_{k,n}^{(r)}\right), \tag{8}$$

where $t_{k,n}^{(r)}\left(b_k^{(r)}, \rho_{k,n}^{(r)}\right)$ is the latency of worker (k,n) defined in (3) and its three components are described in (4), (5), and (6).

Next, we define the total latency in the r-th round. Given synchronized updates, the total latency in this communication round depends on the "slowest" group:

$$t^{(r)}\left(\{b_k^{(r)}\}, \{\rho_{k,n}^{(r)}\}\right) = \max_k t_k^{(r)}\left(b_k^{(r)}, \{\rho_{k,n}^{(r)}\}\right). \tag{9}$$

III. Problem Formulation and Simplification

Based on the models described in the preceding section, the problem of learning-latency minimization is formulated in this section under two constraints, one on the total number of parameters and the other on the total bandwidth. Then, the problem is simplified as an equivalent one-round problem.

A. Problem Formulation

The learning latency depends on two factors. One is the *number of communication rounds required for model convergence*, denoted as R, and the other is the *per-round latency*. Thus, the total learning latency is given as

$$t_{\rm learn}\left(\{b_k^{(r)}\}, \{\rho_{k,n}^{(r)}\}\right) = \sum_{r=1}^{R} t^{(r)}\left(\{b_k^{(r)}\}, \{\rho_{k,n}^{(r)}\}\right), \tag{10}$$

where $t^{(r)}\left(\{b_k^{(r)}\}, \{\rho_{k,n}^{(r)}\}\right)$ is the latency of the r-th round defined in (9). We aim at minimizing the learning latency by optimizing the distribution of parametric blocks, or called parameter allocation, and the bandwidth allocation. Parameter allocation must satisfy the following constraints on the total number of parameters:

$$\text{(C1.1)} \quad \begin{cases} \sum_{k=1}^{K} b_k^{(r)} = N_{\rm p},\ 1\le r\le R, \\ b_k^{(r)} \in \mathbb{Z}^+,\ \forall k,\ 1\le r\le R, \end{cases} \tag{11}$$

where $N_{\rm p}$ is the total number of parameters and $b_k^{(r)}$ is length of the parametric block assigned to group $\mathcal{G}_k$ in the r-th round. On the other hand, the bandwidth allocation should satisfy the following constraints on the total bandwidth:

$$\text{(C1.2)} \quad \begin{cases} \sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k} \rho_{k,n}^{(r)} \le 1,\ 1\le r\le R, \\ \rho_{k,n}^{(r)} \ge 0,\ \forall (k,n),\ 1\le r\le R, \end{cases} \tag{12}$$

where $\rho_{k,n}^{(r)}$ is the ratio of the bandwidth allocated to worker (k,n) in the r-th round. Under the constraints, the problem of learning-latency minimization can be formulated as

$$\begin{aligned} \text{(P1)} \quad & \min_{\{b_k^{(r)}\},\{\rho_{k,n}^{(r)}\}} \sum_{r=1}^{R} t^{(r)}\left(\{b_k^{(r)}\}, \{\rho_{k,n}^{(r)}\}\right) \\ & \text{s.t. (C1.1) \& (C1.2).} \end{aligned} \tag{13}$$

In the sequel, we prove that (P1) can be reduced to an equivalent one-round problem.

B. Equivalent One-Round Latency Minimization

As shown in the following lemma, the convergence rates (in rounds) of the learning algorithms implemented at PARTEL are equivalent to the corresponding centralized ones, where the whole training process, including gradient calculation and model updating, is performed at the server.

Lemma 1 (How many communication rounds?)**.** The distributed learning algorithms implemented at PARTEL are equivalent to the corresponding centralized ones in terms of convergence rate as measured by the required number of communication rounds (see e.g., [27], for convergence analysis on the latter). Specifically, for distributed learning, the values of gradients or proximal gradients calculated in each round and the number of rounds required for model convergence are independent on parameter allocation and bandwidth allocation.

Proof: See Appendix A.

The following proposition follows from Lemma 1.

Proposition 1 (Problem simplification)**.** The learning-latency-minimization problem in (P1) is equivalent to separately minimizing the latencies for all rounds:

$$\text{(P2)} \quad \min_{\{b_k^{(r)}\},\{\rho_{k,n}^{(r)}\}} t^{(r)}\left(\{b_k^{(r)}\},\{\rho_{k,n}^{(r)}\}\right) \quad \text{s.t. (C1.1) \& (C1.2).} \tag{14}$$

The simplified problem is solved in the following sections to obtain the optimal polices for PABA. For simplicity, the notation (r) is omitted.

IV. Bandwidth Aware Parameter Allocation

In this section, the scheme of bandwidth aware parameter allocation is designed based on the approach on the left of Fig. 1(a). Given bandwidth allocation, the optimal parameter allocation is proposed, which requires the latencies of all groups equal to the optimum. Besides, according to the optimal solution, the length of the parametric block assigned to group $\mathcal{G}_k$ is inversely proportional to its slowest worker's total latency for computing and uploading one gradient element.

First, the bandwidths are allocated to the workers independent of their assigned parametric block, e.g., equal bandwidth allocation. Next, given allocated bandwidths, the parameters are allocated by solving (P2), giving the algorithm of bandwidth aware parameter allocation. Specifically, given the bandwidth-allocation scheme $\{\rho_{k,n}^*\}$, the problem of one-round-latency minimization in (P2) can be simplified as

$$\text{(P3)} \quad \begin{aligned} \min_{\{b_k\}} \max_k \ & t_k\left(b_k\right), \\ \text{s.t. } & b_k \in \mathbb{Z}^+,\ 1 \le k \le K, \\ & \sum_{k=1}^{K} b_k = N_{\rm p}, \end{aligned} \tag{15}$$

where $t_k\left(b_k\right)$ and b_k are the latency and the parametric-block length of group $\mathcal{G}_k$, respectively. By substituting the push latency in (4), the computation latency in (5), and the push latency in (6) into the group latency in (8), we can obtain

$$t_k\left(b_k\right) = T_{\rm ph} + \max_{n\in\mathcal{G}_k}\left\{\frac{D_{k,n}O}{f_{k,n}^{\rm c}} + \frac{A_{\rm g}}{\rho_{k,n}^* BR_{{\rm u},k,n}}\right\} b_k + T_{\rm s}, \tag{16}$$

which shows that $t_k\left(b_k\right)$ is a linear function of b_k. Furthermore, by defining $t_{\rm PA} = \max_k t_k\left(b_k\right)$, (P3) can be converted into the following *mixed-integer linear problem* (MILP),

$$\begin{aligned} \min_{\{b_k\},t_{\rm PA}} \ & t_{\rm PA}, \\ \text{s.t. } & b_k \in \mathbb{Z}^+,\ 1 \le k \le K, \\ & \sum_{k=1}^{K} b_k = N_{\rm p}, \\ & t_k\left(\{b_k\}\right) \le t_{\rm PA},\ 1 \le k \le K. \end{aligned} \tag{17}$$

To solve (17), we follow the typical way, which first relaxes $\{b_k\}$ to be continuous and then round the solution. The error caused by the relaxation and rounding is just one parameter and is negligible due to the typically large values of $\{b_k\}$ (e.g., thousands to tens of thousands).

