Orchestrating Blockchain with Decentralized Federated Learning in Edge Networks

Yibo Jin[1], Lei Jiao[2], Zhuzhong Qian[1], Ruiting Zhou[3], Lingjun Pu[4]
[1]Nanjing University, China [2]University of Oregon, USA
[3]Southeast University, China [4]Nankai University, China

***Abstract*—Decentralized federated learning across edge networks can leverage blockchain with consensus mechanisms for training information exchange among participants over costly and distrustful wide-area networks. However, it is non-trivial to optimally operate the blockchain to support decentralized federated learning due to the complex cost structure of blockchain operations, the balance between blockchain overhead and model convergence, and the dynamics and uncertainties of edge network environments. To overcome these challenges, we formulate a non-linear time-varying integer program that jointly places blockchain nodes and determines the number of training iterations to minimize the long-term blockchain computation and communication cost. We then design an online polynomial-time approximation algorithm that decomposes the problem and solves the subproblems alternately on the fly using only estimated inputs. We rigorously prove the sublinear regret of our approach. We further implement our approach with a prototype system, and conduct extensive trace-driven experiments to validate the superiority of our approach over other alternatives.**

I. Introduction

Federated learning often adopts a star topology consisting of multiple rounds of training where in each round the participating devices train local models and send them to the central server for aggregation [1, 2]. This paradigm suffers from server congestion, single point of failure, single point of attack, and straggling participants. To overcome these performance issues, one approach is to decentralize the federated learning process, i.e., letting participants directly exchange information among themselves and conduct iterative aggregations to train the model collaboratively [3, 4]. Yet, this is still not a panacea, especially when such decentralized federated learning is deployed and operated across multiple network carriers or Internet Service Providers (ISPs) [5]. In this case, information exchange among participants through Wide Area Networks (WANs) could be costly and distrustful [6], impairing the quality and accuracy of the models being trained.

Blockchain can be a promising solution to facilitate across-WAN decentralized federated learning by providing consensus mechanisms to ensure consistent information sharing among federated learning participants [7]. In fact, blockchain provides encryption, verification, and immutability, among others, which could all be useful for securing and protecting privacy for decentralized federated learning. As a distributed ledger, blockchain can further provide incentivization functionalities by tracing each participant's contribution of training data. In this paper, we study the scenario as shown in Fig. 1, where a service provider would like to leverage a blockchain to help

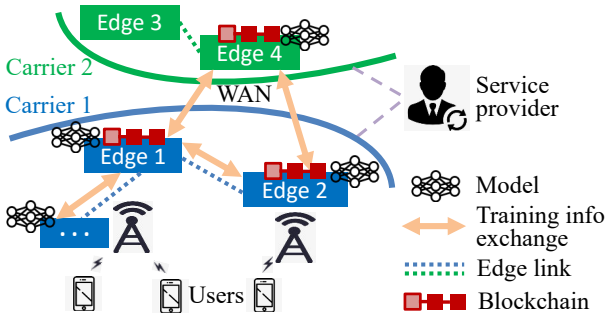


Fig. 1: Decentralized federated learning upon blockchain

with its decentralized federated learning across edge servers in different networks operated by multiple carriers.

Unfortunately, provisioning the blockchain optimally to support decentralized federated learning faces multiple challenges.

First, the coexistence of the blockchain and the decentralized federated learning requires careful orchestration upon a complex cost structure. Traditionally, the consensus achieved by blockchain is built upon any node that joins the blockchain. Yet, the exchange of training information (e.g., intermediate models and/or gradients) in decentralized federated learning is directed—only desired participants are supposed to receive it. Consequently, permission control is needed for the blockchain. That is, in each edge network, one may choose edge servers among those which exchange training information with others across WANs to serve as blockchain nodes, while considering the costs of blockchain operations including raw data encryption and decryption, intra- and across-WAN communication, and "Proof of Work" for competing for new blocks [8, 9].

Second, the overhead of the blockchain and the quality (e.g., convergence) of the model being trained by the decentralized federated learning need to be balanced. To reduce the overhead of the blockchain operations triggered by decentralized federated learning, one may decrease the frequency of training information exchange in terms of the number of training iterations; however, the model convergence actually relies on sufficient training. The best suitable number of training iterations needs to be found to address this trade-off. Unfortunately, before actually performing the training process, it is typically hard to calculate a precise number of training iterations.

Third, the edge network dynamics and uncertainties further hamper us from continuously controlling the efficient provisioning of the blockchain with decentralized federated learn-

ing. For example, as the transmission cost over WAN varies over time [6], it is desirable to reconfigure the blockchain placement accordingly. Note that such input variations about the network environment are often unpredictable beforehand, and we can only make changes to the blockchain placement using the estimated inputs. What could be worse is that the estimations could often be inaccurate compared to the actual inputs revealed afterwards, and mislead the control decisions irrevocably. It is therefore non-trivial to optimize the total cumulative cost in the long run in this situation.

