Optimal Device Selection for Federated Learning over Mobile Edge Networks

Cheng-Wei Ching[†], Yu-Chun Liu[‡‖], Chung-Kai Yang[§‖], Jian-Jhih Kuo[†*], and Feng-Ting Su[†]

***Abstract*—Data privacy preservation has drawn much attention with emerging machine learning applications. Federated Learning is thus developed to offer decentralized learning on user devices. However, it is difficult to jointly address multiple issues such as device selection, upload scheduling, and payment minimization. To jointly optimize the issues above, we first formulate a new optimization problem, named TRAIN, to minimize the training cost (including incentive payment and upload time) while ensuring the data requirement. We then prove the NP-hardness and propose a 3-approximation algorithm, named DETECT to obtain a near-optimal solution. Simulation results manifest that DETECT reduces the training cost by 50% compared with other traditional methods and achieves high accuracy and short convergence time.**

***Index Terms*—federated learning, approximation algorithm, mobile edge network**

I. Introduction

Nowadays, the growing demands for Artificial Intelligence (AI) have grabbed much attention. Many advanced applications [1] enrich our daily life. However, due to the data-intensive property, current AI models have two critical issues. One is data security and privacy. Application service providers (ASPs) require enormous personal data to train AI models while the personal data is commonly stored in user devices [2]. Another is the need of powerful platforms for AI model training. To collect and process amount of data from devices, third-party ASPs have to afford plentiful transmission, computation, and storage resources. Nevertheless, most emerging ASPs cannot afford much infrastructure expenditure.

To remedy the above issues, Federated Learning (FL) sheds light on decentralized training framework [3]. The central server selects a set of user devices, broadcasts the global model, and each selected device trains the model with its local data [4]. Then, when the local computing time is up,[1] the central server starts to retrieve the local parameters from the selected devices to jointly update the parameters of global model in the central server. Accordingly, FL has following advantages: 1) Data privacy is secured by reserving data in the devices. 2) Data and computations are distributed among numerous devices, that is, the central server's burden is lower. 3) More data involved boost smarter services offered by third-party ASPs.

Intuitively, an appropriate amount of data and number of devices usually bring better accuracy and rate of convergence of trained models. Over-selection may require a high *incentive payment*, which is paid to users to have the aid of their computing and data, thereby including *computing expenditure* and *data expenditure* [6]. Besides, the *upload time* for parameters via networks from devices takes a large proportion of time [7]. The devices with low transmission rates may prolong the *upload time*, which is inevitable in mobile networks since devices resources are heterogeneous and the transmission rate depends on the relative location to the base station (BS) [8]. Also, a BS can simultaneously receive data from multiple channels while each device utilizes at most one channel at a time [9]. Nonetheless, the issues above are separately studied [10]–[12].

Jointly optimizing device selection and upload scheduling to minimize relevant costs (i.e., incentive payment and upload time) for updating a global model raises several new challenges as follows. 1) *Complicated upload time minimization*. Existing scheduling problems minimize the completion time for a set of tasks. The proposed problem, though, is more challenging since it not only schedules the uploadings of devices but shorten the upload completion time by shunning the devices with overlengh upload time. 2) *Training cost trade-off*. The data quantity requirement should be met to achieve the desired accuracy and convergence rate [13]. To reduce training cost, ASPs have to balance *incentive payment* [14] and *upload time* [15]. 3) *Obscure training cost*. It has to consider *incentive payment* and *upload time* jointly to minimize the training cost. Nevertheless, it is difficult to estimate the overall upload time of the selected devices. The more selected devices do not always lead to a longer overall upload time since they can upload the local parameters in parallel via multiple channels.

To address the above challenges, we present a new optimization problem, termed Time-Cost **Tr**ade-off for Feder**a**ted Learning via Smart Dev**i**ces over Mobile **N**etworks (TRAIN). With the following given parameters: 1) data quantity of each device, 2) incentive payment of each device (related to the data stored in a device), 3) upload time of each device, and 4) number of channels of the BS, TRAIN *finds a subset of devices* for training and schedules the upload sequence over the channels to minimize training cost including: 1) incentive payment and 2) upload time,[2] while ensuring *data quantity requirement* (i.e., the quantity of considered data should meet the provider requirement). Then, we prove that TRAIN is NP-hard and design a 3-approximation algorithm, Bi**d**-Bas**e**d **T**ime-Cost-Balanced D**e**vice Selection and Upload S**c**heduling Algori**t**hm (DETECT). To address *complicated upload time minimization*, DETECT sets distinct limits to device upload

[†]Dept. of Computer Science, National Chung Cheng University, Taiwan
[‡]Dept. of Electrical Engineering, National Chung Cheng University, Taiwan
[§]Dept. of Computer Science, National Tsing Hua Univerisity, Taiwan
* indicates the corresponding author; ‖ denotes the equal contributions.
Corresponding author's email: lajacky@cs.ccu.edu.tw

[1]The devices train the model in parallel, and each device can adjust the number of local updates within a round to mitigate synchronization delays in waiting for straggling and slow devices [5]. In this way, the devices can have similar local computing time. However, the upload time of each device depends on its transmission rate, and thus the upload scheduling is inevitable to achieve time-cost trade-off. Therefore, this paper focuses on upload scheduling.