Theorem 1 (Relaxed parameter allocation)**.** By relaxing the integer constraints $\{b_k \in \mathbb{Z}^+,\ 1 \le k \le K\}$ to $\{b_k \ge 0,\ 1 \le k \le K\}$, the problem in (17) can be solved by linear programming. The solution requires all groups to have the same latency:

$$t_k\left(b_k\right) = t_{\rm PA}^*,\ \forall k, \tag{18}$$

where $t_{\rm PA}^*$ solves the following equation and can be computed using e.g., a bisection search:

$$\sum_{k=1}^{K} \frac{t_{\rm PA}^* - T_{\rm ph} - T_{\rm s}}{\max\limits_{n\in\mathcal{G}_k}\left\{\frac{D_{k,n}O}{f_{k,n}^{\rm c}} + \frac{A_{\rm g}}{\rho_{k,n}^* BR_{{\rm u},k,n}}\right\}} = N_{\rm p}. \tag{19}$$

The optimal parameter-allocation policy assigns b_k^* parameters to group $\mathcal{G}_k$ with b_k^* given by

$$b_k^* = \frac{t_{\rm PA}^* - T_{\rm ph} - T_{\rm s}}{\max\limits_{n\in\mathcal{G}_k}\left\{\frac{D_{k,n}O}{f_{k,n}^{\rm c}} + \frac{A_{\rm g}}{\rho_{k,n}^* BR_{{\rm u},k,n}}\right\}},\ 1 \le k \le K. \tag{20}$$

Proof: See Appendix B.

Two observations can be made from Theorem 1. First, according to (19), *the minimal per-round latency, $t_{\rm PA}^*$, linearly increases as the total number of parameters, $N_{\rm p}$, grows*. Second, in (20), the terms $\frac{D_{k,n}O}{f_{k,n}^{\rm c}}$ and $\frac{A_{\rm g}}{\rho_{k,n}^* BR_{{\rm u},k,n}}$ are the time for computing one gradient element and the time for uploading the element for worker (k,n), respectively. From (20), we can observe that *the optimal parametric-block length assigned to group $\mathcal{G}_k$, say b_k, is inversely proportional to its slowest worker's total latency for computing and uploading one gradient element*.

Rounding the real-valued numbers of parameters assigned to the groups gives the algorithm of bandwidth aware parameter allocation in Algorithm 1.

V. Computation Aware Bandwidth Allocation

In this section, the scheme of parameter aware bandwidth allocation is designed based on the approach on the right of Fig. 1(a). Given parameter allocation, the optimal bandwidth allocation is proposed, where all workers' latencies equal to the optimum and most bandwidth should be allocated to the worker with smallest communication rate and longest computation time.

Algorithm 1 Bandwidth Aware Parameter Allocation

1: **Input:** $\{\rho_{k,n}^*\}$, $\{R_{\mathrm{u},k,n}\}$.
- $\{\rho_{k,n}^*\}$, pre-determined bandwidth-allocation scheme,
- $\{R_{\mathrm{u},k,n}\}$, the uplink spectrum efficiencies.

2: Get the optimal latency t_{PA}^* of the relaxed problem by solving (19) with bisection method.
3: Determine the practical block size $\{\hat{b}_k^*\}$ as

$$\begin{cases} \hat{b}_k^* = \mathrm{round}\left(\dfrac{t_{\mathrm{PA}}^* - T_{\mathrm{ph}} - T_{\mathrm{s}}}{\max\limits_{n\in\mathcal{G}_k} \left\{ \dfrac{D_{k,n}O}{f_{k,n}^c} + \dfrac{A_{\mathrm{g}}}{\rho_{k,n}^* B R_{\mathrm{u},k,n}} \right\}} \right), \ k < K, \\ \hat{b}_K^* = N_{\mathrm{p}} - \sum\limits_{k=1}^{K-1} \hat{b}_k. \end{cases}$$

4: Calculate the near-optimal latency $\hat{t}_{\mathrm{LB}}^*$ with $\{\hat{b}_k^*\}$.
5: **Output:** $\{\hat{b}_k^*\}$ and $\hat{t}_{\mathrm{LB}}^*$.

First, the parametric-block lengths are assigned to the groups independent of their spectrum efficiencies, e.g., the parametric-block length assigned to one group is proportional to its computation latency of computing one gradient element. Next, given assigned parametric blocks, the bandwidths are allocated by solving (P2), giving the algorithm of parameter aware bandwidth allocation. Specifically, given the parameter-allocation policy $\{\hat{b}_k^*\}$, the problem of one-round-latency minimization in (P2) reduces to

$$\text{(P4)} \quad \begin{aligned} &\min_{\{\rho_{k,n}\}} \max_k t_k\left(\{\rho_{k,n}\}\right), \\ &\text{s.t. } \rho_{k,n} \geq 0, \ \forall (k,n), \\ &\sum_{k=1}^{K} \sum_{n\in\mathcal{G}_k} \rho_{k,n} \leq 1, \end{aligned} \tag{21}$$

where $t_k\left(\{\rho_{k,n}\}\right)$ is the latency of group $\mathcal{G}_k$ defined in (8) and $\rho_{k,n}$ is the uplink bandwidth ratio allocated to worker (k,n).

Lemma 2 (Convexity of bandwidth allocation)**.** (P4) is a convex problem.

Proof: See Appendix C.

By solving the convex problem, the minimal latency and the optimal bandwidth-allocation scheme can be obtained, as shown in the following theorem.

Theorem 2 (Parameter aware bandwidth allocation)**.** The optimal solution of the bandwidth-allocation problem in (P4) requires all workers have the same latency: $t_{k,n}\left(\rho_{k,n}\right) = t_{\mathrm{BA}}^*, \ \forall (k,n)$, where t_{BA}^* is the minimal latency that solves the following equation and can be computed using e.g., a bisection search:

$$\sum_{k=1}^{K} \sum_{n\in\mathcal{G}_k} \frac{\hat{b}_k^* A_{\mathrm{g}}}{\left(t_{\mathrm{BA}}^* - T_{\mathrm{s}} - T_{\mathrm{ph}} - \hat{T}_{k,n}\right) R_{\mathrm{u},k,n}} = B, \tag{22}$$

where $\{\hat{T}_{k,n} = \hat{t}_{k,n}(\hat{b}_k^*)\}$ are the computation latency and are constants. The resultant scheme of computation aware bandwidth allocation is given as

$$\rho_{k,n}^* = \frac{\hat{b}_k^* A_{\mathrm{g}}}{\left(t_{\mathrm{BA}}^* - T_{\mathrm{s}} - T_{\mathrm{ph}} - \hat{T}_{k,n}\right) R_{\mathrm{u},k,n} B}, \ \forall (k,n). \tag{23}$$

Proof: See Appendix D.

Two observations can be made from Theorem 2. First, in (22), the system bandwidth is a strict decreasing function of the optimum t_{BA}^*. In turn, it's easy to show that *the optimal latency t_{BA}^* strictly decreases as the system bandwidth B increases*. Second, for the optimal bandwidth-allocation scheme in (23), the allocated bandwidth of any one worker is a decreasing function of its uplink data rate and is an increasing function of its computation time. In other words, *the most bandwidth should be allocated to the worker with smallest uplink rate and longest computation time*.

The optimal scheme of parameter-aware bandwidth allocation is summarized in Algorithm 2.

Algorithm 2 Parameter Aware Bandwidth Allocation

1: **Input:** $\{\hat{b}_k^*\}$ and $\{R_{\mathrm{u},k,n}\}$.
- $\{\hat{b}_k^*\}$, the pre-determined parametric-block lengths,
- $\{R_{\mathrm{u},k,n}\}$, the uplink spectrum efficiencies.

2: Calculate the optimal latency t_{BA}^* by solving (22) with bisection method.
3: Determine the bandwidth $\{\rho_{k,n}^*\}$ as (23).
4: **Output:** $\{\rho_{k,n}^*\}$ and t_{BA}^*.