Existing research falls insufficient for addressing the aforementioned challenges. Some focus on the design and optimization of federated learning at edge [10–14] and others focus on the management and application of blockchains at edge [8, 9, 15–17], both covering only part of the scope and not capturing the problem targeted in this paper. Those few on decentralized federated learning with blockchains [18–22] neither orchestrate the two systems jointly across WANs nor consider online optimization under uncertainties.

We firstly model and formulate the optimization problem that jointly places the nodes in the blockchain and determines the number of training iterations in the decentralized federated learning across edge networks. Our formulation optimizes over continuous time epochs the total computation and communication cost of the blockchain triggered by the decentralized federated learning, subject to the requirements of ensuring eventual model convergence with at least one blockchain node in each edge network at each time epoch. Our problem turns out to be a non-linear integer program, which is NP-hard, and is general to capture arbitrary time-varying system dynamics.

To solve this problem in polynomial time in an online manner using only estimated inputs, we then design an online approximation algorithm that decomposes the problem carefully into two subproblems and solves these two subproblems alternately on the fly. In the first subproblem, we determine the fractional number of training iterations for the current time epoch based on the blockchain node placement in the previous time epoch via a primal-dual-based online learning approach [23]. This approach essentially solves a one-shot optimization problem with a transformed objective in each individual time epoch, while ensuring the time-averaged convergence violation is upper-bounded and vanishes as time goes to infinity. In the second subproblem, we determine the fractional blockchain node placement for the current time epoch based on the number of training iterations at the same epoch by applying the null-space method [24] to a quadratic program through a corresponding linear system that combines the original constraint and the optimality conditions. For all the fractional solutions, we design a randomized rounding algorithm to round them into integers. To produce the estimated inputs for these subproblems, we also design a lightweight input estimation approach which explicitly tolerates inaccurate estimations.

Via a rigorous proof, next, we demonstrate the performance analysis that the regret, which measures the gap between the actual cost incurred by the online decisions of our proposed approach based on the estimated inputs and the offline optimal actual cost, only grows sublinearly along with time. This proof is non-trivial, based on connecting different problems and solutions and the design of all of our algorithms. This analysis also in turn guides the design of our input estimation approach.

TABLE I: Major notations

Input	Description
a_{ijt}	Transmission cost across WAN from edge i to edge j in t
$\boldsymbol{A}_t$	Matrix indicating across-WAN transmission cost in t
$\boldsymbol{W}$	Matrix indicating information exchanges in decentralized FL
α_t	Unit cost of encryption and decryption in t
β_t	Cost for intra-carrier user information collection in t
γ_t	Cost for operating one blockchain node in t
m	Size of data to be exchanged
G_t	Gradient (i.e., a column vector) involved in training in t
ε	Required model convergence for decentralized FL
$\mathcal{P}_{t,1}, \mathcal{P}_{t,2}$	Objectives of subproblems at t after decomposition
Decision	Description
x_{it}	Whether to place a blockchain node on edge i in t
y_t	Number of training iterations for decentralized FL in t

Finally, we implement our proposed algorithms with a prototype system, and conduct extensive trace-driven experiments to validate the superiority of our approach. Our prototype uses FedML [4] for decentralized federated learning, and creates a blockchain upon the distributed InterPlanetary File System (IPFS) [25]. Our algorithms are implemented in AMPL [26], invoking the IPOPT [27] optimization solver. Using real-world edge network data [28], WAN cost [6], and binary classification tasks [29], we observe the following results: (i) Our approach reduces the real-time blockchain cost by around 30% on average compared to multiple alternative approaches; (ii) WAN pricing, intra-edge-network communication, blockchain operation overhead, and the model size impact the blockchain cost of our approach to different extents; (iii) Our approach is robust to inaccurate estimations of inputs, reducing training iterations conservatively while still guaranteeing convergence; (iv) Our approach is efficient in execution time, only taking hundreds of milliseconds for a time epoch of 15 minutes.

II. Model and Formulation

A. System Settings and Models

We summarize all our major notations in TABLE I.

Edge Networks: We consider a set of edge networks, which may be owned and operated by different carriers. Each edge network is connected to end users through cellular or wireline access, and these edge networks are connected to one another via Wide Area Networks (WANs). Each edge network consists of multiple edges, where an "edge" refers to a micro data center [30] or server cluster co-located with a cellular base station or a WiFi access point. We consider that each edge has a globally unique index or ID across all the edge networks, and use $\mathcal{E}$ to denote the set of all the edges in the system. We also use $\mathcal{K}$ to denote the set of all the edge networks. We study the system over a series of time epochs $\mathcal{T} = \{1, ..., T\}$.

Decentralized Federated Learning: We consider a service provider who uses the edge resources from the carriers to train machine learning models at edge via decentralized federated learning over time. The output for the service provider is the final model at the end of the last (i.e., T-th) time epoch.

We adopt a pre-specified matrix $\boldsymbol{W} = (W_{ij}) \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ for the decentralized federated learning process. For i and j in $\mathcal{E}$, $W_{ij} > 0$ indicates that there is information exchange during the decentralized federated learning process between the edges i and j; and $W_{ij} = 0$ indicates that there is no information exchange between the edges i and j during this process. The latter case occurs when, for example, at least one of these two edges does not participate in the decentralized federated learning process. Without loss of generality, we enforce $\sum_{j'\in\mathcal{N}_{in}^i} W_{j'i} = 1$ (e.g., by normalization), where $\mathcal{N}_{in}^i$ is the set of the edges that exchange information with the edge i. As $\boldsymbol{W}$ is fixed and pre-specified by the service provider, the edges that participate in the decentralized federated learning process stay unchanged over all the time epochs under consideration.

