Optimal Number of Edge Devices in Distributed Learning Over Wireless Channels

Jaeyoung Song and Marios Kountouris
Communication Systems Department, EURECOM
Sophia-Antipolis, France
Email: {jaeyoung.song, marios.kountouris}@eurecom.fr

***Abstract*—We consider a distributed learning system, where a parameter server (PS) assigns data and computational tasks to edge devices to build a global model. Distributing data to multiple workers involves communication between PS and edge devices and entails a fundamental tradeoff between computation and communication. In this paper, we aim at characterizing the optimal number of edge devices required for guaranteeing convergence and for achieving a certain accuracy within a finite time horizon.**

***Index Terms*—Distributed machine learning, mobile edge computing, parameter server.**

I. Introduction

The continuous generation of abundant and ubiquitous data renders inefficient or infeasible for a single entity to handle huge data sets for computation, training, and inference. Using multiple devices to share the processing load of the whole data set is an alternative that has recently attracted significant attention. Leveraging parallel computing, the response/completion time of distributed learning systems can be significantly reduced compared with centralized architectures. At the same time, users' demands for enhanced quality of experience and ultra low latency necessitate data-driven decision making closer to the end users, i.e., at the network edge [1]. In this context, distributed edge learning has become promising for large-scale learning applications.

In a distributed edge learning system, a parameter server (PS) splits the entire data into separate sets and distributes them to the edge devices. Since each edge device holds only part of the data set, the server needs to aggregate the outcomes of the edge devices' computations. Hence, intermediate results are sent from the edge devices to the server and coordination information is delivered to the edge devices from the server. In this context, communication between parameter server and edge devices becomes a key factor - and often the bottleneck - that determines the performance of distributed learning. Consequently, the communication overhead should be taken into account when optimizing distributed edge learning systems.

There are various approaches to reduce the communication overhead. Apart from the PS architecture, in which the server aggregates the local models from all edge devices and produces a global model, another promising approach is federated learning (FL) [2]. In FL, edge devices build local models based on their local data set and transmit only the local models, reducing the communication overhead and preserving user data privacy. Several approaches have been proposed for efficient use of communication and computation resources [3], [4]. Finding the best clustering under constraints of computation rate-coverage tradeoff in distributed learning is studied in [5]; the problem is shown to be NP-complete. The authors in [6] have investigated a tradeoff between local updates and global aggregations. Furthermore, a distributed learning framework that exploits duality principle in optimization has been proposed in [7]. The proposed framework is proven to converge to optimal solution by solving local subproblems and aggregating local solutions. This framework has been employed to evaluate the performance of different scheduling policies for local users [8].

Previous work on communication-efficient distributed learning assumes a fixed number of edge devices geographically distributed according to a given topology. However, the number of edge devices is a key system parameter, which heavily influences both computing performance and wireless communication features, such as multiple access and resource allocation. On the one hand, adding edge devices is always beneficial from a computation point of view, but on the other hand a higher number of edge devices is not always beneficial in terms of communication efficiency. Since additional edge device will share data and computational load, the workload of existing edge devices can be reduced. In other words, increasing the number of edge devices is equivalent to acquiring additional computational resources. However, as the number of edge devices increases, wireless resources per edge device decrease, which in turn affects the wireless communication performance, such as the transmission rate.

In many machine learning applications, completion time, defined here as the time required for achieving a certain convergence gap, is an important and practically relevant performance metric. Expectedly, completion time depends on the number of edge devices, which in turn dictates the time spent in computation and communication. Our objective is to to figure out the optimal number of edge devices that minimizes the completion time. For that, we cast the problem of completion time minimization in wireless distributed edge learning and investigate the tradeoff between computation and communication in terms of number of edge devices.

II. System Model

We consider a distributed edge learning system, as shown in Fig. 1, with a single PS and a set of K edge devices denoted by $\mathcal{K} = \{1, \cdots K\}$.

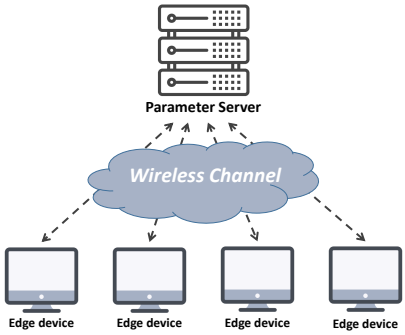

Fig. 1. A distributed edge learning system over a wireless channel.

