Accelerating Partitioned Edge Learning via Joint Parameter-and-Bandwidth Allocation

Dingzhu Wen
Univeristy of Hong Kong
Email: dzwen@eee.hku.hk

Mehdi Bennis
University of Oulu
Email: mehdi.bennis@oulu.fi

Kaibin Huang
Univeristy of Hong Kong
Email: huangkb@eee.hku.hk

***Abstract*—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 their local data. 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) 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 one scheme, namely *parameter aware bandwidth allocation*. It allocates the largest bandwidth to the slowest worker. Second, PABA are jointly optimized. Despite its 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%).**

I. INTRODUCTION

An enormous amount of data and computation resources are distributed over 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 aera [2].

A. Federated Edge Learning

A 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 [3]. Implementing stochastic gradient descent, 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. 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 [4], [5] 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 [6], multiuser scheduling [7], and their joint design [8]. Specifically, the schemes of data importance aware RRM are investigated in [9].

B. Partitioned Edge Learning

1) Partitioned Learning: A representative framework for partitioned distributed learning is called parameter server, which distributes a large-scale learning task over many resource-constrained machines by *model partitioning* [10]. Implementing the classic method of *block coordinate descent*, the framework iteratively and distributively solves a large-scale model-optimization problem with a decomposable objective function [e.g., SVM]. Specifically, a parameter sever 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 and updating the particular parametric block of the global model, completing the round. Recent research on partitioned learning focuses on CNN models. Such a model comprising nested layers does not have a decomposable loss function, making direct model partitioning sub-optimal. Overcoming the limitation has driven researchers to extend the BCD method to CNN [11]. Specifically, by introducing and conditioning on auxiliary variables, the layers in a CNN become conditional independent and thus can be trained separately. 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-sever framework with a decomposable function.

2) Edge Implementation: While communication channels are abstracted as bit piles in prior work, PARTEL concerns the design of new communication techniques for efficient implementation of partitioned learning in wireless networks. 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 and large-scale dataset. 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.

3) FEEL v.s. PARTEL: 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. Summary of Contributions

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 of improving the efficiency of PARTEL is to minimize the total 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 jointly design *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. Consider the optimization of the bandwidth allocation conditioned on giving parameter allocation, namely parameter aware bandwidth allocation. In this case, the minimal latency can be solved efficiently. The resultant optimal policy 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. It intelligently nests a bisection search and solving a *convex* problem.

II. Models and Metrics

A. System Model

A single-cell system is considered. 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 static in the learning duration. We assume that the AP has the channel state information 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 uplink channel gains of worker (k,n) is denoted as $H_{k,n}$, 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 can be made feasible for CNN models using the method of auxiliary variables [11]. Following the literature, a decomposable objective function has the following form: $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{F}(\boldsymbol{\theta}) + \mathcal{R}(\boldsymbol{\theta})$, 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 the trained learning model, respectively. Specifically, the loss function can be written as $\mathcal{F}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\varphi(\boldsymbol{\theta}; \mathbf{x}_m)$, where $\mathcal{X} = \{\mathbf{x}_m\}$ is the dataset, M is the size of the dataset, and $\varphi(\cdot)$ 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 the training, if the regularization function $\mathcal{R}(\cdot)$ is smooth, gradient descent can be used for updating the learning model. Otherwise, the learning model is updated by another method, called proximal gradient descent [12].

C. PARTEL Architecture

In the PARTEL architecture, 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 need 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.

In the PARTEL framework, one training iteration is called *one (communication) round*. In each round, the training includes three steps, described 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.

In each round, all updates uploaded from all workers should be synchronized. Hence, the latency in the round is decided by the "slowest" worker.

Remark 1 (Relation with FEEL)**.** In the case of only one worker group and hence no model partitioning, PARTEL reduces to FEEL (with uploading per round).

D. Latency Model

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

$$t_{k,n} = T_{\rm ph} + \hat{t}_{k,n} + \tilde{t}_{k,n}, \tag{1}$$

where $T_{\rm ph}$, $\hat{t}_{k,n}$, and $\tilde{t}_{k,n}$ correspond to the three steps, namely push, computation, and pull, respectively.