VI. Joint Parameter Allocation and Bandwidth Allocation

In this section, joint PABA based on the design approach in Fig. 1(b) is considered. Leveraging the results in preceding sections, the optimization problem (P2) is simplified. This allows an efficient solution method to be developed for computing the optimal policy for joint PABA. To gain further insights, two special cases are considered.

A. Optimal Joint PABA

The optimal joint PABA policy for PARTEL is computed and analzyed by solving Problem (P2) following a series of steps as follows.

1) Problem Simlification: First, we simplify (P2) by using the results in Theorem 2 and relaxing the parametric-block lengths $\{b_k\}$ to be continuous. According to Theorem 2, to achieve the minimal latency, all workers should have the same one-round latency [defined in (9)], namely $t_{k,n} = t, \ \forall (k,n)$. In Theorem 2, $\{b_k\}$ are given but they are variables in the current case. Thus, the bandwidth-allocation policy in (23) should be rewritten as a function of $\{b_k\}$:

$$\rho_{k,n}\left(b_k, t\right) = \frac{b_k A_{\mathrm{g}}}{\left[t - T_{\mathrm{s}} - T_{\mathrm{ph}} - \hat{t}_{k,n}\left(b_k\right)\right] R_{\mathrm{u},k,n} B}, \ \forall (k,n), \tag{24}$$

where $\hat{t}_{k,n}\left(b_k\right) = \dfrac{b_k D_{k,n} O}{f_{k,n}^c}$ is the computation latency following from (5), and T_{s} and T_{ph} are the server updating and push latency, respectively. Moreover, we relax the parametric-block lengths (in bits) $\{b_k\}$, to be continuous to simplify the solution of (P2), which can be rounded to yield the policy. As mentioned, in large-scale learning models, the values of $\{b_k\}$ are large and the performance loss caused by rounding is

negligible. By substituting $t_{k,n} = t$ and relaxing $\{b_k\}$, Problem (P2) is simplified as

$$\text{(P5)} \quad \begin{aligned} \min_{\{b_k\},t} \ & t \\ \text{s.t.} \ & b_k \geq 0, \ \forall k, && \text{(C5.1)} \\ & \sum_{k=1}^{K} b_k = N_{\mathrm{p}}, && \text{(C5.2)} \\ & \sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k, t\right) \leq 1, && \text{(C5.3)} \end{aligned}$$

where $\rho_{k,n}\left(b_k, t\right)$ is given in (24).

2) A Useful Property: It is easy to show that (P5) is non-convex. To transform the problem into a tractable convex problem, we derive a useful property. To this end, two definitions are introduced. One is *worker bandwidth allocation rate*, defined as the required bandwidth per additional parameter for one worker. Using (24), it can be written mathematically as

$$\frac{\partial \rho_{k,n}\left(b_k, t\right)}{\partial b_k} = \frac{A_{\mathrm{g}}[1 + \hat{t}_{k,n}\left(b_k\right)/\tilde{t}_{k,n}\left(b_k\right)]}{\tilde{t}_{k,n}\left(b_k\right) BR_{\mathrm{u},k,n}}, \ \forall (k,n). \tag{25}$$

One can recall that t denotes the one-round latency, $\hat{t}_{k,n}\left(b_k\right)$ and $\tilde{t}_{k,n}\left(b_k\right)$ are the computation latency and uploading time of worker (k,n), respectively. Note that in (25), the physical meaning of the term, $\frac{A_{\mathrm{g}}}{\tilde{t}_{k,n}\left(b_k\right) BR_{\mathrm{u},k,n}}$, is the required bandwidth for worker (k,n) to upload one gradient element; the scaling factor, say $\hat{t}_{k,n}\left(b_k\right)/\tilde{t}_{k,n}\left(b_k\right)$, accounts for computation latency. Next, the *group bandwidth allocation rate* is defined as the required bandwidth per additional parameter for one group, which sums the bandwidth allocation rates over the workers in the same group as follows:

$$\sum_{n \in \mathcal{G}_k} \frac{\partial \rho_{k,n}\left(b_k, t\right)}{\partial b_k} \triangleq \sum_{n \in \mathcal{G}_k} \frac{A_{\mathrm{g}}[1 + \hat{t}_{k,n}\left(b_k\right)/\tilde{t}_{k,n}\left(b_k\right)]}{\tilde{t}_{k,n}\left(b_k\right) BR_{\mathrm{u},k,n}}, \ \forall k. \tag{26}$$

From (25) and (26), two observations can be made. One is that worker/group bandwidth allocation rates depend on both their computation capacities and communication rates. The other is that given fixed one-round latency t, the rates increase as the length of the assigned parametric blocks ($\{b_k\}$) grows. The reason is that heavier load increases computation latency and thereby shortens allowed communication latency, making it necessary to have a larger bandwidth. Based on the above definitions, one key property of the optimal policy is derived as shown below.

Lemma 3 (Uniform Group Bandwidth Allocation Rates)**.** Given a constant C, a necessary and sufficient condition for the solution of (P5) is

$$\sum_{n \in \mathcal{G}_k} \frac{\partial \rho_{k,n}\left(b_k, t\right)}{\partial b_k} = C, \quad \forall k, \tag{27}$$

for a fixed model size, namely $\sum_{k=1}^{K} b_k = N_{\mathrm{p}}$, and under the constraint on the bandwidth allocation ratios: $\sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k, t\right) = 1$.

Proof: See appendix E.

3) Problem of Model Size Maximization: Though the property in Lemma 3 is insightful, the direct policy computation by solving the equations in (27) requires a $(K+2)$-dimensional search, which is impractical when K is large. A more efficient solution can be derived by relating Problem (P5) to the convex problem of model size maximization introduced as follows.

Given one-round latency t for an arbitrary round, let $N_{\mathrm{p}}^{*}\left(t\right)$ denote the maximum size of a model that can be updated within the round. Then $N_{\mathrm{p}}^{*}\left(t\right)$ solves the following problem of model size maximization

$$\text{(P6)} \quad \begin{aligned} N_{\mathrm{p}}^{*}(t) = \max_{\{b_k\}} \ & \sum_{k=1}^{K} b_k \\ \text{s.t.} \ & b_k \geq 0, \ \forall k, \\ & \sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k, t\right) \leq 1, \end{aligned}$$

where the notation follows that in Problem (P5). Note that given the one-round latency t, if and only if the solution in Problem (P6) satisfies $N_{\mathrm{p}}^{*}(t) \geq N_{\mathrm{p}}$, where N_{p} is the target updating model size, Problem (P5) is feasible, as all its constraints can be satisfied.

Two useful lemmas for relating Problems (P5) and (P6) are given as follows, which are proved in Appendices F and G.

Lemma 4 (Relation of maximal feasible model size and latency)**.** The maximal model size $N_{\mathrm{p}}^{*}(t)$ is a monotonously increasing function of the one-round latency t.

It follows from the result in Lemma 4 that the solution for Problem (P5) is the minimum latency t^{*}, for which the updatable model size $N_{\mathrm{p}}^{*}(t^{*})$ is no smaller than the target size N_{p}. This suggests a solution method of Problem (P5) by a search for t^{*} using the criterion $N_{\mathrm{p}}^{*}(t^{*}) \geq N_{\mathrm{p}}$ as elaborated in the next subsection.

This requires solving Problem (P6) so as to compute the function $N_{\mathrm{p}}^{*}(t^{*})$. To this end, the following result is useful.

Lemma 5. Given t, Problem (P6) is convex.