A. Distributed Learning Framework

Suppose we have a set of N data examples represented by a matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$, where $\mathbf{x}_n \in \mathbb{R}^M$ is the n-th data example characterized by M features. The objective function $F(\mathbf{w})$ given a parameter $\mathbf{w} \in \mathbb{R}^M$ is defined as

$$F(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} l_n \left(\mathbf{x}_n^\top \mathbf{w}\right) + \lambda r(\mathbf{w}), \tag{1}$$

where $l_n(\cdot)$ is loss function, $r(\cdot)$ is regularizer such as ℓ_1 or ℓ_2-norm, and λ is the weight. For simplicity, we assume that $l_n(\cdot)$ is 1-smooth function and $r(\mathbf{w})$ is 1-strongly convex function. Note that our result can be readily extended to arbitrary smooth and convex functions. The goal of the server is to find the optimal (global) parameter that minimizes $F(\mathbf{w})$.

$$\min_{\mathbf{w} \in \mathbb{R}^d} F(\mathbf{w}). \tag{2}$$

However, it is not easy to handle very large data sets and distributed edge computing should be employed instead, i.e., the server can build the global model by coordinating multiple edge devices. First, the server splits the data set into K subsets which correspond to each edge device. Let us denote a set of indexes of data examples allocated to edge device k as $C_k \subseteq \{1, \cdots, N\}$. Thus, $\{C_k\}_{k=1}^K$ forms a partition of entire data set. We assume that duplicate allocation of a data example is not allowed and all examples should be given to at least one edge device. (i.e., $C_k \cap C_{k'} = \emptyset$ and $\cup_{k=1}^K C_k = \{1, \cdots, N\}$). The number of data examples assigned to edge device k is represented by n_k, with $\sum_{k=1}^K n_k = N$.

Given a data partition, the server utilizes distributed learning framework to efficiently solve (2) with a large set of data examples. There exist several distributed learning frameworks [4], [6], [7]. We employ the communication-efficient distributed dual coordinate ascent (CoCoA) framework, which guarantees convergence and shows wide applicability [7]. The CoCoA framework is shown in Algorithm 1. There are two key variables: a global parameter and local updates. The global parameter denoted as $\boldsymbol{\alpha}$ is directly related with global parameter $\mathbf{w}$ via dual relationship. The global parameter is updated based on a set of local updates which are solutions of local subproblems given to each edge device. Hence, once data is distributed, edge devices find local updates by solving local subproblems defined on the data set each of them receives and the current global parameter. Then, edge devices transmit local updates to the server. Next, the server produces the global parameter of next round by aggregating local updates. Hence, local computation of updates and transmission of local updates and the global parameter are repeated until convergence. We define a global iteration as a round of local computation and transmission of edge devices and the server. The local subproblem that edge device k is supposed to solve is written as:

Algorithm 1 CoCoA-Distributed Learning Algorithm

Input: Initial point $\boldsymbol{\alpha}^0$, aggregation parameter γ, partition of the entire data set $\{C_k\}_{k=1}^K$
for $t = 0, 1, 2, \cdots,$ **do**
  for $k \in \{1, 2, \cdots, K\}$ each edge device k does in parallel **do**
    for $\tau = 1, 2, \cdots, \tau_{\epsilon_l} - 1$ **do**
      $\Delta\boldsymbol{\alpha}_{[k]}^{\tau+1} \leftarrow \Delta\boldsymbol{\alpha}_{[k]}^{\tau} - \theta\nabla\left(\Delta D_k\left(\Delta\boldsymbol{\alpha}_{[k]}^{\tau}; \mathbf{X}\boldsymbol{\alpha}^t, \alpha_{[k]}^t\right)\right)$
    end for
    Transmit $\Delta\boldsymbol{\alpha}_{[k]}^{\tau_{\epsilon_l}}$ to the server
  end for
  Produce global parameter: $\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t + \gamma \sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]}^{\tau_{\epsilon_l}}$
  Multicast $\mathbf{X}\boldsymbol{\alpha}^{t+1}$ to all edge devices.
end for

$$\min_{\Delta\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^N} \Delta D_k\left(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{X}\boldsymbol{\alpha}^t\right) \tag{3}$$