[2]Note that our method can efficiently reduce upload time within a round in FL (i.e., the time for the devices to upload their local parameters to the central server) without sacrificing the convergence and accuracy shown in Section VI.

time to generate different candidate solutions for avoiding selecting devices with overlength upload time. Also, for each candidate solution, it adopts an intelligent *bidding process* for device selection to deal with *training cost trade-off* while meeting *data requirement*. Finally, to tackle *obscure training cost*, it introduces the notion of *binal cost* to estimate the upload time with a bounded error since finding the optimal scheduling is difficult. Simulation results based on the MNIST data manifest that DETECT outperforms the traditional approaches by 100% and reduce convergence time.

II. Related Work

A. Decentralized Learning Framework Evolution and Anaysis

Smith *et al.* extend the decentralized framework to multiple models according to the relations among multiple tasks [16]. Mohri *et al.* propose a new framework to mitigate the degradation of performance due to skewed data distributions [12]. Zhao *et al.* analyze the performance of models trained with non-skewed data and skewed data via FL and provide a data-sharing method to remedy the performance downgrade [17].

B. Decentralized Training Efficiency Optimization

Konečnỳ *et al.* optimize transmission efficiency between central servers and devices by reducing transmitted data sizes [10]. The approach above ignores device selections. Nishio *et al.* maximize the number of selected devices within a time period for FL [4]. Although device selection is considered, the optimization jointly examining device selection, upload scheduling, and payment minimization has not been explored.

III. The TRAIN Problem and Its Hardness

To train a model with FL, this paper considers a mobile edge network that consists of 1) a base station (BS) with a set of orthogonal channels M and an edge server which acts as a central server for FL and 2) a set of smart devices K with *non-identical* transmission rates for transmitting the training results to the central server. Once the local computing time is up,[3] the central server starts to take back the training results from the selected devices, and thus K only includes the devices that can finish the training before that time. Let $t(i)$ denote the upload time (length) of device $i \in K$ calculated by its transmission rate.[4] However, the upload completion time (denoted by T) cannot be acquired by summing up the upload time of all the selected devices. Instead, T should be carefully determined by scheduling the devices over $|M|$ channels of the BS. TRAIN is formulated as a *mixed-integer linear programming* as follows.

Let binary *decision variable* x_i denote whether to select device $i \in K$ and binary *decision variable* z_{im} denote whether device $i \in K$ uses channel $m \in M$. To ensure that a selected device should use a channel to upload the local parameters, we have the first constraint as follows.

$$\sum_{m \in M} z_{im} = x_i, \qquad \forall i \in K. \tag{1}$$

[3]The local computing time of each device is assumed to be similar by adjusting the number of local updates [5]. Please refer to the footnote 1.

[4]The transmission rate can be derived by Shannon capacity [18]. Moreover, to explore the intrinsic property of TRAIN, following [9], it is reasonable to assume that each device has the similar transmission rate via different channels.

TABLE I
Initial value

	U_1	U_2	U_3	U_4	U_5
$t(i)$	0.6	0.5	0.4	1.9	0.2
$D(i)$	440	350	300	550	250
$c(i)$	0.80	0.66	0.58	0.98	0.50

TABLE II
Upload time group

Group	l_h
$\mathcal{G}_1 = \{U_5\}$	0.2
$\mathcal{G}_2 = \{U_5,U_3\}$	0.4
$\mathcal{G}_3 = \{U_5,U_3,U_2\}$	0.5
$\mathcal{G}_4 = \{U_5,U_3,U_2,U_1\}$	0.6
$\mathcal{G}_5 = \{U_5,U_3,U_2,U_1,U_4\}$	1.9

Let variable T_m denote the upload time sum of selected devices via channel $m \in M$ and T denote the overall upload completion time of all selected devices (i.e., the time at which all the selected devices finish uploading). Thus, we have[5]

$$T_m = \sum_{i \in K} t(i) \cdot z_{im}, \qquad \forall m \in M. \tag{2}$$

$$T \geq T_m, \qquad \forall m \in M. \tag{3}$$

The *incentive payment* C includes: 1) *Computation expenditure* $e(i) \in \mathbb{R}^+$ representing the expenditure to request device $i \in K$ to train the global model with its local data and 2) *Data expenditure* $d(i) \in \mathbb{R}^+$ denoting the expenditure to locally access the data of device $i \in K$ for training. Let $c(i)$ denote the expenditure sum of device i, i.e., $c(i) = e(i) + d(i)$. Therefore,

$$C = \sum_{i \in K} x_i \cdot c(i). \tag{4}$$

To ensure the inference accuracy, the total amount of considered data must meet the *data quantity requirement* $\mathcal{D}$, i.e.,

$$\sum_{i \in K} x_i D(i) \geq \mathcal{D}, \tag{5}$$

where $D(i)$ denotes the data amount device $i \in K$ possesses.