The convexity allows Problem (P6) to be solved using the traditional primal-dual method. Some needed notations are defined as follows. Let η_{λ} and $\{\eta_{b_k}\}$ denote the step sizes of gradient descent. The Lagrange function $\mathcal{L}_{\mathrm{P6}}$ is defined as

$$\mathcal{L}_{\mathrm{P6}} = -\sum_{k=1}^{K} b_k + \lambda \left[\sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k\right) - 1 \right], \ \text{with} \ \lambda \geq 0,$$

where $\{\rho_{k,n}\left(b_k\right), \forall (k,n)\}$ are defined in (45) and λ is the multiplier. Using the notations, the application of the primal-dual method yields Algorithm 3 for solving Problem (P6).

4) Solution by Nested Optimization: It follows from the results in the preceding subsection that Problem (P5) can be solved by nesting a one-dimensional search over t and the solution of a convex problem, namely Problem (P6). The search can be efficiently implemented using the bisection method given the monotonicity in Lemma 4 while the solution of Problem (P6) relies on Algorithm 3. Nesting them yields Algorithm 4 for computing the optimal joint PABA policy.

Algorithm 3 Model Size Maximization

Input: $\{R_{\mathrm{u},k,n}\}$ and T.
- $\{R_{\mathrm{u},k,n}\}$, the uplink spectrum efficiencies.
- T, the given one-round latency.

Initialize $t = T$, $\lambda^{(0)}$, and $l = 1$.
Loop
  $\lambda^{(l)} = \max\left\{\lambda^{(l-1)} + \eta_\lambda \frac{\partial \mathcal{L}_{\mathrm{P6}}}{\partial \lambda}, 0\right\}$.
  Initialize $\{b_k^{(0)}\}$ and $i = 1$.
  Loop
    $b_k^{(i)} = b_k^{(i-1)} - \eta_{b_k}\left[-1 + \lambda^{(l)} \sum\limits_{n\in\mathcal{G}_k} \frac{\partial \rho_{k,n}(b_k)}{\partial b_k}\right], \forall k.$
    $i = i + 1$.
  Until Convergence
  $\{b_k^* = b_k^{(i-1)}, \forall k\}$ and $l = l + 1$.
Until Convergence
Get $N_{\mathrm{p}}^*(T) = \sum_{k=1}^K b_k^*$.
Output: $N_{\mathrm{p}}^*(T)$ and $\{b_k^*\}$.

The computational complexity of Algorithm 4 can be divided into two parts. One is the inner loop for solving the convex Problem (P6). Its complexity is $\mathcal{O}(K^3)$, where K is the number of worker groups. The other is the outer loop of bisection search, with complexity of $\mathcal{O}[\log(1/\delta)]$ and δ being the convergence tolerance. The overall computational complexity of Algorithm 4 is $\mathcal{O}[\log(1/\delta)K^3]$. As K is small (e.g., $K = 15$) compared with the number of parameters (e.g., $N_{\mathrm{p}} > 10^6$), the computational overhead caused by Algorithm 4 at the server can be ignored.

Algorithm 4 Optimal Joint Parameter Allocation and Bandwidth Allocation

Input: $\{R_{\mathrm{u},k,n}\}$.
- $\{R_{\mathrm{u},k,n}\}$, the uplink spectrum efficiencies.

Select $t_{\mathrm{u}} = T_{\mathrm{u}}$ so that $t = t_{\mathrm{u}}$ makes $N_{\mathrm{p}}^*(t_{\mathrm{u}})$ defined in (P6) larger than N_{p}.
Select $t_{\mathrm{l}} = T_{\mathrm{l}}$ so that $t = t_{\mathrm{l}}$ makes $N_{\mathrm{p}}^*(t_{\mathrm{l}}) < N_{\mathrm{p}}$.
While $t_{\mathrm{u}} \neq t_{\mathrm{l}}$
  Let $t_{\mathrm{m}} = (t_{\mathrm{u}} + t_{\mathrm{l}})/2$. And substitute $t = t_{\mathrm{m}}$ in to (P6).
  Solve (P6) with Algorithm 3 to obtain $N_{\mathrm{p}}^*(t_{\mathrm{m}})$ and $\{b_k^*\}$ by inputing $\{R_{\mathrm{u},k,n}\}$ and t_{m}.
  If $N_{\mathrm{p}}^*(t_{\mathrm{m}}) \geq N_{\mathrm{p}}$
    $t_{\mathrm{u}} = t_{\mathrm{m}}$.
  Else
    $t_{\mathrm{l}} = t_{\mathrm{m}}$.
  End if
End while
$t^* = t_{\mathrm{m}}$.
Output: t^* and $\{b_k^*\}$.

B. Two Special Cases

1) Single-Worker Groups: Consider the case when each group comprises only a single worker. The dataset is not partitioned and each worker uses the whole dataset to update an assigned parametric block. Using the property of uniform group allocation rates in (27), the parametric-block length can be derived as functions of the latency t and the unknown variable C:

$$b_k(t, C) = \frac{f_{k,1}}{D_{k,1}O}\left(t - T_{\mathrm{ph}} - T_{\mathrm{s}} - \sqrt{\frac{A_{\mathrm{g}}t}{CR_{\mathrm{u},k,1}}}\right), \forall k. \quad (28)$$

From (28), the number of parameters assigned to group $\mathcal{G}_k$ decreases with its computation time to calculate one gradient element, say $\frac{f_{k,1}}{D_{k,1}O}$, and increases with its uploading rate $R_{k,1}^{\mathrm{u}}$. This is aligned with the analysis in (20). By substituting $b_k(t, C)$ in (28) into the constraints (C5.2) and (C5.3), it's easy to show that the following equations hold if the latency is at its minimum:

$$\begin{cases} \sum\limits_{k=1}^K b_k(t^*, C) = N_{\mathrm{p}}, \\ \sum\limits_{k=1}^K \sum\limits_{n\in\mathcal{G}_k} \rho_{k,n}(b_k(t^*, C)) = 1. \end{cases} \quad (29)$$

Using (29), the optimal latency t^* and the corresponding C can be efficiently solved for since there are only two variables. Then the optimal parameter allocation $\{b_k^*\}$ follows from (28). Substituting $\{\{b_k^*\}, t^*\}$ into (23) gives the optimal bandwidth allocation $\{\rho_{k,1}^*\}$.

2) Uniform Intra-group Computation Capacities: In this case, all workers within the same group, say group $\mathcal{G}_k$, have identical computation capacities and hence the same computation time: $\left\{\frac{D_{k,n}}{f_{k,n}^c} = \frac{D_{k,1}}{f_{k,1}}, \forall n \in \mathcal{G}_k\right\}$. In this case, a similar load-balancing scheme as the preceding special case can be derived:

$$b_k(t, C) = \frac{f_{k,1}}{D_{k,1}O}\left(t - T_{\mathrm{ph}} - T_{\mathrm{s}} - \sqrt{\frac{A_{\mathrm{g}}t}{C}\sum_{n\in\mathcal{G}_k}\frac{1}{R_{\mathrm{u},k,n}}}\right), \forall k.$$

A similar solution method can be applied to compute the optimal PABA policy, specified by the optimal load assignments $b_k(t^*, C)$ and bandwidth allocation $\{\rho_{k,n}^*\}$.

VII. Experimental Results

A. Experiment Setup

Consider a single-cell wireless network in a disk with a radius of 0.15 kilometres. The AP (edge server) is located at the center with multiple workers randomly located within the disk. The workers are separated in to K groups each having N workers. The total bandwidth is B. By default, their values are set as $K = 15$, $N = 15$, and $B = 100$ MHz unless specified otherwise. The workers' computation capacities are uniformly selected from the set $\{0.1, 0.2, \cdots, 1.0\}$ GHz. The learning task is a ℓ_1-regularized logistic regression task for training a news-filtering model using the News20 dataset collected in [28]. The model has $N_{\mathrm{p}} = 1,241,220$ parameters. The training dataset contains 15936 samples and the test dataset contains 3993 samples. For each group, the training dataset is uniformly partitioned into N subsets. Each subset is downloaded by one worker. Wireless channels are modelled with the following parameters. The noise power density is $N_0 = -174$ dBm/Hz. The transmission power of AP and workers is $P_{\mathrm{b}} = 46$ dBm and $P_{\mathrm{u}} = 24$ dBm, respectively. The path loss between worker

and AP is $128.1 + 37.6 \log d$ with the distance d in kilometre. Rayleigh fading is assumed.