Within each epoch, decentralized federated learning consists of multiple "iterations". We assign a globally unique index to each iteration of each epoch. That is, each iteration u at each edge i that participates in the decentralized federated learning process consists of multiple "steps" as follows:

- *Step 1:* The edge i uses its local model $\boldsymbol{w}_u^i$ from the previous iteration $u-1$ to incur the loss $f_u^i(\boldsymbol{w}_u^i)$ upon its local training data. It then computes an intermediate parameter $z_{u+\frac{1}{2}}^i = z_u^i - \gamma\nabla f_u^i(\boldsymbol{w}_u^i)$, where z_u^i is also obtained from $u-1$ with the initial value $z_1^i = \mathbf{0}$.
- *Step 2:* The edge i transmits the two-tuple information $\left(W_{ij}z_{u+\frac{1}{2}}^i,\ W_{ij}\varpi_u^i\right)$ to its each neighbor j. W_{ij} is defined as above. ϖ_u^i is another parameter prepared in the previous iteration $u-1$ for the edge i, whose initial value is $\varpi_1^i = 1$.
- *Step 3:* The edge i receives $\left(W_{j'i}z_{u+\frac{1}{2}}^{j'}, W_{j'i}\varpi_u^{j'}\right)$ from its each neighbor, and conducts the updates as follows for the next iteration $u+1$: $z_{u+1}^i = \sum_{j'\in\mathcal{N}_{in}^i} W_{j'i}z_{u+\frac{1}{2}}^{j'}$; $\varpi_{u+1}^i = \sum_{j'\in\mathcal{N}_{in}^i} W_{j'i}\varpi_u^{j'}$; and $\boldsymbol{w}_{u+1}^i = z_{u+1}^i/\varpi_{u+1}^i$.

The training data are produced and contributed by end users. Due to regulatory restrictions, the training data often have to stay within each corresponding edge network. Distributed federated learning is thus used, where only the (intermediate) models and/or parameters leave the edge network boundaries. For the decentralized federated learning process described as above, we actually know the following convergence result [3]:

Proposition 1. *For a model trained by the decentralized federated learning process, the following holds:*

$$\sum_{u=0}^{U}\sum_{i=1}^{n}||\boldsymbol{w}_{u+1}^i - \bar{z}_{u+1}||_2^2 \leq \frac{4\kappa^2C^2s^2}{\delta_{min}^2(1-s)^2}\sum_{u=0}^{U}||G_u||_F^2,$$

where U is the total number of iterations; $\boldsymbol{w}_{u+1}^i$ is the local model produced on the edge i at the end of the iteration u; $\bar{z}_{u+1}$ is the average of $\{z_{u+1}^i, \forall i\}$; G_u is the gradient in the iteration u (i.e., $[\nabla f_u^1(\boldsymbol{w}_u^1), ..., \nabla f_u^i(\boldsymbol{w}_u^i), ..., \nabla f_u^{|\mathcal{E}|}(\boldsymbol{w}_u^{|\mathcal{E}|})]$); κ is the learning rate; C, s, and δ_{min} are constants; and $||\cdot||_F$ is the Frobenius Norm.

Based on the above, if we set $\kappa = 1/U$, then we have

$$\sum_{u=0}^{U}\sum_{i=1}^{n}||\boldsymbol{w}_{u+1}^{(i)} - \bar{z}_{u+1}||_2^2 \leq \frac{4C^2s^2}{U\delta_{min}^2(1-s)^2}\{||G||_F^2\}_{avg},$$

where $\{||G||_F^2\}_{avg} = \sum_{u=0}^{U}||G_u||_F^2/U$ is the averaged gradient. Introducing a "model convergence parameter" ε to ensure the right-hand side, the number of required training iterations is

$$U \geq \mathcal{O}(\{||G||_F^2\}_{avg}/\delta_{min}^2)/\varepsilon.$$

Blockchain: We consider a blockchain in the system, which is mainly used to make the consensus for the information exchanged among the edges during the decentralized federated learning process. For the best resource utilization, we allow the dynamic selection of the edges as the "blockchain nodes".

Gradient transmission over blockchain runs as follows in each training iteration. Consider transferring the gradient from the edge i to the edge j. First, the gradient is encrypted, where a unique password is used to produce the encrypted gradient and the corresponding hash value. Here, encryption is to ensure trustful transmission. Second, the hash value, instead of the raw or the encrypted gradient, is recorded into the blockchain. Only the last block of the blockchain manages these hash values, and once the life cycle of the last block is terminated, blockchain nodes will compete for a new block which will be attached to the end of the blockchain. Third, the edge j sends the password, and the edge i verifies it by by re-generating a new hash value and comparing it to the existing hash value in the blockchain. Then the edge i sends the encrypted gradient and the edge j gets and decrypts it.