Definition 1. Given a set of devices K, data quantity $D(i)$, data requirement $\mathcal{D}$, upload time $t(i)$, number of channels $|M|$, expenditure $d(i)$ and $e(i)$, the Time-Cost **Tr**ade-off for Feder**a**ted Learning via Smart Dev**i**ces over Mobile **N**etworks (TRAIN) finds a set of devices and schedules the uploadings of them to meet the constraints (1), (2), and (5) while minimizing the training cost including (3) and (4), i.e.,

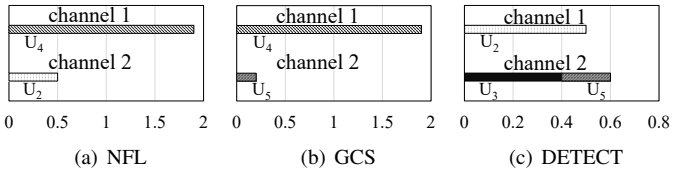
$$\text{minimize} \quad \alpha \cdot C + \beta \cdot T, \tag{6}$$

where parameters $\alpha, \beta \in \mathbb{R}^+ \cup \{0\}$ are *tuning knobs* (please refer to the simulation setting in Section VI) for ASPs to *differentiate the importance of incentive payment and upload time*. Remark that x_i and z_{im} are *decision variables* in TRAIN.

Example 1. The example shows how to select devices and evaluate incentive payment and upload time for different methods, where K and M include five devices (i.e., U_1, U_2, U_3, U_4 and U_5) and two channels. Table I summarizes the given values. Moreover, assume that $\alpha = \beta = 0.5$, and the given data requirement $\mathcal{D} = 800$. Fig. 1 compares the solutions of DETECT with two traditional approaches:[6] Naive Federated

[5]The upload completion time T in (3) seems unbounded, whereas the minimization of the objective (6) can limit T to be $\max_{m \in M} T_m$. Moreover, the objective (6) leaves us avoiding the devices with overlength upload time.

[6]Note that both the two traditional approaches only handle device selection, but neglect upload scheduling. To make their solutions feasible, we employ the existing algorithm for Parallel Machine Scheduling (PMS) [19] to complete the upload scheduling to generate the solutions (see Figs. 1(a) and 1(b)).



(a) NFL (b) GCS (c) DETECT

Fig. 1. Upload scheduling for selected devices

Learning (NFL) [3] and greedy-based client selection (GCS) [19]. NFL *randomly* selects devices until $\mathcal{D}$ is satisfied (e.g., U_2 and U_4) where relevant incentive payment and upload time are $0.98 + 0.66 = 1.64$ and 1.9 (see Fig. 1(a)), respectively. Therefore, the training cost is $0.5 \times 1.64 + 0.5 \times 1.9 = 1.77$ for NFL. By contrast, GCS iteratively selects the device with *the largest ratio*[7] *of effective data quantity to incentive payment*. It first selects U_4 with $\frac{550}{0.98}$ and then selects U_5 with $\frac{250}{0.5}$ rather than U_1 with $\frac{\min\{440,800-550\}}{0.8}$. Incentive payment and upload time are $0.98+0.5 = 1.48$ and 1.9 (see Fig. 1(b)), respectively. We have training cost of $0.5 \times 1.48 + 0.5 \times 1.9 = 1.69$. Finally, DETECT (detailed in Section IV) selects devices U_2, U_3 and U_5 (see Fig. 1(c)) and only requires the incentive payment $0.66 + 0.58 + 0.5 = 1.74$ and upload time 0.6 (see Fig. 1(c)). The training cost $0.5 \times 1.74 + 0.5 \times 0.6 = 1.17$ is also *optimal*. Compared with NFL and GCS, DETECT reduces training cost by 34% and 31% with the upload time speedup of 3.16x. ■

TRAIN is NP-hard since the NP-complete decision version of *Parallel Machine Scheduling Problem* (D-PMS) [19] can be reduced to the decision version of TRAIN (D-TRAIN)[8] in polynomial time. The proof is omitted due to the page limit.

IV. Algorithm Design

Two traditional approaches can be applied to TRAIN. One is the native Federated Learning algorithm (NFL) [3]. It *randomly selects devices until data requirement $\mathcal{D}$ is met* to train the global model. However, it can ensure neither minimum expenditure nor decent upload time. Another is a greedy-based client selection approach (GCS) [19]. It iteratively selects device with *the largest ratio of effective data quantity to incentive payment* until $\mathcal{D}$ is met.[7] Nonetheless, it neglects the upload time of selected devices. To solve TRAIN, we design an approximation algorithm named DETECT to carefully address training cost trade-off, complicated upload time minimization, and obscure training cost simultaneously. First, DETECT makes use of the novel notion of *binal cost* to approximate the exact training cost jointly including incentive payment and upload time. Specifically, the *binal cost* of each device i is defined as

$$\hat{c}(i) = \alpha \cdot c(i) + \beta \cdot (\frac{t(i)}{|M|}),$$

where $t(i)$ is the upload time of device i, $c(i)$, the expenditure for selecting device i to train the global model (i.e., $c(i) = e(i) + d(i)$), and $|M|$, the number of channels. The binal cost can be acquired in constant time and approximate the exact training cost with a bounded error (detailed later). However, a device with overlength upload time may still prolong upload completion time. Thus, DETECT constructs multiple solutions being considered in the end, and each solution has different *upload time limit*. That is, a device with upload time larger than the upload time limit cannot be selected into the solution.