1) Push latency: The push latency ,denoted as $T_{\rm ph}$, is defined as the time for the server to broadcast the whole parameter vector $\boldsymbol{\theta}$ to all workers. Note that it 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 $f^{\rm c}_{k,n}$. Then, the computation latency of worker (k,n) is a function of its assigned load b_k, i.e., the length of its assigned parametric block $\boldsymbol{\theta}_k$. It can be written as

$$\hat{t}_{k,n}\left(b_k\right) = \frac{b_k D_{k,n} O}{f^{\rm c}_{k,n}}. \tag{2}$$

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. The other is the time for the server to update the learning model, which is ignored, as it is a constant for all workers and relatively small. Thereby, the pull latency is

$$\tilde{t}_{k,n}\left(b_k, \rho_{k,n}\right) = \frac{A_{\rm g} b_k}{\rho_{k,n} B R_{k,n}}, \tag{3}$$

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

We define the group latency per 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:

$$t_k\left(b_k, \{\rho_{k,n}\}\right) = \max_{n \in \mathcal{G}_k} \; t_{k,n}\left(b_k, \rho_{k,n}\right), \tag{4}$$

where $t_{k,n}\left(b_k, \rho_{k,n}\right)$ is the latency of worker (k,n) in (1).

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

$$t\left(\{b_k\}, \{\rho_{k,n}\}\right) = \max_{k} \; t_k\left(b_k, \{\rho_{k,n}\}\right). \tag{5}$$

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.The learning latency depends on two factors. One is the *number of communication rounds required for model convergence* and the other is the *per-round latency*. According to [13], the overall learning latency minimization is equivalent to separately minimizing the per-round latency, as the number of rounds till model convergence is irrelevant to the parameter and bandwidth allocation. In the sequel, the per-round latency minimization problem is formulated.

Consider an arbitrary round. We aim at minimizing the its latency, say $t\left(\{b_k\}, \{\rho_{k,n}\}\right)$ defined in (5), 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:

$$\text{(C1.1)} \begin{cases} \sum\limits_{k=1}^{K} b_k = N_{\rm p}, \\ b_k \in \mathbb{Z}^+, \quad 1 \le k \le K, \end{cases} \tag{6}$$

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

$$\text{(C1.2)} \begin{cases} \sum\limits_{k=1}^{K} \sum\limits_{n \in \mathcal{G}_k} \rho_{k,n} \le 1, \\ \rho_{k,n} \ge 0, \;\; \forall (k,n), \end{cases} \tag{7}$$

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

$$\text{(P1)} \; \min_{\{b_k\}, \{\rho_{k,n}\}} \; t\left(\{b_k\}, \{\rho_{k,n}\}\right), \;\; \text{s.t. (C1.1) \& (C1.2).}$$

IV. Computation Aware Bandwidth Allocation

In this section, the scheme of parameter aware bandwidth allocation is designed. 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.

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 (P1), 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 (P1) reduces to

$$\text{(P2)} \min_{\{\rho_{k,n}\}} \max_k \; t_k\left(\{\rho_{k,n}\}\right), \;\; \text{s.t. (C1.2)}, \tag{8}$$

where $t_k\left(\{\rho_{k,n}\}\right)$ is the latency of group $\mathcal{G}_k$ defined in (4).

Lemma 1. (P2) is a convex problem.

Proof: See Appendix A.

By solving the convex problem, the optimal solution can be obtained, as shown in the following theorem.

Theorem 1 (Parameter aware bandwidth allocation)**.** The optimal solution of problem (P2) 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:

$$\sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \frac{\hat{b}_k^* A_{\mathrm{g}}}{\left(t_{\mathrm{BA}}^* - T_{\mathrm{ph}} - \hat{T}_{k,n}\right) R_{k,n}} = B, \tag{9}$$

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

$$\rho_{k,n}^* = \frac{\hat{b}_k^* A_{\mathrm{g}}}{\left(t_{\mathrm{BA}}^* - T_{\mathrm{ph}} - \hat{T}_{k,n}\right) R_{k,n} B}, \; \forall (k,n). \tag{10}$$

Proof: See Appendix B.

Two observations can be made from Theorem 1. First, in (9), 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 (10), 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*.