where

$$\Delta D_k(\Delta\boldsymbol{\alpha}_{[k]}^t; \mathbf{X}\boldsymbol{\alpha}) = \frac{\lambda}{K} r^*\left(\frac{1}{\lambda N}\mathbf{X}\boldsymbol{\alpha}^t\right) + \frac{1}{N}\mathbf{w}^\top\left(\mathbf{X}\Delta\boldsymbol{\alpha}_{[k]}\right)$$
$$+\frac{\gamma K}{2\lambda N^2}\left\|\mathbf{X}\Delta\boldsymbol{\alpha}_{[k]}\right\|^2 + \frac{1}{N}\sum_{n \in C_k} l_n^*\left(-\alpha_n^t - \left(\Delta\boldsymbol{\alpha}_{[k]}\right)_n\right), \tag{4}$$

and $\gamma^*(\cdot)$ and $l_n^*(\cdot)$ are convex-conjugate functions of $\gamma(\cdot)$ and $l_n(\cdot)$, respectively. Also, γ is an aggregation weight, $\Delta\boldsymbol{\alpha}_{[k]}$ is the local update of edge device k, and $\boldsymbol{\alpha}^t$ is the global parameter at the t-th global iteration.

Since CoCoA is based on duality, its convergence is determined by duality gap defined as $G(\boldsymbol{\alpha}^t) = F(\mathbf{w}(\boldsymbol{\alpha}^t)) - D(\boldsymbol{\alpha})$, where $D(\boldsymbol{\alpha})$ is a dual function of $F(\mathbf{w})$. Note that it can be proven that optimality gap is smaller than duality gap. Thus, a solution satisfying duality gap is always within optimality gap from the optimal solution. If we repeat the whole procedure of local computation and exchange of the local updates and global parameters, after a finite number of iterations, we can

obtain the following duality gap [7]: for $n_k = \frac{N}{K}$, if $t \geq T_G$, then $G(\alpha^t) \leq \epsilon_G$, where

$$T_G = \frac{1}{\gamma(1-\epsilon_l)} \frac{\lambda+\gamma}{\lambda} \ln\left(\frac{(\lambda+\gamma)}{\gamma\lambda(1-\epsilon_l)} \cdot \frac{1}{\epsilon_G}\right). \tag{5}$$

The above result states that we can find a sequence of solutions produced by Algorithm 1 which reduces dual function exponentially. As a result, the number of iterations required to satisfy a given duality gap is a logarithmic function of the duality gap.

B. Communication Model

To achieve convergence, the PS and the edge devices need to exchange the local updates and global parameters; in our work the communication is performed over a wireless channel. Due to the broadcasting property of the wireless medium, there exists interference when multiple edge devices transmit simultaneously. To avoid interference, we assume that bandwidth is orthogonally allocated to each edge device k. Let us denote β_k the bandwidth allocated to edge device k. The transmission rate of edge device k to the server during the i-th global iteration can be expressed as

$$R_k(i) = \beta_k \log\left(1 + \rho_k(i)\right) \tag{6}$$

where $\rho_k(i)$ is the received signal-to-noise ratio (SNR) at the server from edge device k in the i-th global iteration.

Similarly, when the server transmits the global parameter to edge devices, the transmission rate to edge device k is given as

$$R'_k(i) = \beta_k \log\left(1 + \rho'_k(i)\right) \tag{7}$$

where $\rho'_k(i)$ is the received SNR at the edge device k in the i-th global iteration.

When the server sends the partition of the data set to each edge device, the size of data set to transmit is large enough to experience various fading channel realizations. Hence, the transmission rate for distributing data is set to be ergodic rate given by

$$\overline{R}_k = \mathbb{E}_{\rho'_k}\left[\beta_k \log\left(1 + \rho'_k\right)\right]. \tag{8}$$

III. Optimal number of edge devices

As edge devices can work simultaneously, the computing time can be reduced proportionally to the number of edge devices. In addition to that, the computing load of edge devices decreases as the number of edge devices increases. However, exploiting parallelism necessitates communication between the PS and edge devices. Furthermore, as the number of edge devices increases, time to transferring the local updates and global parameter becomes longer. Hence, there exists a tradeoff between exploiting parallel computing and reducing communication time. In this section, we investigate the optimal number of edge devices that optimizes the average completion time subject to communication constraints for achieving a certain duality gap.