TABLE III
First Iteration Initialization

	U_1	U_2	U_3	U_5
$\hat{c}(i)$	0.55	0.45	0.39	0.30
$s^\delta(i)$	440	350	300	250
b_i	0	0	0	0

TABLE IV
First Iteration Bidding Process

Device	Bidding Process	$y_\delta(i)$
U_1	$0 + 440 \times y_\delta(1) = 0.55$	0.00125
U_2	$0 + 350 \times y_\delta(2) = 0.45$	0.00128
U_3	$0 + 300 \times y_\delta(3) = 0.39$	0.00130
U_5	$0 + 250 \times y_\delta(5) = 0.30$	**0.00120**

Moreover, for each constructed solution, we design an intelligent *bidding process* for the ASP to gradually grow the *bid* to pay a fair cost. Explicitly, the bid offered by the ASP for each device i grows simultaneously at *different increase rate* denoted by $s^\delta(i)$, which is defined as

$$s^\delta(i) = \min\{D(i), \mathcal{D} - D(\delta)\}, \tag{7}$$

where $\delta \subseteq K$ denotes the current set of selected devices with data quantity $D(\delta)$ and thus initially $\delta = \emptyset$ and $D(\delta) = 0$. Once a device's bid increases to its *binal cost*, the device will be selected in the solution. Meanwhile, the increase rate of bid for each device will update according to the current selected devices by eq. (7). The bidding repeats until the total data quantity of selected devices meets the requirement $\mathcal{D}$.

DETECT includes the following four phases: 1) Candidate Devices (CD) Grouping, 2) Candidate Devices (CD) Gathering, 3) Candidate Devices (CD) Scheduling, and 4) Final Devices (FS) Selection. More specifically, CD Grouping first establishes multiple groups with different limits on upload time length and puts the devices into groups according to their upload time length for the later construction of candidate devices. For each group, CD Gathering then exploits the bidding process to select a set of suitable devices based on its binal cost until meeting data requirement. Afterward, for each group, CD Scheduling schedules the selected devices over the channels. FS Selection eventually chooses a solution with minimum cost among all the groups. To achieve the approximation ratio, it is important for CD grouping and CD Gathering to carefully examine the maximum upload time length and subtly exploit the binal cost in bidding processes to select suitable devices.

1) Candidate Devices (CD) Grouping: Initially, CD Grouping identifies and sorts all the devices with distinct upload time length in non-decreasing order. For each distinct upload time length l in non-decreasing order, CD Grouping establishes a group and puts every device whose upload time is no greater than l into the group. Therefore, CD Grouping obtains at most $|K|$ groups $\mathcal{G}_1, \mathcal{G}_2, ...$, where the maximum upload time length of devices in $\mathcal{G}_h$ is smaller than that in $\mathcal{G}_{h+1}$. That is,

$$\mathcal{G}_h = \{i \in K | t(i) \leq l_h\},$$

where l_h denotes the upload time limit of group $\mathcal{G}_h$. Thus, the last group includes all the devices. Later, to generate a candidate solution for each group $\mathcal{G}_h$, only the devices put in the same group can be selected. That is to say, CD

[7] Let K' denote the current set of selected devices. Initially, $K' = \emptyset$. GCS iteratively selects the device with $\max_{i \in K \setminus K'} \frac{\min\{D(i), \mathcal{D} - \sum_{j \in K'} D(j)\}}{c(i)}$.

[8] D-TRAIN asks whether a solution with a training cost less than a given cost threshold q exists.

TABLE V
SECOND ITERATION INITIALIZATION

	U_1	U_2	U_3
$s^\delta(i)$	440	350	300
b_i	*0.53*	*0.42*	*0.36*

TABLE VI
SECOND ITERATION BIDDING PROCESS

Device	Bidding Process	$y_\delta(i)$
U_1	$0.53 + 440 \times y_\delta(1) = 0.55$	**0.00005**
U_2	$0.42 + 350 \times y_\delta(2) = 0.45$	0.00008
U_3	$0.36 + 300 \times y_\delta(3) = 0.39$	0.00010

TABLE VII
CANDIDATE DEVICES

Group	Sel. Devices	Compl. Time	Incen. Paym.	Data Quan.	Train. Cost
$\mathcal{G}_3$	U_2, U_3, U_5	0.6	1.74	900	1.17
$\mathcal{G}_4$	U_1, U_3, U_5	0.6	1.88	990	1.24
$\mathcal{G}_5$	U_1, U_3, U_5	0.6	1.88	990	1.24

Grouping is designed to restrict the maximum upload time length for each group. Clearly, let l^* denote the maximum upload time of devices selected in the optimal solution. Then the optimal solution must only select the devices classified in the corresponding group $\mathcal{G}^*$ with the upload time restriction l^*. To some extent, the solutions generated by later phases for $\mathcal{G}^*$ stand more chance to achieve *complicated upload time minimization* since it avoids the devices with overlength upload time, which are also ignored by the optimal solution. Lastly, to avoid the redundant computing, CD Grouping deletes the groups whose total data quantity is less than $\mathcal{D}$.