Control Decisions: We make two types of control decisions for the decentralized federated learning process and for the blockchain, respectively. We denote by $y_t \in \mathbb{Z}^+$ the number of training iterations conducted in t for decentralized federated learning. We use $x_{it} \in \{1, 0\}$ to denote whether or not the edge i is selected as a blockchain node in t. Note that an edge that participates in decentralized federated learning can also be selected as a blockchain node simultaneously.

Blockchain Computation Cost: Two pieces of information are recorded into the blockchain. The first is the hash values of the gradients to be transferred across edges. We denote by m the size of the gradient. Note that the model being trained has a fixed model structure or a fixed number of model parameters; the training process just determines the values of these model parameters, and so the size of the exchanged information (e.g., gradient) stays unchanged over time. We also denote by α_t the amount of computation resources consumed for encrypting and decrypting a single-unit information of gradients within a training iteration. Thus, the cost per training iteration is $m\alpha_t$. The second is the contribution (e.g., amount) of training data from end users, which can often be recorded in the blockchain for potential further rewards from the service provider. We denote by β_t the amount of resources consumed for collecting the user contribution from the edges. Regarding the resources consumed for Proof of Work when creating a new block, we denote by γ_t the cost for running one blockchain node, including the resources for competing blocks and the resources for achieving the consensus. The total computation cost is thus

$$y_t \cdot \{m\alpha_t + \beta_t + ||\boldsymbol{x}_t||_0\gamma_t\},$$

where $\boldsymbol{x}_t$ is the column vector of $\{x_{it}, \forall i\}$, and $||\boldsymbol{x}_t||_0$ is the norm counting the number of non-zero values in the vector.

Blockchain Communication Cost: We adopt a matrix $\boldsymbol{A}_t = (a_{ijt}) \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$, where $a_{ijt} \geq 0$ equals the transmission cost (e.g., network delay) from the edge i to the edge j at the time epoch t for transferring encrypted gradients across the WAN, and a_{ijt} equals positive infinity if either at least one of the two edges does not participate in the blockchain or there is no across-WAN transmission involved. The blockchain WAN communication cost is thus

$$\boldsymbol{x}_t^\top \boldsymbol{A}_t \boldsymbol{x}_t = \textstyle\sum_i \sum_j x_{it} x_{jt} a_{ijt}.$$

Note that, for $x_{it}x_{jt}a_{ijt}$ and $x_{jt}x_{it}a_{jit}$, the involved links over WAN could be the same. Therefore, without loss of generality, we assume that $\boldsymbol{A}_t$ is symmetric. Note that, decentralized federated learning incurs its own cost of computation and (intra- and/or inter-WAN) communication; as the participating edges for decentralized federated learning stay unchanged, we can incorporate all cost of decentralized federated learning per training iteration at t into β_t.

B. Problem Formulation, Goal, and Challenges

Control Problem $\mathbb{P}$: Having the system models above, we formulate the total cost optimization problem:

$$\begin{aligned}
\min \quad & \textstyle\sum_t \mathcal{P}_t \triangleq \sum_t \{\boldsymbol{x}_t^\top \boldsymbol{A}_t \boldsymbol{x}_t + y_t\{m\alpha_t + \beta_t + ||\boldsymbol{x}_t||_0 \gamma_t\}\} \\
s.t. \quad & \textstyle\sum_t y_t \geq \mathcal{O}(\{||G||_F^2\}_{avg}/\delta_{min}^2)/\varepsilon, && (1)\\
& \forall t, k: \mathbf{1}_k^\top \boldsymbol{x}_t \geq 1, && (2)\\
& \forall t: \boldsymbol{x}_t \in \{1,0\}^n, y_t \in \mathbb{Z}^+. && (3)
\end{aligned}$$

In the above, $\sum_t y_t$ refers to the total number of the iterations over all the time epochs; G_t refers to the gradient revealed at the end of the last iteration in the time epoch t; and $\mathbf{1}_k$ is a column vector specified by the edge network $k \in \mathcal{K}$, where the i-th element equals 1 if the edge i is within this edge network and equals 0 if not. The optimization objective is the total blockchain computation and communication cost. Constraint (1) ensures a sufficient number of training iterations for the desired convergence of the model being trained. Constraint (2) ensures at least one blockchain node in each edge network. Constraint (3) specifies the domains of the decision variables.