For comparison, we consider four algorithms as follows [1].

- *Baseline*: The number of parameters assigned to one worker is proportional to its computation capability and bandwidths are equally allocated.
- *Bandwidth-aware parameter allocation*: The bandwidth is first equally allocated and then the parameters are allocated by the scheme of bandwidth aware parameter allocation in Algorithm 1.
- *Parameter-aware bandwidth allocation*: The parameters are proportionally allocated first and then the bandwidth is allocated using the scheme of parameter aware bandwidth allocation in Algorithm 2.
- *Joint PABA*: The parameter allocation and bandwidth allocation are jointly optimized using Algorithm 4.

B. Learning Performance

Latency minimization PABA can accelerate the model convergence. To evaluate the gain, the curves of (model) training and test accuracies versus latency are plotted in Fig. 3. As observed, the partially integrated and joint PABA algorithms outperform the baseline algorithm in terms of model convergence. For example, given the latency of 100 second, the proposed joint PABA, parameter aware bandwidth allocation, and bandwidth aware parameter allocation achieve a training accuracy of (5.24%, 3.49%, 2.26%) and a test accuracy of (4.40%, 2.84%, 1.99%) higher than that of the baseline algorithm, respectively. With respect to the baseline, the performance gains of the PABA schemes are due to the integration of workload and bandwidth. Specifically, additional bandwidth (or less workload) is allocated to a device to compensate for slow computation (or weak link), thereby reducing the per-round latency. The PABA schemes with varying complexity support different degrees of integration and as a result, achieve different levels of latency reduction with joint PABA performing the best.

C. PARTEL v.s. FEEL

In Fig. 4, the proposed PARTEL framework and the FEEL framework are compared. As mentioned in Remark 2, the PARTEL framework reduces to the FEEL framework in the case of only one group. Consider there are 50 workers in the cell. For PARTEL, they are clustered into 5 groups, each of which has 10 workers. For FEEL, they are in one group and hence no model partition. For fairness, we use the same scheme of joint PABA to compare PARTEL and FEEL. From the figure, the proposed PARTEL framework outperforms the FEEL framework with a latency reduction of 48.43% on average to achieve the same accuracy. The reason is as follows. For the PARTEL framework, each worker only needs to transmit the gradient of a parametric block while for the FEEL framework, the gradient of the whole parameter vector is needed to be uploaded by each worker. This makes the uploading latency much larger for the latter, especially when the wireless resources are limited.

[1]The source codes for implementing PARTEL are available at https://www.eee.hku.hk/%7ewirelesslab/resources/Demo.zip.

D. Latency Performance

The latency performance of joint and partially integrated PABA and baseline scheme in terms of expected one-round latency are compared in Fig. 5 for a varying bandwidth and a varying number for worker groups. First, as expected, the latency of all algorithms are observed to decrease as either the bandwidth or group number increase, representing more communication and computation resources, respectively. For a large bandwidth (or a group number), the latency saturates as it is dominated by computation latency (or communication latency). Next, the PABA algorithms are observed to significantly reduce the latency with respect to the baseline scheme. In particular, joint PABA achieves latency reduction of 46.73% for the bandwidth of 70 MHz and 46.92% for the number of groups equal to 18. Among the PABA algorithms, joint PABA outperforms two partially integrated PABA algorithms at the cost of higher complexity. On the other hand, the latency comparison between parameter aware bandwidth allocation and bandwidth aware parameter allocation suggests the former is more effective. Because the former can cope with channel heterogeneity for all workers while the latter can only cope with the computation capacity heterogeneity in group level.

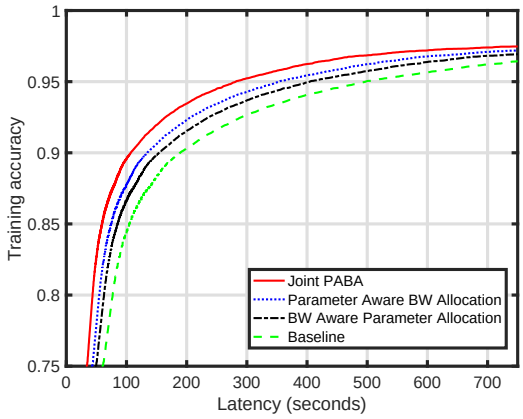
Consider the case where the number of worker groups is fixed but the group size grows. The growth has two conflicting effects. On one hand, the computation load, say the number of assigned parameters, per worker reduces, resulting in decreasing computation latency. On the other hand, more workers sharing a fixed bandwidth causes increases the communication latency. This suggests an optimal group size for learning latency minimization as confirmed by the curves of latency versus number of workers per group plotted in Fig. 6. The optimal group size differs for different algorithms e.g., 20 for joint PABA and 18 for the baseline scheme.

The simulation results above show that the proposed joint PABA scheme has the best performance and verify our analysis in Sections IV, V, and VI.

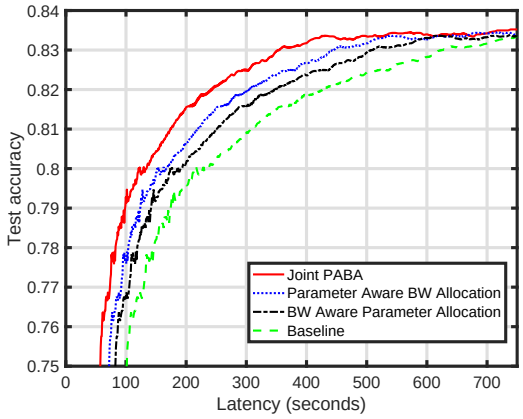
VIII. Conclusion

In this paper, we have proposed the new edge-learning framework, PARTEL, for performing a large-scale learning task in a wireless network. The framework features both data-and-model partitioning for distributing learning at many resource-constrained mobile devices. For efficient edge implementation of PARTEL, we have jointly designed the functional blocks of *parameter allocation and bandwidth allocation* (PABA), resulting in substantial latency reduction.

The current work opens several interesting directions for future investigation. One direction is to extend the joint PABA design to the cases of multi-cell systems with inter-cell interference management required and coexisting of networks/services (PARTEL and non-PARTEL users). Another direction is to investigate worker scheduling to balance distribute computation capacities and multi-access latency. Designing communication techniques for PARTEL such as multi-

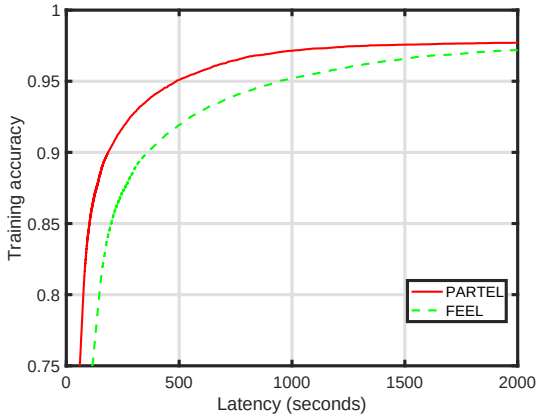


(a) Training accuracy versus latency.