Example 2. Table II follows Example 1 to illustrate an example of CD Grouping where the devices are grouped by distinct upload time length $t(i)$ (i.e., the first row of Table I). We have five groups in Table II, where l_h is the maximal upload time length in each group $\mathcal{G}_h$. Remark that if the maximum upload time length in the optimal solution is 0.5, all the selected devices in the optimal solution must be in group $\mathcal{G}_3$. ■

2) Candidate Devices (CD) Gathering: This phase develops an intelligent bidding process for each group $\mathcal{G}_h$. Initially, the selected devices δ in the constructed solution $\mathcal{G}_h^t$ is $\emptyset$ (i.e., no device selected). Each bid b_i for device $i \in \mathcal{G}_h \setminus \delta$ is initially set to zero and then gradually grows at different increase rate $s^\delta(i)$ until the bid of some device i^δ meets it's binal cost, i.e.,

$$i^\delta = \arg\min_{i \in \mathcal{G}_h \setminus \delta} y_\delta(i), \text{ where } y_\delta(i) = \frac{\hat{c}(i) - b_i}{s^\delta(i)}. \tag{8}$$

Note that $y_\delta(\cdot)$ is used to find the first-saturated device (i.e., the device to be selected in this iteration) when the set of selected devices is δ. Therefore, the increase unit of bid for all devices not in δ is set to

$$y_\delta = \min_{i \in \mathcal{G}_h \setminus \delta} y_\delta(i). \tag{9}$$

The bid b_i of each device $i \in \mathcal{G}_h \setminus \delta$ is updated to $b_i + s^\delta(i) \cdot y_\delta$, and then device i^δ is selected in the solution, i.e., $\delta = \delta \cup \{i^\delta\}$. The bidding repeats until the total data quantity meets the requirement, i.e., $D(\delta) \geq \mathcal{D}$. For each group $\mathcal{G}_h$, CD Gathering employs the bidding process to obtain the solution $\mathcal{G}_h^t$. Remark that the binal cost does not precisely capture the upload completion time. Instead, the binal cost is exploited to circumvent the obscure training cost during the device selection. Therefore, the next phase still has to schedule the device uploads for each candidate solution to approximate the exact training cost.

Example 3. Table III follows Example 2 and shows the initial variables for each device. Take group $\mathcal{G}_4$ as an example for CD Gathering where four devices are available for selection. In the first iteration, $\delta = \emptyset$ and $b_i = 0$, $\forall i \in \mathcal{G}_4$. Table IV lists four equations that bound the increase of y_δ implying $y_\delta = 0.0012$ by (9). Thus we update $\delta = \{U_5\}$ by (8). Remark that $s^\delta(i)$ does not change since $D(i) < \mathcal{D} - D(\delta)$ holds in (7). Then, Tables V and VI show the updated b_i and the equations for remaining three devices. For example, $b_1 = 0 + 440 \cdot 0.0012 \approx 0.53$ for device U_1. By (8), we can obtain $\delta = \{U_5, U_1\}$. Eventually, we can derive $\mathcal{G}_4^t = \delta = \{U_5, U_1, U_3\}$. CD Gathering obtains a candidate solution for each group. ■

3) Candidate Devices (CD) Scheduling: This phase schedules the selected devices $\mathcal{G}_h^t$ for each group $\mathcal{G}_h$. It is natural to assume the number of channels is usually smaller than that of devices (i.e., $|M| < |K|$). Otherwise, each device can be assigned to a different channel, which explicitly indicates no need for scheduling. In particular, for each group $\mathcal{G}_h$ with a set of selected source devices $\mathcal{G}_h^t$ that meets the user requirement (i.e., $D(\mathcal{G}_h^t) \geq \mathcal{D}$), CD Scheduling first sorts the selected devices in $\mathcal{G}_h^t$ in non-increasing order by upload time length of devices. Then, CD Scheduling iteratively assign the devices in $\mathcal{G}_h^t$ by its upload time in non-increasing order to a channel with the lowest current upload completion time until all the devices are assigned. Let $\mathcal{G}_h^t(m)$ denote the set of devices in $\mathcal{G}_h^t$ assigned to channel $m \in M$. Then, CD Scheduling finishes scheduling and obtains the approximated upload completion time $\tau(\mathcal{G}_h^t)$ for each group $\mathcal{G}_h$ over the channels, i.e.,

$$\tau(\mathcal{G}_h^t) = \max_{m \in M} \sum_{k \in \mathcal{G}_h^t(m)} t(k), \quad \text{for all groups } \mathcal{G}_1, \mathcal{G}_2, ...$$

Thus, the exact cost of the selected devices $\mathcal{G}_h^t$ for each group $\mathcal{G}_h$ can be approximated by the following formula.

$$c(\mathcal{G}_h^t) = \alpha \sum_{k \in \mathcal{G}_h^t} c(k) + \beta \cdot \tau(\mathcal{G}_h^t), \quad \text{for all groups } \mathcal{G}_1, \mathcal{G}_2, ...$$

Remark that CD Scheduling only exploits $\tau(\mathcal{G}_h^t)$ to approximate the minimum upload completion time for scheduling the devices in $\mathcal{G}_h^t$ since finding the minimum upload completion time for scheduling over multiple channels is NP-hard.