Control Problem $\widehat{\mathbb{P}}$ with Estimated Inputs: We highlight that the inputs, including α_t and β_t, are *posterior*. That is, these inputs are only revealed at (the end of) each epoch t after the actual execution of the decentralized federated learning. If we want to make control decisions on the fly at (the beginning of) each epoch t, as is desired, then we have to estimate these inputs and make control decisions based on such estimations, because the actual inputs remain unknown at that time point. Therefore, we can also formulate the following problem:

$$\begin{aligned}
\min \quad & \textstyle\sum_t \widehat{\mathcal{P}}_t \triangleq \sum_t \{\boldsymbol{x}_t^\top \boldsymbol{A}_t \boldsymbol{x}_t + y_t\{m\widehat{\alpha}_t + \widehat{\beta}_t + ||\boldsymbol{x}_t||_0 \gamma_t\}\} \\
s.t. \quad & \textstyle\sum_t y_t \geq \mathcal{O}(\{||G||_F^2\}_{avg}/\delta_{min}^2)/\varepsilon, \\
& \forall t, k: \mathbf{1}_k^\top \boldsymbol{x}_t \geq 1, \\
& \forall t: \boldsymbol{x}_t \in \{1,0\}^n, y_t \in \mathbb{Z}^+,
\end{aligned}$$

Algorithm 1 Controlling Blockchain with Decentralized FL

1: Initialize $\widehat{\boldsymbol{x}}_1, \widehat{y}_1$ as a feasible fractional solution;
2: **for** $t \in [1, T]$ **do**
// Obtain and implement decisions for current epoch
3: Round $\widehat{y}_t$, $\widehat{\boldsymbol{x}}_t$ to integers $\bar{y}_t$, $\bar{\boldsymbol{x}}_t$, respectively;
4: Place blockchain and run decentralized FL by $\bar{y}_t$, $\bar{\boldsymbol{x}}_t$;
// Make fractional decisions for next epoch
5: Estimate inputs for $\widehat{\mathcal{P}}_{t+1,1}$ and $\widehat{\mathcal{P}}_{t+1,2}$;
6: Solve $\widehat{y}_{t+1}$ from $\widehat{\mathcal{P}}_{t+1,1}$ via "online learning", given $\widehat{\boldsymbol{x}}_t$;
7: Solve $\widehat{\boldsymbol{x}}_{t+1}$ from $\widehat{\mathcal{P}}_{t+1,2}$, given $\widehat{y}_{t+1}$;
8: **end for**

where $\widehat{\cdot}$ means the estimation. We note that $\{G_t, \forall t\}$ are also only revealed after the actual training in t; however, as it is involved in a long-term constraint accumulated over time, we choose to treat it differently and resort to dynamically adjusting the control decisions to restrict the cumulative violation of this constraint, instead of estimating G_t on the fly.

Algorithmic Goal: Given the two problem formulations as above, we clarify the goal of our algorithm design. We use $\{\bar{x}_{it}, \forall i\}$ and $\bar{y}_t$ to represent the decisions at t produced by our online algorithms, whose aggregated representations are denoted as $\{\bar{\boldsymbol{x}}_t, \bar{y}_t, \forall t\}$. We also use $\{\boldsymbol{x}_t^*, y_t^*, \forall t\}$ to represent the offline optimal solutions (i.e., oracle) of the problem $\mathbb{P}$ assuming all the actual inputs over time are all given at once. We define the performance metric of *regret* as

$$r = \mathcal{P}(\{\bar{\boldsymbol{x}}_t, \bar{y}_t, \forall t\}) - \mathcal{P}(\{\boldsymbol{x}_t^*, y_t^*, \forall t\}).$$

We use $\mathcal{P}$ to refer to the objective function of $\mathbb{P}$ (and use $\widehat{\mathcal{P}}$ to refer to the objective function of $\widehat{\mathbb{P}}$). Our goal is to design online algorithms that can produce $\{\bar{\boldsymbol{x}}_t, \bar{y}_t\}$ *on the fly* at each time epoch t via appropriately solving the problem $\widehat{\mathbb{P}}$ with the estimated inputs $\{\widehat{\alpha}_t, \widehat{\beta}_t\}$ and can provably upper-bound the regret r sublinearly when evaluating $\{\bar{\boldsymbol{x}}_t, \bar{y}_t, \forall t\}$ in $\mathcal{P}$.

Algorithmic Challenges: It is fundamentally non-trivial to design online algorithms to achieve the aforementioned goal. First, the inputs need to be strategically estimated on the fly, such that solving $\widehat{\mathbb{P}}$ can produce online solutions that have provably bounded regret with regards to $\mathbb{P}$. Second, even with inputs known, $\widehat{\mathbb{P}}$ has the long-term constraint accumulated over time regarding y_t, making it not straightforward to assign any value to y_t as time goes to t. At t, setting y_t small could end up having to increase y_t as time goes closer to the end, which could increase the objective $\widehat{\mathcal{P}}$ and damage the minimization; setting y_t large could in contrast lead to sub-optimality in the first place. Third, $\widehat{\mathbb{P}}$ contains non-linear terms in the objective with integer domains. The problem is actually NP-hard even in the offline setting, due to minimizing $||\boldsymbol{x}_t||_0$ [31]. Requiring to solve it online can only escalate the difficulty.

III. Online Algorithm

To overcome all the aforementioned challenges, we propose Algorithm 1. Section III-A presents the overall structure and idea of our algorithm, and Sections III-B∼III-D elaborate the different parts of our algorithm mathematically.