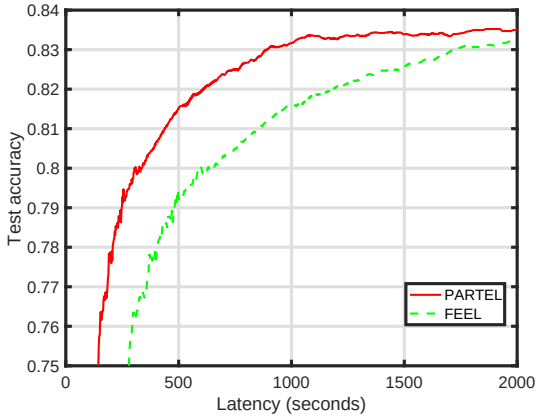


(b) Test accuracy versus latency.

Fig. 3. Learning performance versus (communication-plus-computation) latency.



(a) Training accuracy versus latency.



(b) Test accuracy versus latency.

Fig. 4. Comparison between PARTEL and FEEL.

antenna and millimeter-wave transmission is also an interesting direction to explore.

APPENDIX

A. Proof of Lemma 1

In the case of smooth regularization, both $\mathcal{F}(\boldsymbol{\theta})$ and $\mathcal{R}(\boldsymbol{\theta})$ are smooth functions. For an arbitrary round, say the $(r+1)$-th, gradient descent is used to update the model parameters:

$$\theta_{n_{\mathrm{p}}}^{(r+1)} = \theta_{n_{\mathrm{p}}}^{(r)} + \eta_r \nabla_{\theta_{n_{\mathrm{p}}}^{(r)}} \mathcal{L}\left(\boldsymbol{\theta}^{(r)}\right), 1 \leq n_{\mathrm{p}} \leq N_{\mathrm{p}}, \tag{30}$$

where $\theta_{n_{\mathrm{p}}}^{(r)}$ is the n_{p}-th element of $\boldsymbol{\theta}^{(r)}$, η_r is the learning rate, and $\nabla_{\theta_{n_{\mathrm{p}}}^{(r)}} \mathcal{L}\left(\boldsymbol{\theta}^{(r)}\right)$ is the gradient defined in (2). Besides, due to the decomposable structure of the objective function $\mathcal{L}(\cdot)$, $\nabla_{\theta_{n_{\mathrm{p}}}^{(r)}} \mathcal{L}\left(\boldsymbol{\theta}^{(r)}\right)$ is independent of $\nabla_{\theta_{n_{\mathrm{p}}}^{\prime(r)}} \mathcal{L}\left(\boldsymbol{\theta}^{(r)}\right)$, for any $n_{\mathrm{p}} \neq n_{\mathrm{p}}^{\prime}$. Hence, in the distributed algorithms, the ground-true gradients can be calculated for each parametric block. Thereby, the distributed gradient descent algorithm is equivalent to the centralized one.

In the case of non-smooth regularization, $\mathcal{R}(\boldsymbol{\theta})$ is non-smooth. For an arbitrary round, say the $(r+1)$-th, proximal gradient descent is used to update the learning parameters:

$$\boldsymbol{\theta}^{(r+1)} = \mathrm{prox}\left(\boldsymbol{\theta}^{(r)} - \eta_r \nabla \mathcal{F}(\boldsymbol{\theta}^{(r)})\right), \tag{31}$$

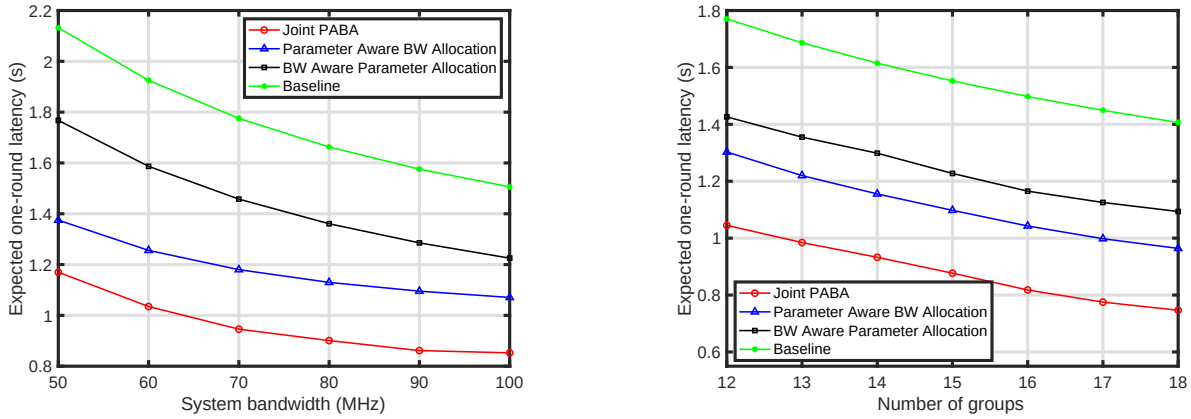
where $\mathrm{prox}(\mathbf{y}) = \arg\min_{\boldsymbol{\mu}} \left(\mathcal{R}(\boldsymbol{\mu}) + \frac{1}{2}||\boldsymbol{\mu} - \mathbf{y}||_2^2\right)$. Similarly, for any $n_{\mathrm{p}} \neq n_{\mathrm{p}}^{\prime}$, $\nabla_{\theta_{n_{\mathrm{p}}}} \mathcal{F}$ is independent of $\nabla_{\theta_{n_{\mathrm{p}}^{\prime}}} \mathcal{F}$. Besides, $\mathcal{R}(\boldsymbol{\mu})$ and $||\boldsymbol{\mu} - \mathbf{y}||_2^2$ are block separable. Thereby, the distributed proximal gradient descent algorithm is equivalent to the centralized one.

In summary, the distributed implementation has no impact on the calculated proximal gradients or gradients. Hence, the convergence rates (in rounds) only depend on the learning algorithms themselves and are irrelevant to parameter allocation and bandwidth allocation.

B. Proof of Theorem 1

After relaxation, the first constraint of (17) turns to be $\{b_k \geq 0\}$, while the other parts remain the same. The relaxed problem is a linear program. KKT conditions are used to solve the linear program. The Lagrange function can be written as

$$\mathcal{L} = t_{\mathrm{PA}} + \mu \left(\sum_{k=1}^{K} b_k - N_{\mathrm{p}}\right) + \sum_{k=1}^{K} \lambda_k \left[t_k\left(b_k\right) - t_{\mathrm{PA}}\right], \tag{32}$$



(a) Effect of Bandwidth (b) Effect of Number of Groups

Fig. 5. Latency performance comparison for (a) a varying bandwidth and (b) a varying number of worker groups.