Example 4. Table VII follows Example 3 to show an example of CD Scheduling where only three candidate solutions meet the requirement in Table VII. For example, the candidate solution for group $\mathcal{G}_3$ selects devices U_2, U_3, U_5 with a scheduling of upload completion time 0.6 in Fig 1(c). ■

4) Final Solution (FS) Selection: The final phase selects the solution with device set $\mathcal{G}'$ among all the device sets $\mathcal{G}_1^t, \mathcal{G}_2^t, ...$ for all groups $\mathcal{G}_1, \mathcal{G}_2, ...$ that 1) has the minimum approximated cost and 2) guarantees the total performance meeting the ASP's requirement. That is,

$$\mathcal{G}' = \arg\min_{\mathcal{G}_h^t : D(\mathcal{G}_h^t) \geq \mathcal{D}} c(\mathcal{G}_h^t).$$

Example 5. Following Example 4, this example demonstrates FS Selection. FS Selection obtains the training cost of the three candidate solutions for $\mathcal{G}_3$, $\mathcal{G}_4$ and $\mathcal{G}_5$, which are $0.5 \times 1.74 + 0.5 \times 0.6 = 1.17$, $0.5 \times 1.88 + 0.5 \times 0.6 = 1.24$, and 1.24

respectively, as shown in Table VII. Thus, FS Selection chooses the minimum-cost candidate solution $\mathcal{G}_3^t$ of group $\mathcal{G}_3$. ■

V. Theoretical Analysis

The time complexity and approximation ratio are as follows.

Time Complexity. The time complexity is $O(|K|^3)$. We omit the details due to page limit. DETECT is efficient and can be accelerated with multiple processors (see Section VI). ■

Theorem 1. DETECT is a 3-approximation algorithm.

Proof. Let $\mathcal{G}'$ denote the set of devices with training cost $c(\mathcal{G}')$ selected by FS Selection, where all devices in $\mathcal{G}'$ are iteratively selected from group $\mathcal{G}_{h'}$ by CD Gathering (i.e., $\mathcal{G}' = \mathcal{G}_{h'}^t$). There exist two possible cases for the maximum upload time l' of device set $\mathcal{G}'$. Consider the first case, l' is equal to the maximum upload time l^* of device set $\mathcal{G}^*$ selected by the optimal solution, and $l' \neq l^*$ in the second case. In the following, let c^* denote the optimal training cost.

For the first case, we calculate the upload completion time $\tau(\mathcal{G}')$ for devices in $\mathcal{G}'$. Since CD Scheduling iteratively assigns the devices with the maximum upload time to the channel with the lowest workload, the device with the last upload completion time in the scheduling must start transmitting before the time $\frac{\sum_{i\in\mathcal{G}'} t(i)}{|M|}$. Besides, the upload time length of last device must be no longer than that of the longest one, $\max_{i\in\mathcal{G}'} t(i)$. Therefore, we know

$$\begin{aligned} c(\mathcal{G}') &= \alpha \sum_{i\in\mathcal{G}'} c(i) + \beta \cdot \tau(\mathcal{G}') \\ &\leq \alpha \sum_{i\in\mathcal{G}'} c(i) + \beta\left(\frac{\sum_{i\in\mathcal{G}'} t(i)}{|M|} + \max_{i\in\mathcal{G}'} t(i)\right) \\ &= \sum_{i\in\mathcal{G}'} \hat{c}(i) + \beta\left(\max_{i\in\mathcal{G}'} t(i)\right) \leq \sum_{i\in\mathcal{G}'} \hat{c}(i) + c^*. \end{aligned} \quad (10)$$

The last inequality holds since the optimal solution has a device with upload time $\max_{i\in\mathcal{G}'} t(i)$ in the first case. Let $y_\delta = 0$ if y_δ is not set by (9) to generate solution $\mathcal{G}'$ and thus we know

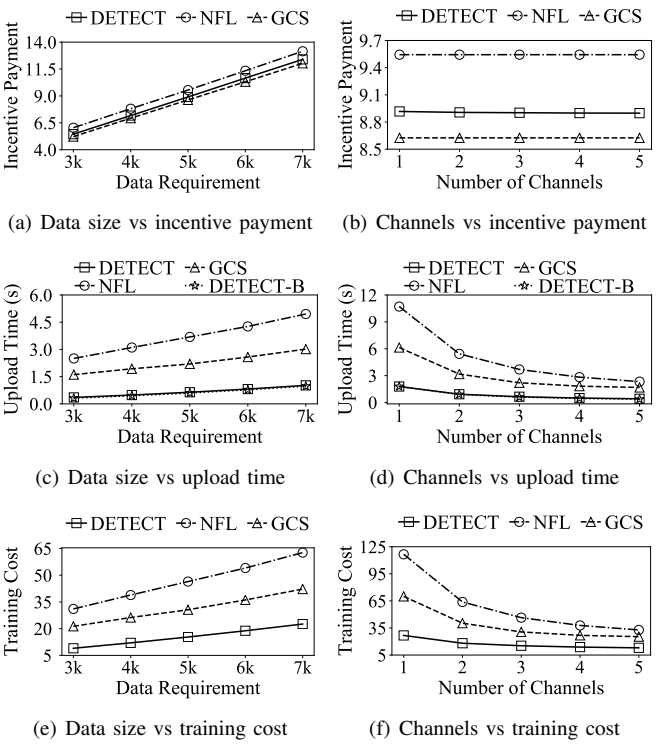
$$\sum_{i\in\mathcal{G}'} \hat{c}(i) = \sum_{\delta\subseteq\mathcal{G}_{h'}} y_\delta \sum_{i\in\mathcal{G}'\setminus\delta} s^\delta(i).$$