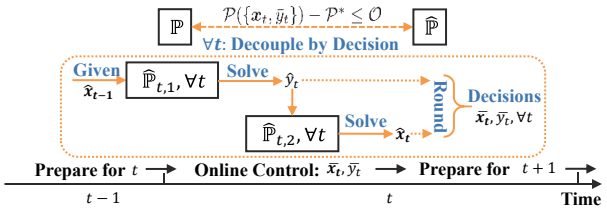


Fig. 2: Algorithm workflow

A. Algorithm Overview with Problem Decomposition

We design the following decomposition and decompose the problem $\widehat{\mathbb{P}}$ into two problems, $\widehat{\mathbb{P}}_1$ and $\widehat{\mathbb{P}}_2$, for the decentralized federated learning and for the blockchain, respectively.

$$\begin{aligned} \min \quad & \textstyle\sum_t \widehat{\mathcal{P}}_{t,1}(y_t) \triangleq \sum_t \{y_t\{m\widehat{\alpha}_t + \widehat{\beta}_t + (\mathbf{1}^\top \boldsymbol{x}_t)\gamma_t\}\} \\ s.t. \quad & \textstyle\sum_t \mathcal{C}_{t,1}(y_t) \triangleq \sum_t \{\mathcal{O}(||G||_F^2/\delta_{min}^2)/\varepsilon - y_t\} \leq 0, \\ & \forall t : y_t \in \mathbb{Z}^+. \end{aligned}$$

$$\begin{aligned} \min \quad & \textstyle\sum_t \widehat{\mathcal{P}}_{t,2}(\boldsymbol{x}_t) \triangleq \sum_t \{\boldsymbol{x}_t^\top \boldsymbol{A}_t \boldsymbol{x}_t + y_t\{m\widehat{\alpha}_t + \widehat{\beta}_t + \boldsymbol{\gamma}_t^\top \boldsymbol{x}_t\}\} \\ s.t. \quad & \forall t : \mathcal{C}_{t,2}(\boldsymbol{x}_t) \triangleq [..., -\mathbf{1}_k^\top \boldsymbol{x}_t, ...]^\top \preceq -\mathbf{1}, \\ & \boldsymbol{x}_t \in \{1,0\}^n. \end{aligned}$$

We introduce some new notations used in the above and also in the rest of this paper. We use $\widehat{\mathcal{P}}_{t,1}(y_t)$ and $\mathcal{C}_{t,1}(y_t)$ to denote the objective and the constraint functions at t for $\widehat{\mathbb{P}}_1$, and use $\widehat{\mathcal{P}}_{t,2}(\boldsymbol{x}_t)$ and $\mathcal{C}_{t,2}(\boldsymbol{x}_t)$ to denote the objective and the constraint functions at t for $\widehat{\mathbb{P}}_2$. $\widehat{\mathbb{P}}_1$ treats $\{y_t, \forall t\}$ as the decision variables and treats $\{\boldsymbol{x}_t, \forall t\}$ as the inputs. $\widehat{\mathbb{P}}_2$ treats $\{\boldsymbol{x}_t, \forall t\}$ as the decision variables and treats $\{y_t, \forall t\}$ as the inputs. In $\widehat{\mathbb{P}}_1$, we have $||\boldsymbol{x}_t||_0 = ||\boldsymbol{x}_t||_1 = \mathbf{1}^\top \boldsymbol{x}_t$ when $\boldsymbol{x}_t$ takes the integers $\{1, 0\}$. In $\widehat{\mathbb{P}}_2$, we have $\forall k$, $\mathbf{1}_k^\top \boldsymbol{x}_t \geq 1$ as in Constraint (2); then, $\mathcal{C}_{t,2}(\boldsymbol{x}_t)$ is a column vector that aggregates such constraints $\{-\mathbf{1}_1^\top \boldsymbol{x}_t, ..., -\mathbf{1}_{|\mathcal{K}|}^\top \boldsymbol{x}_t\}$ and is organized in the standard form, i.e., $\preceq -\mathbf{1}$. Also, we denote by $\widehat{\mathcal{P}}_{t,1}(y_t)$ and $\widehat{\mathcal{P}}_{t,2}(\boldsymbol{x}_t)$ the objectives with the estimated inputs; analogously, we denote by $\mathcal{P}_{t,1}(y_t)$ and $\mathcal{P}_{t,2}(\boldsymbol{x}_t)$ the objectives with the actual inputs.

Algorithm 1 proceeds as follows. At each time epoch t, we first round the fractional decisions into integers and implement such integral decisions for t, as in Lines 3∼4; next, we estimate the inputs for the next epoch $t+1$, and solve $\widehat{\mathcal{P}}_{t+1,1}$ (given $\widehat{\boldsymbol{x}}_t$) and $\widehat{\mathcal{P}}_{t+1,2}$ (given $\widehat{y}_{t+1}$), respectively, to acquire the fractional decisions of the number of training iterations and the placement of blockchain nodes for $t+1$, as in Lines 5∼7. This algorithm can be visualized in Fig. 2.

B. Controlling Decentralized Federated Learning

We focus on Line 6 of Algorithm 1 in this section.