To characterize the time required to achieve a certain duality gap, we first derive the time for each of the four phases in the distributed learning process. The first phase involves distributing the partitioned data to each edge device. Let S denote the size of single data. Since the transmission rate for distributing data set is given by (8), the time T_k^{dist} required for transmitting data set $\mathbf{X}_{[k]}$ to edge device k can be expressed as

$$T_k^{\text{dist}} = \frac{n_k S}{\mathbb{E}_{\rho'_k}\left[\beta_k \log\left(1 + \rho'_k\right)\right]}. \tag{9}$$

Considering T_k^{dist} can be different depending on n_k and the distribution of ρ'_k, T^{dist}, which is defined as the time to deliver the entire data set to all edge devices, is given by the longest time among edge devices, i.e.,

$$T^{\text{dist}} = \max_{k \in \mathcal{K}} T_k^{\text{dist}}. \tag{10}$$

After distribution of data set, edge device k solves the local subproblem to find optimal local update in every global iteration. It is assumed that all edge devices use gradient descent(GD) to solve local subproblems. It is known that the number of iterations required to have ϵ_l-accuracy with GD is $O\left(\frac{1}{\epsilon_l}\right)$. Hence, this implies that edge devices need local iterations proportional to $\frac{1}{\epsilon_l}$ to achieve ϵ_l-accuracy of their solutions. Furthermore, the processing time to calculate a gradient is proportional to the number of data examples given to edge devices. Hence, the time for solving a local subproblem by edge device k is given by

$$T_k^{\text{local}} = c\frac{n_k}{\epsilon_l} \tag{11}$$

where c is proportional constant.

Depending on the size of data allocated to edge devices, computing time can be different. Furthermore, the server needs to wait until receiving from all edge devices. Thus, the time for local computing is defined as the maximum of T_k^{local}, i.e.,

$$T^{\text{local}} = \max_{k \in \mathcal{K}} T_k^{\text{local}}. \tag{12}$$

The output of this local computation is transmitted to the server for calculating the global parameter for the next iteration. Suppose the size of the local update is S'. Since the transmission rate of edge devices is given by (6), the time to upload the local update for edge device k at the i-th global iteration is obtained as

$$T_k^{\text{up}}(i) = \frac{S'}{\beta_k \log\left(1 + \rho_k(i)\right)}. \tag{13}$$

Since the server needs to aggregate the local updates from all edge devices, time for uploading should account for the edge device of which channel quality is the lowest. Therefore, the time for uploading all local updates at i-th global iteration, $T^{\text{up}}(i)$, is given by

$$T^{\text{up}}(i) = \max_{k \in \mathcal{K}} T_k^{\text{up}}(i). \tag{14}$$

After receiving local updates from all edge devices, the global parameter for the next (global) iteration can be generated following Algorithm 1. The time for aggregation, which involves a simple addition operation, is ignored. Once a a new global parameter is generated for the next round, the server delivers this global parameter to the edge devices. Since the global parameter is common information to all edge devices, multicast transmission which is more efficient than unicast can be applied. However, the rate for multicast transmission is determined so that all edge devices can decode correctly. Hence, the minimum rate of K edge devices becomes the transmission rate of multicast for delivering the global parameter. Provided that the size of global parameter is S'', the time for multicast at i-th global iteration can be expressed as

$$T^{\text{mul}}(i) = \frac{S''}{\log(1 + \min_k \rho'_k(i))}. \tag{15}$$

For simplicity, we assume that edge devices are synchronized at each phase. In the asynchronous case, each edge device does not need to wait for the other edge devices to finish data reception. Hence, edge devices that has received data allocated can start solving the local subproblem earlier. Moreover, the transmission of local parameters can start earlier than in a synchronous system, thus reducing the completion time. Nevertheless, the server still needs to wait for all local updates from all edge devices. Hence, due to the bottleneck of aggregation step, completion time for asynchronous case could be approximated - under some assumptions - by the completion time of the synchronous case. Thus, if we denote the time required for i-th global iteration as $T^{\text{global}}(i)$,

$$T^{\text{global}}(i) = T^{\text{local}} + T^{\text{up}}(i) + T^{\text{mul}}(i). \tag{16}$$