Let i_l be the last device in $\mathcal{G}'$ selected by DETECT. Thus, $D(\mathcal{G}') \geq \mathcal{D}$ and $D(\mathcal{G}' \setminus \{i_l\}) < \mathcal{D}$. Hence, for all the devices $i \in \mathcal{G}'$ except i_l, $s^\delta(i) = \min\{D(i), \mathcal{D} - D(i)\} = D(i)$ at that time in CD Gathering where δ was the set of devices at that point. Then, let $R(\delta) = \mathcal{D} - D(\delta)$, and we have

$$\sum_{i\in\mathcal{G}'\setminus\delta} s^\delta(i) = s^\delta(i_l) + D(\mathcal{G}' \setminus \{i_l\}) - D(\delta) < 2R(\delta).$$

The last inequality holds since $s^\delta(i_l) = \mathcal{D} - D(i) \leq R(\delta)$ and $D(\mathcal{G}' \setminus \{i_l\}) - D(\delta) < \mathcal{D} - D(\delta) = R(\delta)$. Then, we can see that for any $\delta \subseteq \mathcal{G}^*$, $R(\delta) \leq \sum_{i\in\mathcal{G}^*\setminus\delta} s^\delta(i)$. Thus, due to (9),

$$\begin{aligned} \sum_{i\in\mathcal{G}'} \hat{c}(i) &< 2 \sum_{\delta\subseteq\mathcal{G}_{h'}} y_\delta R(\delta) \leq 2 \sum_{\delta\subseteq\mathcal{G}_{h'}} y_\delta \sum_{i\in\mathcal{G}^*\setminus\delta} s^\delta(i) \\ &= 2 \sum_{i\in\mathcal{G}^*} \sum_{\delta\subseteq\mathcal{G}_{h'}: i\notin\delta} y_\delta s^\delta(i) \leq 2 \sum_{i\in\mathcal{G}^*} \hat{c}(i) \leq 2c^*. \end{aligned} \quad (11)$$



(a) Data size vs incentive payment (b) Channels vs incentive payment

(c) Data size vs upload time (d) Channels vs upload time

(e) Data size vs training cost (f) Channels vs training cost

Fig. 2. Effect of different parameters on different metrics.

By (10) and (11), we have $c(\mathcal{G}') \leq 3c^*$. The first case holds.

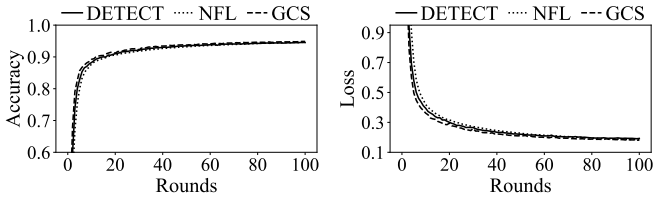
For the second case, recall that FS Selection examines the candidate solution for each possible maximum upload time length and then selects the one with the minimum training cost. Since the candidate solution with the maximum upload time length l^* must be examined by FS Selection, $c(\mathcal{G}')$ must be bounded within $3c^*$. The theorem follows. □

VI. Performance Evaluation

A. Simulation and Implementation Settings

DETECT is compared with two traditional approaches NFL [3] and GCS [19] (introduced in Example 1 and Section IV) on several devices that connect to a BS with multiple channels. We extract 55000 images in [20] for training and distribute them to 100 devices (i.e, $D(i)$ is randomly distributed) with Gaussian distribution, and leave 10000 images for testing. A convolutional neural network is adopted as the global model which comprises two 3×3 convolutional layers (the 1^{st} and 2^{nd} layers with 32 and 64 channels, each of which is activated by ReLU) and followed by one 2×2 max pooling with dropout, flatten, and one fully connected layer with 128 units with dropout (with ReLU activation and another 10 units activated by soft-max). The local minibatch size, the number of local epochs, and the learning rate are set to 64, 5, and 0.01.

For incentive payment and upload time, we set $e(i)$ randomly with a mean of 0.104 and set $d(i) = 0.001584 \times D(i)$ (i.e., 0.001584 per datum) [21], [22], and $t(i)$ randomly from 0.1 to 2 (i.e., seconds for transferring one-megabyte local parameters with data rate from 4 to 80 Mbps in 4G LTE). The default $|M|$ and $\mathcal{D}$ are set to 3 and 5K, respectively. We adopt 100-round FL and α and β are set to 1 and 10. The above parameters



(a) Effect on accuracy (b) Effect on loss

Fig. 3. Accuracy and loss of different approaches.

are changed to observe the metrics including 1) training cost, 2) incentive payment, 3) upload time, 4) accuracy, and 5) loss. Each simulation result is averaged over 100 samples.