First, note that the optimization

$$\min \textstyle\sum_t \widehat{\mathcal{P}}_{t,1}(y_t), \quad s.t. \sum_t \mathcal{C}_{t,1}(y_t) \leq 0, \quad y_t \in \mathbb{Z}^+,$$

can be equivalently written in the convex-concave form:

$$\min_{y_t \in \mathbb{Z}^+} \max_{\lambda_t \in \mathbb{R}_{\geq 0}} \textstyle\sum_t \left(\widehat{\mathcal{P}}_{t,1}(y_t) + \lambda_t \mathcal{C}_{t,1}(y_t)\right),$$

where λ_t is the Lagrange multiplier and the domain of y_t is positive integers, making it hard to solve as already mentioned previously. Thus, we relax the problem:

$$\min_{y_t \in \mathbb{R}^+} \max_{\lambda_t \in \mathbb{R}_{\geq 0}} \textstyle\sum_t \left(\widehat{\mathcal{P}}_{t,1}(y_t) + \lambda_t \mathcal{C}_{t,1}(y_t)\right).$$

Second, we split the relaxed problem to a series of subproblems over the entire time horizon and solve each of them in an online manner as time goes. That is, $\forall t+1$, we have

$$\min_{y_{t+1} \in \mathbb{R}^+} \left\{\nabla \widehat{\mathcal{P}}_{t,1}(\widehat{y}_t)(y_{t+1} - \widehat{y}_t) + \frac{||y_{t+1} - \widehat{y}_t||^2}{\xi_1} + \lambda_{t+1} \mathcal{C}_{t,1}(y_{t+1})\right\}, \quad (4)$$

where we can introduce new additional terms, i.e., the first two terms in (4), to approximate $\widehat{\mathcal{P}}_{t+1,1}(y_{t+1})$, with the parameter ξ_1 as the step size. λ_{t+1} is then updated as

$$\lambda_{t+1} = [\lambda_t + \xi_2 \mathcal{C}_{t,1}(\widehat{y}_t)]^+, \quad (5)$$

where ξ_2 is the step size, and $[\cdot]^+ \triangleq \max\{\cdot, 0\}$.

We are essentially conducting online learning here, as we do not use any information for $t+1$ but only use the information from before $t+1$ when determining the value of y_{t+1}. Specifically, at each t, we execute (5), and then solve (4) via any existing standard convex optimization solver. Note that we have broken the original constraint of $\sum_t \mathcal{C}_{t,1}(y_t) \leq 0$, which is accumulated over time, into individual constraints of $\mathcal{C}_{t,1}(y_t) \leq 0$ and incorporated each of such constraints into the objective at each corresponding t, as in (4). We will later show that this transformation, with the approximation introduced in the above, can provably upper-bound the cumulative violation regarding the original constraint. The factional control decision, i.e., $\widehat{y}_{t+1}$, will also be rounded as described later for controlling decentralized federated learning at $t+1$. Following a previous literature [23], we have Proposition 2 which upper-bounds the regret and the constraint violation for $\widehat{\mathbb{P}}_1$, respectively, when $\{\widehat{\boldsymbol{x}}_t, \forall t\}$ is given.

Proposition 2. *The following inequalities hold for $\widehat{\mathbb{P}}_1$:*

$$\begin{aligned} &\textstyle\sum_t \{\widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t) - \widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t^*)\} \leq \mathcal{O}(T^\tau), \\ &\mathcal{O}(\{||G_t||_F^2\}_{avg}/\delta_{min}^2)/\varepsilon - \textstyle\sum_t \widehat{y}_t \leq \mathcal{O}(T^{\tau'}), \end{aligned}$$

where $\widehat{y}_t^*$ is the fractional optimal solution for $\widehat{\mathbb{P}}_1$ at t given $\widehat{\boldsymbol{x}}_t$ at t; and $\tau < 1$ and $\tau' < 1$ are constants.

C. Controlling Blockchain

We focus on Line 7 of Algorithm 1 in this section.

For the optimization

$$\min \textstyle\sum_t \widehat{\mathcal{P}}_{t,2}(\boldsymbol{x}_t), \quad s.t. \ \mathcal{C}_{t,2}(\boldsymbol{x}_t) \preceq -\mathbf{1}, \ \ \boldsymbol{x}_t \in \{1,0\}^n,$$

we can transform the inequality constraint to an equality constraint via replacing its right-hand side by a constant vector. Such a constant vector $\boldsymbol{c}$ for the optimum can be enumerated through the binary lookup. Thus, we have, $\forall t+1$,

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}_{t+1}^\top (2\boldsymbol{A}_{t+1})\boldsymbol{x}_{t+1} + \widehat{y}_{t+1}\boldsymbol{\gamma}_t^\top \boldsymbol{x}_{t+1} + \widehat{y}_{t+1}(m\widehat{\alpha}_t + \widehat{\beta}_t) \\ s.t. \quad & \mathcal{C}_{t+1,2}(\boldsymbol{x}_{t+1}) = \boldsymbol{c}, \ \ \boldsymbol{x}_{t+1} \in \{1,0\}^n, \end{aligned}$$