V. Joint Parameter Allocation and Bandwidth Allocation

In this section, joint PABA is considered. Leveraging the results in preceding sections, the optimization problem (P1) is simplified. This allows an efficient solution method to be developed for computing the optimal policy for joint PABA.

The optimal joint PABA policy for PARTEL is computed and analzyed by solving Problem (P1) following a series of steps as follows.

1) Problem Simlification: First, we simplify (P1) by using the results in Theorem 1 and relaxing the parametric-block lengths $\{b_k\}$ to be continuous. According to Theorem 1, to achieve the minimal latency, all workers should have the same one-round latency [defined in (5)], namely $t_{k,n} = t, \; \forall (k,n)$. In the theorem, $\{b_k\}$ are given but they are variables in the current case. Thus, the bandwidth-allocation policy in (10) should be rewritten as a function of $\{b_k\}$:

$$\rho_{k,n}\left(b_k, t\right) = \frac{b_k A_{\mathrm{g}}}{\left[t - T_{\mathrm{ph}} - \hat{t}_{k,n}\left(b_k\right)\right] R_{k,n} B}, \; \forall (k,n), \tag{11}$$

where $\hat{t}_{k,n}\left(b_k\right) = \dfrac{b_k D_{k,n} O}{f_{k,n}^{\mathrm{c}}}$ is the computation latency following from (2), and T_{ph} is push latency. Moreover, we relax the parametric-block lengths (in bits) $\{b_k\}$, to be continuous to simplify the solution of (P1), 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 (P1) is simplified as

$$\text{(P3)} \quad \min_{\{b_k\}, t} \; t \quad \text{s.t. (C1.1)} \; \& \sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k, t\right) \leq 1,$$

where $\rho_{k,n}\left(b_k, t\right)$ is given in (11).

2) Problem of Model Size Maximization: An efficient solution can be derived by relating Problem (P3) 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

$$\text{(P4)} \quad \begin{aligned} N_{\mathrm{p}}^*(t) = \max_{\{b_k\}} \; & \sum_{k=1}^{K} b_k \\ \text{s.t.} \; & b_k \geq 0, \; \forall k, \\ & \sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}\left(b_k, t\right) \leq 1, \end{aligned}$$

where the notation follows that in Problem (P3).

Two useful Lemmas for relating Problems (P3) and (P4) are given as follows, whose proofs can be referred to Appendices F and G in [13].

Lemma 2 (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 2 that the solution for (P3) 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 suggest a solution method of Problem (P3) 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 (P4) so as to compute the function $N_{\mathrm{p}}^*(t^*)$. To this end, the following result is useful.

Lemma 3. Given t, Problem (P4) is convex.

The convexity allows Problem (P4) 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 $\mathcal{L}_{P4}$ is defined as

$$\mathcal{L}_{P4} = -\sum_{k=1}^{K} b_k + \lambda \left[\sum_{k=1}^{K} \sum_{n \in \mathcal{G}_k} \rho_{k,n}(b_k) - 1 \right], \text{ with } \lambda \geq 0, \quad (12)$$

where $\{\rho_{k,n}(b_k), \forall (k,n)\}$ are defined in (11) and λ is the multiplier. Using the notations, the application of the primal-dual method yields Algorithm 1 for solving Problem (P4).

Algorithm 1 Model Size Maximization

1: **Input:** uplink spectrum efficiencies $\{R_{k,n}\}$ and latency T.
2: **Initialize** $t = T$, $\lambda^{(0)}$, and $l = 0$.
3: **Loop**
4: $\quad \lambda^{(l+1)} = \max\left\{\lambda^{(l)} + \eta_\lambda \frac{\partial \mathcal{L}_{P4}}{\partial \lambda}, 0\right\}$.
5: $\quad$ **Initialize** $\{b_k^{(0)}\}$ and $i = 0$.
6: $\quad$ **Loop**
7: $\quad\quad b_k^{(i+1)} = b_k^{(i)} - \eta_{b_k}\left(-1 + \lambda^{(l+1)} \sum_{n \in \mathcal{G}_k} \frac{\partial \rho_{k,n}(b_k)}{\partial b_k}\right), \forall k$.
8: $\quad\quad i = i + 1$.
9: $\quad$ **Until Convergence**
10: $\quad \{b_k^* = b_k^{(i-1)}, \forall k\}$ and $l = l + 1$.
11: **Until Convergence**
12: Get $N_{\mathrm{p}}^*(T) = \sum_{k=1}^{K} b_k^*$.
13: **Output:** $N_{\mathrm{p}}^*(T)$ and $\{b_k^*\}$.