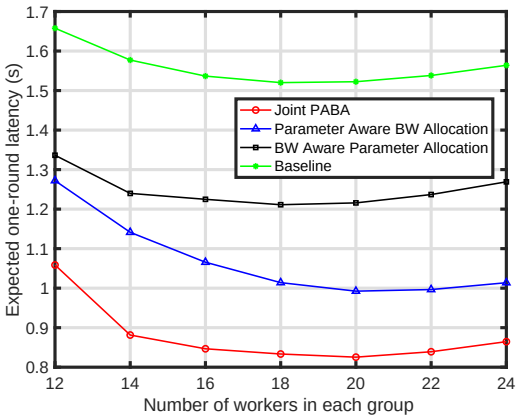


Fig. 6. One-round latency versus number of workers in each group

where μ and $\{\lambda_k \geq 0\}$ are the Lagrangian multipliers. Some useful KKT conditions are given by

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial b_k} = \mu + \lambda_k \max\limits_{n \in \mathcal{G}_k} \left\{ \dfrac{D_{k,n} O}{f^{\mathrm{c}}_{k,n}} + \dfrac{A_{\mathrm{g}}}{\rho^*_{k,n} B R_{\mathrm{u},k,n}} \right\} = 0, \ \forall k, \\ \dfrac{\partial \mathcal{L}}{\partial t_{\mathrm{PA}}} = 1 - \sum\limits_{k=1}^{K} \lambda_k = 0, \\ \lambda_k \left(t_k \left(b_k \right) - t_{\mathrm{PA}} \right) = 0, \ \forall k, \end{cases} \tag{33}$$

In (33), the second condition indicates $\exists k, \ \lambda_k \neq 0$. Then, together with the first condition, we can show $\mu \neq 0$, and further show that $\{\lambda_k \neq 0, \forall k\}$ as $\mu \neq 0$. Next, from the third condition, we have $t_k \left(b_k \right) - t_{\mathrm{PA}} = 0, \ 1 \leq k \leq K$. By substituting the group latency $t_k \left(b_k \right)$ in (16) into the above equation, (19) can be derived. Next, by substituting (19) into the constraint $\sum_{k=1}^{K} b_k = N_{\mathrm{p}}$, we can get (20). In (20), the optimum t^*_{PA} can be solved by bisection search because N_{p} increases with t_{PA}. Then, the optimal parameter-allocation scheme $\{b^*_k\}$ is given in (19).

C. Proof of Lemma 2

(P4) is convex if its objective function is convex, as the constraint is a linear set. First, for any one worker, say worker (k, n), by substituting the push latency in (4), the computation latency in (5), and the pull latency in (6), its total latency in (3) can be derived as

$$t_{k,n} \left(\rho_{k,n} \right) = T_{\mathrm{ph}} + \hat{T}_{k,n} + \frac{\hat{b}^*_k A_{\mathrm{g}}}{\rho_{k,n} B R_{\mathrm{u},k,n}} + T_{\mathrm{s}}, \tag{34}$$

where $\hat{T}_{k,n} = \hat{t}_{k,n}(\hat{b}^*_k)$ is a constant. In (34), $t_{k,n} \left(\rho_{k,n} \right)$ is a convex function of $\rho_{k,n}$. Then, the objective function of (P4)is also convex, as max operation preserves convexity.

D. Proof of Theorem 2

To solve (P4), we derive and solve an equivalent convex problem. First, define $t_{\mathrm{BA}} = \max_k t_k \left(\{\rho_{k,n}\} \right)$. Then, substituting it and $t_k \left(\{\rho_{k,n}\} \right) = \max_{n \in \mathcal{G}_k} t_{k,n} \left(\rho_{k,n} \right)$ into (P4), it can be equally derived as

$$\min_{\{\rho_{k,n}\}, t_{\mathrm{BA}}} t_{\mathrm{BA}}, \ \text{s.t.} \ \sum_{(k,n)} \rho_{k,n} \leq 1, \ \& \ t_{k,n} \left(\rho_{k,n} \right) \leq t_{\mathrm{BA}}, \ \forall (k,n).$$

KKT conditions are used to solve the convex problem above. The Lagrange function is

$$\mathcal{L} = t_{\mathrm{BA}} + \mu \left(\sum_{(k,n)} \rho_{k,n} - 1 \right) + \sum_{(k,n)} \lambda_{k,n} [t_{k,n} \left(\rho_{k,n} \right) - t_{\mathrm{BA}}],$$

where μ and $\{\lambda_{k,n} \geq 0\}$ are the multipliers, $t_{k,n} \left(\rho_{k,n} \right)$ is defined in (34). Then, the KKT conditions are

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \rho_{k,n}} = \mu - \lambda_{k,n} \dfrac{\hat{b}^*_k A_{\mathrm{g}}}{\rho^2_{k,n} B R_{\mathrm{u},k,n}} = 0, \ \forall (k,n), \\ \dfrac{\partial \mathcal{L}}{\partial t_{\mathrm{BA}}} = 1 - \sum\limits_{k=1}^{K} \sum\limits_{n \in \mathcal{G}_k} \lambda_{k,n} = 0, \\ \lambda_{k,n} \left[t_k \left(\{\rho_{k,n}\} \right) - t_{\mathrm{BA}} \right] = 0, \ \forall (k,n), \\ \sum\limits_{k=1}^{K} \sum\limits_{n \in \mathcal{G}_k} \rho_{k,n} - 1 = 0, \ t_k \left(\{\rho_{k,n}\} \right) \leq t_{\mathrm{BA}}, \ \forall (k,n). \end{cases} \tag{35}$$

In (35), the second condition shows that $\exists (k,n), \ \lambda_{k,n} \neq 0$. Then, together with the first condition, it can be derived that $\mu \neq 0$, and hence $\{\lambda_{k,n} \neq 0, \ \forall (k,n)\}$. In addition, according to the third condition in (35), we have $\{t_{k,n} \left(\rho_{k,n} \right) - t_{\mathrm{BA}} = 0, \ \forall (k,n)\}$. In the next, by substituting $t_{k,n} \left(\rho_{k,n} \right)$ in (34) into

the above equation, we can derive (22). Then, by substituting (22) into the condition $\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}-1=0$, (23) can be derived. In (23), bisection search can be used to find t_{BA}^*, as B strictly decreases with t_{BA}. Then, the optimal bandwidth-allocation scheme $\{\rho_{k,n}^*\}$ can be decided by (22).

E. Proof of Lemma 3

The KKT conditions of (P5) are used to show the sufficient and necessary conditions. First, the Lagrangian function is given by

$$\mathcal{L}=t+\mu\left(\sum_{k=1}^{K}b_k-N_{\mathrm{p}}\right)+\lambda\left[\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}\left(b_k,t\right)-1\right], \quad (36)$$

where $\rho_{k,n}\left(b_k,t\right)$ is defined in (24), and μ and $\lambda\geq 0$ are multipliers. Then, KKT conditions are necessary to achieve the optimum, which are given by

$$\begin{cases}\dfrac{\partial\mathcal{L}}{\partial t}=1+\lambda\sum\limits_{k=1}^{K}\sum\limits_{n\in\mathcal{G}_k}\dfrac{\partial\rho_{k,n}\left(b_k,t\right)}{\partial t}=0,\\ \dfrac{\partial\mathcal{L}}{\partial b_k}=\mu+\lambda\sum\limits_{n\in\mathcal{G}_k}\dfrac{\partial\rho_{k,n}\left(b_k,t\right)}{\partial b_k}=0,\ \forall k,\\ \lambda\left(\sum\limits_{(k,n)}\rho_{k,n}-1\right)=0,\sum\limits_{k=1}^{K}b_k=N_{\mathrm{p}},\ \sum\limits_{(k,n)}\rho_{k,n}\leq 1.\end{cases} \quad (37)$$

From the first condition in (37), we have $\lambda\neq 0$. Then, the second conditions can be derived as

$$\sum_{n\in\mathcal{G}_k}\frac{\partial\rho_{k,n}\left(b_k,t\right)}{\partial b_k}=C,\ \forall k, \quad (38)$$

where $C=-\mu/\lambda$. Next, as $\lambda\neq 0$, the third condition in (37) can be derived as

$$\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}\left(b_k,t\right)=1. \quad (39)$$

As a result, the above two conditions in (38) and (39) together with the forth condition in (37), which are summarized in (27), are necessary to achieve the minimal latency. Furthermore, it is easy to show that only one solution is in (27), which should be optimal. Hence, the conditions in (27) are sufficient and necessary to achieve the optimal latency.

F. Proof of Lemma 4

First, two useful facts are listed below.

- Fact1: The bandwidth allocation ratios $\{\rho_{k,n}\left(b_k,t\right)\}$ defined in (24) are monotonously decreasing function of the overall one-round latency t.
- Fact2: $\rho_{k,n}\left(b_k,t\right)$ is monotonously increasing function of the parametric-block length b_k for all workers.