Clearly, the data distribution occurs only once, whereas all other steps are repeated at every global iteration. From [7], we need to iterate at least $\lceil T_G \rceil$ times to achieve ϵ_G. Therefore, T^{DL}, which represents the completion time so that the distributed learning achieves a duality gap ϵ_G, can be calculated as

$$T^{\text{DL}} = T^{\text{dist}} + \sum_{i=1}^{\lceil T_G \rceil} T^{\text{global}}(i). \tag{17}$$

Since T^{DL} depends on wireless fading, which is a random variable, our goal is to minimize the average completion time of distributed learning, which can be formulated as

$$\min_K \mathbb{E}\left[T^{\text{DL}}\right]. \tag{18}$$

A closed-form expression of the objective function $\mathbb{E}\left[T^{\text{DL}}\right]$ requires an expression for $\mathbb{E}_{\rho_k}\left[\frac{1}{\log(1+\rho_k)}\right]$. However, $\mathbb{E}_{\rho_k}\left[\frac{1}{\log(1+\rho_k)}\right]$ does not exist for widely used fading distributions. Therefore, we use Jensen's inequality to approximate the expectation with a lower bound, i.e.,

$$\mathbb{E}_{\rho_k}\left[\frac{1}{\log(1+\rho_k)}\right] \geq \frac{1}{\mathbb{E}_{\rho_k}\left[\log(1+\rho_k)\right]}. \tag{19}$$

Furthermore, since (18) is a scalar optimization problem, the optimal number of edge devices can be easily found using standard algorithms, such as branch-and-bound.

IV. Simulations

In this section, we provide our simulation results on the completion time and the computation-communication tradeoff. The default simulation parameters are as follows: $N = 10000$, $\lambda = 0.01$, $\gamma = \frac{1}{K}$, $\epsilon_l = 0.01$, $\epsilon_G = 0.01$, $S = 1$ Mbits, $S' = 100$ kbits, $S'' = 100$ kbits, $c = 10^{-6}$, and a bandwidth of 10 MHz is uniformly allocated to edge devices. We also assume independent and identically distributed Rayleigh fading with mean $\overline{\rho}$ between the server and the edge devices.

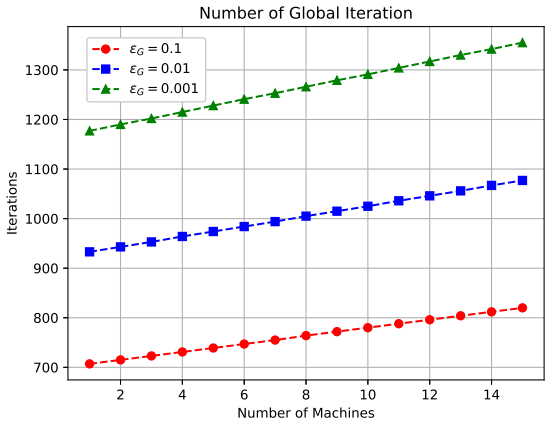

Fig. 2. Required number of global iterations vs. the number of devices for achieving a certain ϵ_G.

First, we plot the number of global iterations required for achieving a certain duality gap ϵ_G as a function of the number of edge devices in Fig. 2. When the server aggregates local updates, the factor γ, which is set as $\frac{1}{K}$, averages out the updates of all edge devices. As a result, the pre-log term in (5) becomes a linear function of the number of edge devices, so the number of global iterations behaves very closely to a linear function.

In Fig. 3 we evaluate the completion time for different numbers of edge devices. When the number of edge devices is small, the completion time significantly decreases, implying that computational advantages of parallelism dominate the communication overhead. However, when the number of edge devices exceeds a certain number, the completion time starts increasing, as in this regime the system becomes communication-limited. In other words, in the low edge device regime, the system has enough wireless resources for high-rate communication between edge devices and the server. However, for large number of edge devices, time spent due to communication increases as bandwidth is shared among more devices and the minimum received SNR is low. In that regime, communication load dominates the reduction of computing time due to parallel computing.