3) Solution by Nested Optimization: It follows from the results in the preceding subsection that Problem (P3) can be solved by nesting a one-dimensional search over t and the solution of a convex problem, namely Problem (P4). The search can be efficiently implemented using the bisection method given the monotonicity in Lemma 2 while the solution of Problem (P4) relies on Algorithm 1. Nesting them yields the algorithm for computing the optimal joint PABA policy, which is omitted due to the page limitation.

VI. Experimental Results

A. Experiment Setup

Consider a single-cell wireless network in a disk with a radius of 0.15 kilometres. The AP (edge sever) 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. 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 [14]. 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.

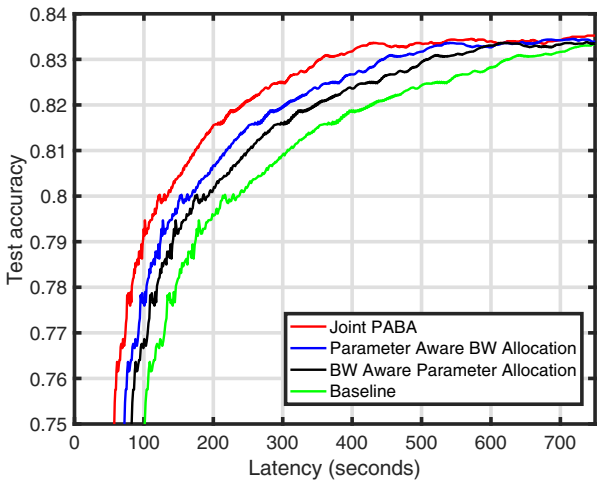

Figure 1. Learning performance versus latency.

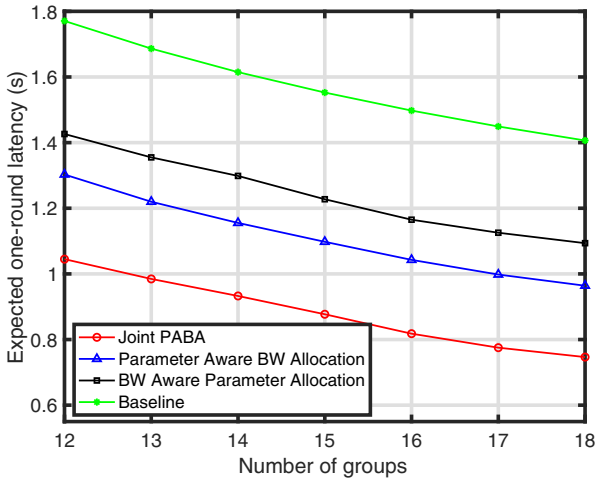

Figure 2. Latency comparison for a varying number of worker groups

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. Three algorithms are used for comparison. One is the baseline, where the number of parameters assigned to one worker is proportional to its computation capability and bandwidths are allocated equally. Other two algorithms are parameter aware bandwidth allocation in Section IV and bandwidth aware parameter allocation in [13].

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. 1. 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 test accuracy of (4.40%, 2.84%, 1.99%) higher than that of the baseline algorithm, respectively.

C. 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. 2 for a varying number for worker groups. First, as expected, the latency of all algorithms are observed to decreases as group number increase, representing more communication and computation resources, respectively. For a large 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 46.92% for the number of groups equal to 18.

The simulation results show that the proposed joint PABA scheme has the best performance and verify our analysis.

VII. Conclusion

In this paper, we have proposed a 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 case of fast fading channels using stochastic optimization as a tool. Another direction is to investigate worker scheduling to balance distribute computation capacities and multi-access latency. Designing communication techniques for PARTEL such as multi-antenna and millimeter-wave transmission is also an interesting direction to explore.