Based on Fact2, to maximize the feasible model size, second condition in (P6) should be

$$\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}\left(b_k,t\right)=1. \quad (40)$$

Then, for $t=T_1$, denote the optimal solution of (P6) as $\{b_{k,1}^*\}$. The corresponding maximal feasible model size and optimal bandwidth allocation ratios are $n_{\mathrm{p}}^*(T_1)=\sum_{k=1}^{K}b_{k,1}^*$ and $\{\rho_{k,n}^*\left(b_{k,1}^*,T_1\right)\}$, respectively.

Next, assume any $T_2>T_1$, from Fact1, we have $\rho_{k,n}\left(b_{k,1}^*,T_2\right)<\rho_{k,n}^*\left(b_{k,1}^*,T_1\right)$ for all workers, which further shows that

$$\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}\left(b_{k,1}^*,T_2\right)<\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}^*\left(b_{k,1}^*,T_1\right)=1. \quad (41)$$

Furthermore, let

$$b_{k,2}=b_{k,1}^*,\ k=2,3,...K. \quad (42)$$

For group $\mathcal{G}_1$, we have

$$\begin{aligned}\sum_{n\in\mathcal{G}_1}\rho_{k,n}\left(b_{1,2},T_2\right)&=1-\sum_{k=2}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}\left(b_{k,2},T_2\right),\\ &>1-\sum_{k=2}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n}^*\left(b_{k,1}^*,T_1\right),\\ &=\sum_{n\in\mathcal{G}_1}\rho_{k,n}^*\left(b_{1,1}^*,T_1\right),\end{aligned} \quad (43)$$

where the two equalities are due to (40) and the inequality is due to (41) and (42). According to Fact2 and (43), we can show that $b_{1,2}>b_{1,1}^*$. That says $\sum_{k=1}^{K}b_{k,2}>\sum_{k=1}^{K}b_{k,1}^*=n_{\mathrm{p}}^*(T_1)$. Furthermore, the maximal feasible model size for $t=T_2$ satisfies $n_{\mathrm{p}}^*(T_2)\geq\sum_{k=1}^{K}b_{k,2}$. That says

$$n_{\mathrm{p}}^*(T_2)>n_{\mathrm{p}}^*(T_1). \quad (44)$$

G. Proof of Lemma 5

First, the objective of (P6) is convex and the the first constraint is a convex set. Then, given $t=T$, from (24), all bandwidth-allocation ratios in the second condition can be written as

$$\rho_{k,n}\left(b_k\right)=\frac{b_kA_{\mathrm{g}}}{\left[T-T_{\mathrm{s}}-T_{\mathrm{ph}}-\hat{t}_{k,n}\left(b_k\right)\right]R_{\mathrm{u},k,n}},\ \forall(k,n), \quad (45)$$

where $\hat{t}_{k,n}\left(b_k\right)=\dfrac{b_kD_{k,n}O}{f_{k,n}^c}$. In (45), $\{\rho_{k,n}\left(b_k\right),\ \forall(k,n)\}$ can be linearly transformed from the convex function $f(x)=\dfrac{x}{1-ax}$ with $ax<1$ and $a>0$. Hence, $\{\rho_{k,n}\left(b_k\right),\ \forall(k,n)\}$ are convex, as linear transformation preserves convexity. Thereby, the second condition of (P6) is a convex set.

REFERENCES

[1] D. Gesbert, D. Gündüz, P. de Kerret, C. R. Murthy, M. van der Schaar, and N. D. Sidiropoulos, "Guest editorial special issue on machine learning in wireless communication-Part I," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2181–2183, 2019.

[2] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Magazine*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[3] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Int. Conf. Comput. Comnun. (INFOCOM)*, Honululu, USA, April 2018.

[4] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Proc. NIPS Workshop on Big Learning*, Lake Tahoe, USA, Dec. 2013.

[5] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Broomfield, USA, Oct. 2014.

[6] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[7] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2014.

[8] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, Dec. 2013.

[9] M. Carreira-Perpinan and W. Wang, "Distributed optimization of deeply nested systems," in *Proc. Int. Workshop on Artificial Intelligence and Statistics (AISTATS)*, Reykjavik, Iceland, April 2014.

[10] Z. Zhang, Y. Chen, and V. Saligrama, "Efficient training of very deep neural networks for supervised hashing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, June 2016.

[11] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *[Online]. Available: https://arxiv.org/pdf/1909.11875.pdf*, 2019.

[12] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," *[Online]. Available: https://arxiv.org/abs/1604.00981.pdf*, 2016.

[13] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, Dublin, Ireland, Sep. 2018.

[14] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2018.

[15] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark, Dec. 2017.

[16] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Oct. 2019.

[17] M. M. Amiri and D. Gunduz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *[Online]. Available: https://arxiv.org/abs/1901.00844, 2019.*, 2019.

[18] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2020.

[19] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *[online]. Available: https://arxiv.org/pdf/1909.07972.pdf*, 2019.

[20] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317–333, Sep. 2019.

[21] Q. Zeng, Y. Du, K. K. Leung, and K. Huang, "Energy-efficient radio resource allocation for federated edge learning," *[online]. Available: https://arxiv.org/pdf/1907.06040.pdf*, 2019.

[22] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," *[online]. Available: https://arxiv.org/pdf/1911.00856.pdf*, 2019.

[23] J. Ren, G. Yu, and G. Ding, "Accelerating DNN training in wireless federated edge learning system," *[online]. Available: https://arxiv.org/pdf/1905.09712.pdf*, 2019.

[24] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *[online]. Available: https://arxiv.org/pdf/1911.02417.pdf*, 2019.

[25] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," *[online]. Available: https://arxiv.org/pdf/1909.02362*, 2019.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *[Online]. Available: https://arxiv.org/abs/1609.04747.pdf*, 2016.

[27] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[28] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 331–339.

Dingzhu Wen received the B.S.E. and M.S.E. degrees in information and communication engineering from Zhejiang University in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of EEE, The University of Hong Kong. His research interests include edge intelligence, distributed machine learning, MIMO communications, device-to-device communications, and full-duplex communications.

Dr. Mehdi Bennis is an Associate Professor at the Centre for Wireless Communications, University of Oulu, Finland, Academy of Finland Research Fellow and head of the intelligent connectivity and networks/systems group (ICON). His main research interests are in radio resource management, heterogeneous networks, game theory and distributed machine learning in 5G networks and beyond. He has published more than 200 research papers in international conferences, journals and book chapters. He has been the recipient of several prestigious awards including the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2016 Best Tutorial Prize from the IEEE Communications Society, the 2017 EURASIP Best paper Award for the Journal of Wireless Communications and Networks, the all-University of Oulu award for research and the 2019 IEEE ComSoc Radio Communications Committee Early Achievement Award. Dr Bennis is an editor of IEEE TCOM and Specialty Chief Editor for Data Science for Communications in the Frontiers in Communications and Networks journal.

Kaibin Huang (Senior Member, 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 2019 Best Tutorial Paper Award, 2015 Asia Pacific Best Paper Award, and 2019 Asia Pacific Outstanding Paper Award as well as Best Paper Awards at IEEE GLOBECOM 2006 and IEEE/CIC ICCC 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 PIMRC 2017 and IEEE CTW 2013. He is/was an Associate Editor for IEEE Transactions on Wireless Communications, IEEE Wireless Communications Letters, IEEE Transactions on Green Communications and Networking, and Journal on Selected Areas in Communications (JSAC). 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 Distinguished Lecturer and an ISI Highly Cited Researcher.