B. Training Cost Trade-Off and Obscure Training Cost

Overall, the training cost increases as $\mathcal{D}$ increases while decreasing as $|M|$ goes up as shown in Fig. 2. GCS minimizes incentive payment, but suffers from longer upload time. By contrast, DETECT benefits from using the *bidding process* to reduce more than 50% upload time and achieves training cost trade-off. Note that neither of NFL and GCS changes incentive payment as the number of channels increases in Fig. 2(b). However, due to *binal cost*, more channels induce DETECT to select the devices with a higher upload time yet a slightly lower incentive payment (from 8.91 to 8.89). Moreover, Figs. 2(c) and 2(d) show the error between the exact cost and binal cost (DETECT-B). Obviously, the exact and binal costs grow together and their difference is bounded within $\beta \cdot \max_{i \in \mathcal{G}'} t(i)$, which meets Theorem 1 and reveals the obscure training cost.

C. Complicated Upload Time Minimization and Convergence

Overall, the upload time of all approaches declines as the number of channels increases in Figs. 2(b), 2(d), and 2(f). DETECT outperforms others since it carefully examines every device to avoid selecting the devices with overlength upload time and reduce the idle time in most of the channels. Besides, Figs. 3(a) and 3(b) manifest that DETECT barely sacrifices accuracy and convergence. Table VIII also shows that DETECT reduces the convergence time by more than 20% compared with other approaches since it optimizes upload scheduling.

D. Running Time with Different Number of Processors

For DETECT, the speedup factor can be almost linear to the number of processors as shown in Table IX. DETECT has the strong scalability since CD Grouping creates multiple device groups and any two groups have no data dependencies. Thus, each device group can be processed by CD Gathering and CD Scheduling in parallel. Finally, FS Selection waits for the slowest processor to collect all the candidate solutions.

VII. Conclusions

In this paper, a new optimization problem, TRAIN, is studied to jointly consider incentive payment and upload time to foster proper decisions for device selection and upload scheduling in FL. It is more difficult than traditional problems due to the challenges, training cost trade-off, complicated upload time minimization, and obscure training cost. We propose a novel algorithm to subtly limit upload time length and exploit insightful notions, *binal cost*, and *bidding process*, to achieve an approximation ratio. Simulation results manifest that DETECT reduces more than 50% training cost compared with the existing approaches. For future research, it will be interesting to modify DETECT for networks in which devices have different computing time. Another one is to extend DETECT to consider multiple FL tasks requested by ASPs at the same time.

TABLE VIII
Convergence time of Different Approaches (sec)

Accuracy	80%	84%	88%	92%
DETECT	12.77 (1x)	15.32 (1x)	25.54 (1x)	68.96 (1x)
NFL	25.05 (1.96x)	35.02 (2.29x)	55.04 (2.15x)	165.11(2.39x)
GCS	15.91 (1.25x)	19.89 (1.30x)	31.82 (1.25x)	95.46 (1.39x)

TABLE IX
Running Time with Different Number of Processors

Number of Processors	1	2	4	8
Running Time (sec)	0.3352	0.1680	0.0845	0.0426
Speedup Factor	1x	1.99x	3.97x	7.87x

References

[1] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[2] T. Yang *et al.*, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[3] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[4] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE ICC*, 2019.

[5] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource constrained distributed machine learning," in *IEEE INFOCOM*, 2018.

[6] Y. Zhang and M. Van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *IEEE INFOCOM*, 2012.

[7] S. Shi, X. Chu, and B. Li, "MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms," in *IEEE INFOCOM*, 2019.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 2322–2358, 2017.

[9] S.-H. Hsu, C.-H. Lin, C.-Y. Wang, and W.-T. Chen, "Breaking bandwidth limitation for mission-critical IoT using semisequential multiple relays," *IEEE Internet Things J.*, vol. 5, pp. 3316–3329, 2018.

[10] J. Konečnỳ *et al.*, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on PMPML*, 2016.

[11] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning based on Over-the-Air computation," in *IEEE ICC*, 2019.

[12] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *ICML*, 2019.

[13] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[14] Y. Gao, Y. Chen, and K. R. Liu, "On cost-effective incentive mechanisms in microtask crowdsourcing," *IEEE Trans. Comput. Intell. AI in Games*, vol. 7, pp. 3–15, 2014.

[15] S. R. Pandey *et al.*, "A crowdsourcing framework for on-device federated learning," *IEEE Transactions on Wireless Communications*, 2020.

[16] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *NIPS*, 2017.

[17] Y. Zhao *et al.*, "Federated learning with Non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[18] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[19] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[20] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[21] Google cloud storage pricing. [Online]. Available: https://cloud.google.com/storage/pricing#operations-pricing

[22] Coinmarketcap. [Online]. Available: https://coinmarketcap.com/currencies/datum/