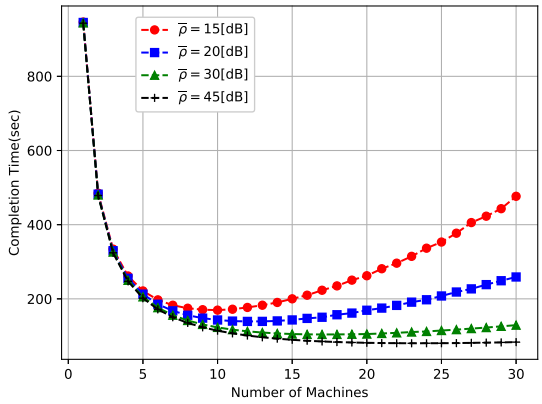

Fig. 3. Completion time vs. number of devices for different average received SNR.

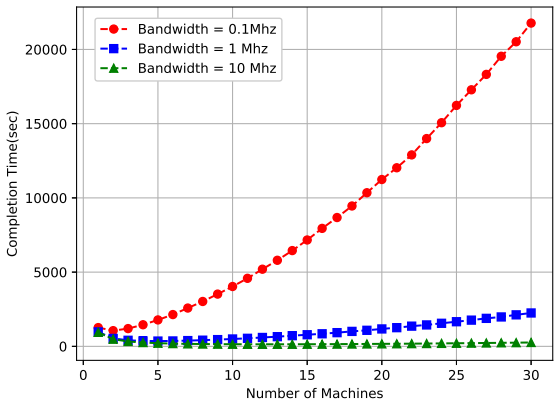

Fig. 4. Completion time vs. number of edge devices for different bandwidth.

In Fig. 4, we study the effect of bandwidth in the completion time. Expectedly, when bandwidth is scarce, the completion time dramatically increases, whereas for relatively wide bandwidth, the completion time does not significantly alter with increasing the number of edge devices.

The dependency of the completion time on the accuracy for the local subproblems is shown in the Fig. 5. We observe that adding edge devices could significantly drop the completion time, when high accuracy of the local updates is required. However, for less stringent requirements on accuracy, the completion time does not remarkably change for increasing number of edge devices.

V. Conclusion

In this work, we investigated the optimal number of edge devices in a distributed edge learning system with over-the-air model exchange. Accounting for the wireless channel characteristics, the completion time minimization is formulated and solved numerically. Although a large number of edge devices could bring significant computational advantages, the

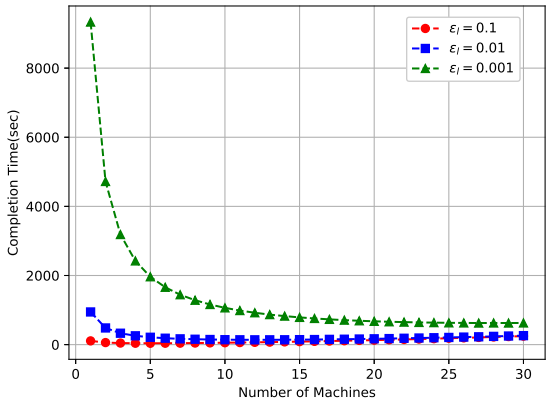

Fig. 5. Completion time vs. number of devices for different accuracy values of the local subproblem.

associated communication overhead may degrade the overall performance. Furthermore, when wireless resources are scarce, the number of edge devices should be carefully selected in order to optimize the distributed edge learning performance under latency constraints.

Acknowledgment

This work has been supported by the EURECOM-Huawei Chair on Advanced Wireless Networks.

References

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

[2] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proc. Machine Learning Research,* vol. 54, pp. 1273-1282, Apr. 2017.

[3] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *Proc. NIPS Workshop on Private Multi-Party Machine Learning,* 2016.

[4] C. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Zomaya, V. Gramoli, "Federated learning over wireless networks: convergence analysis and resource allocation," [Online]. Available: https://arxiv.org/abs/1910.13067.

[5] E. Koyuncu, "The tradeoff between coverage and computation in wireless networks," [Online]. Available: https://arxiv.org/abs/1910.13502.

[6] S. Wang, T. Tuor, T. Salonidis, K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205-1221, June 2019.

[7] C. Ma, J. Konen, M. Jaggi, V. Smith, M. I. Jordan, P. Richtrik, and M. Tak, "Distributed optimization with arbitrary local solvers," in *Optimization Methods and Software*, vol. 32 no. 4, 813-848, 2017.

[8] H. H. Yang, Z. Liu, T. Q. S. Quek and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. on Commun.*, vol. 68, no. 1, pp. 317-333, Jan. 2020.