Appendix

A. Proof of Lemma 1

(P2) is convex if its objective function is convex, as the constraint is a linear set. First, consider worker (k,n), by substituting the computation latency in (2) and the pull latency in (3), its total latency in (1) can be derived as

$$t_{k,n}\left(\rho_{k,n}\right) = T_{\rm ph} + \hat{T}_{k,n} + \frac{\hat{b}_k^* A_{\rm g}}{\rho_{k,n} B R_{k,n}}, \tag{13}$$

where $T_{\rm ph}$ and $\hat{T}_{k,n} = \hat{t}_{k,n}(\hat{b}_k^*)$ are constants. In (13), $t_{k,n}\left(\rho_{k,n}\right)$ is a convex function of $\rho_{k,n}$. Then, the objective function of (P2)is also convex, as max operation preserves convexity.

B. Proof of Theorem 1

First, define $t_{\rm BA} = \max_k t_k\left(\{\rho_{k,n}\}\right)$. 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 (P2), we obtain

$$\min\ t_{\rm BA},\ \text{s.t.} \sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n} \le 1,\ \&\ t_{k,n}\left(\rho_{k,n}\right) \le t_{\rm BA},\ \forall (k,n). \tag{14}$$

The Lagrange function of the problem in (14) is

$$\mathcal{L} = t_{\rm BA} + \mu\left(\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n} - 1\right) + \sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\lambda_{k,n}\left[t_{k,n}\left(\rho_{k,n}\right) - t_{\rm BA}\right],$$

where μ and $\{\lambda_{k,n}\}$ are the multipliers, $t_{k,n}\left(\rho_{k,n}\right)$ in (13). Then, some of the KKT conditions are

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \rho_{k,n}} = \mu - \lambda_{k,n}\dfrac{\hat{b}_k^* A_{\rm g}}{\rho_{k,n}^2 B R_{k,n}} = 0,\ \forall (k,n), \\ \dfrac{\partial \mathcal{L}}{\partial t_{\rm BA}} = 1 - \displaystyle\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\lambda_{k,n} = 0, \\ \lambda_{k,n}\left(t_k\left(\{\rho_{k,n}\}\right) - t_{\rm BA}\right) = 0,\ \forall (k,n), \end{cases} \tag{15}$$

In (15), the second condition shows that $\exists (k,n),\ \lambda_{k,n} \neq 0$. Then, together with the first condition, we have $\mu \neq 0$, and hence $\{\lambda_{k,n} \neq 0,\ \forall (k,n)\}$. In addition, according to the third condition in (15), we have $\{t_{k,n}\left(\rho_{k,n}\right) = t_{\rm BA},\ \forall (k,n)\}$. Next, by substituting $t_{k,n}\left(\rho_{k,n}\right)$ in (13) into the above equation, we can derive (9). Finally, by substituting (9) into the condition $\sum_{k=1}^{K}\sum_{n\in\mathcal{G}_k}\rho_{k,n} = 1$, (10) can be derived. In (10), bisection search can be used to find $t_{\rm BA}^*$, as B strictly decreases with $t_{\rm BA}$. Then, the optimal bandwidth-allocation scheme $\{\rho_{k,n}^*\}$ is found using (9).

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] 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.

[4] 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.

[5] D. Wen, G. Zhu, and K. Huang, "Reduced-dimension design of MIMO over-the-air computing for data aggregation in clustered IoT networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5255–5268, 2019.

[6] M. M. Wadu, S. Samarakoon, and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," *[Online]. Available: https://arxiv.org/abs/2002.00802.pdf*, 2020.

[7] 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.

[8] 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.

[9] D. Wen, X. Li, Q. Zeng, J. Ren, and K. Huang, "An overview of data-importance aware radio resource management for edge machine learning," *Journal of Communications and Information Networks*, vol. 4, no. 4, pp. 1–14, 2019.

[10] 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.

[11] 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.

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

[13] D. Wen, M. Bennis, and K. Huang, "Joint parameter-and-bandwidth allocation for improving the efficiency of partitioned edge learning," *to appear in IEEE Trans. Wireless Commun.*, 2020.

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