where $\boldsymbol{c} = [c_1, ..., c_{|\mathcal{K}|}] \in \mathbb{R}_-^{|\mathcal{K}|}$ such that $\boldsymbol{c} \preceq -\mathbf{1}$. Note that $\boldsymbol{x}_{t+1}$ still falls into the integer domain. We relax the problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}_{t+1}^\top (2\boldsymbol{A}_{t+1})\boldsymbol{x}_{t+1} - \boldsymbol{x}_{t+1}^\top \boldsymbol{b} \quad (6) \\ s.t. \quad & \boldsymbol{B}\boldsymbol{x}_{t+1} = \boldsymbol{c}, \ \ \boldsymbol{x}_{t+1} \in [0,1]^n, \end{aligned}$$

where the matrix $\boldsymbol{B}^\top = [-\mathbf{1}_1, ..., -\mathbf{1}_{|\mathcal{K}|}]$ indicates related domains of the edges and each row of it corresponds to an edge network; and the column vector $\boldsymbol{b}$ is $[-\widehat{y}_{t+1}\gamma_t, ..., -\widehat{y}_{t+1}\gamma_t]$. We can actually combine the stationarity condition in the Karush-Kuhn-Tucker (KKT) optimality conditions and the above equality constraint to characterize the optimal solution. That is, we have the following linear system:

$$\boldsymbol{K}\begin{pmatrix}\widehat{\boldsymbol{x}}_{t+1}\\ \boldsymbol{\phi}^*\end{pmatrix} \triangleq \begin{pmatrix}2\boldsymbol{A}_{t+1} & B^\top\\ B & \mathbf{0}\end{pmatrix}\begin{pmatrix}\widehat{\boldsymbol{x}}_{t+1}\\ \boldsymbol{\phi}^*\end{pmatrix} = \begin{pmatrix}\boldsymbol{b}\\ \boldsymbol{c}\end{pmatrix}, \tag{7}$$

where $\widehat{\boldsymbol{x}}_{t+1}$ is the optimal solution to $\widehat{\mathbb{P}}_{t+1,2}$ and $\boldsymbol{\phi}^*$ is the Lagrange multiplier for the equality constraint.

Now, let us observe the following two propositions. Proposition 3 holds because each row of $\boldsymbol{B}$ is mapped to a unique edge network, and all these edge networks have no overlap by definition. That is, $\boldsymbol{B}$ has full row rank, as no row can be omitted. Proposition 4 follows from an existing literature [24], where $\boldsymbol{Z}$ is a matrix whose columns form a basis of the kernel of $\boldsymbol{B}$ so that $\boldsymbol{Z}$ has full column rank and $\boldsymbol{BZ} = \mathbf{0}$.

Proposition 3. *The matrix $\boldsymbol{B}$ has full row rank.*

Proposition 4. *If $\boldsymbol{Z}^\top \boldsymbol{A}_{t+1}\boldsymbol{Z}$ is positive definite, $\boldsymbol{K}$ is non-singular; further, the KKT system in (7) has a unique solution.*

Then, we adopt the null-space method [24] to solve (7) for $\widehat{\boldsymbol{x}}_{t+1}$. That is, we can write $\widehat{\boldsymbol{x}}_{t+1}$ in the form of

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{Y}w_{\boldsymbol{Y}} + \boldsymbol{Z}w_{\boldsymbol{Z}}, \tag{8}$$

where $\boldsymbol{Y}$ is a matrix such that $[\boldsymbol{Y}\ \boldsymbol{Z}]$ is nonsingular. Putting (8) into the equality constraint of (6), we have

$$\boldsymbol{B}\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{BY}w_{\boldsymbol{Y}} + \boldsymbol{BZ}w_{\boldsymbol{Z}} = \boldsymbol{BY}w_{\boldsymbol{Y}} + \mathbf{0} = \boldsymbol{c},$$

because $\boldsymbol{BZ} = \mathbf{0}$. This actually implies that $\boldsymbol{Y}w_{\boldsymbol{Y}}$ satisfies the equality constraint of (6). Then, putting (8) into the stationarity condition (7), we have

$$2\boldsymbol{A}_{t+1}\boldsymbol{Y}w_{\boldsymbol{Y}} + 2\boldsymbol{A}_{t+1}\boldsymbol{Z}w_{\boldsymbol{Z}} + \boldsymbol{B}^\top\boldsymbol{\phi}^* = \mathbf{0}.$$

Multiplying the above equation by $\boldsymbol{Z}^\top$ and using $\boldsymbol{Z}^\top\boldsymbol{B}^\top = (\boldsymbol{BZ})^\top = \mathbf{0}$, we have

$$\boldsymbol{Z}^\top(2\boldsymbol{A}_{t+1})\boldsymbol{Z}w_{\boldsymbol{Z}} = \boldsymbol{Z}^\top\mathbf{0} - \boldsymbol{Z}^\top(2\boldsymbol{A}_{t+1})\boldsymbol{Y}w_{\boldsymbol{Y}},$$

where $w_{\boldsymbol{Z}}$ can be directly solved using the Cholesky factorization upon the solved $w_{\boldsymbol{Y}}$. Finally, $\widehat{\boldsymbol{x}}_{t+1}$ for the epoch $t+1$ can be obtained from (8) accordingly.

D. Rounding Decisions and Estimating Inputs

We focus on Lines 3∼5 of Algorithm 1 in this section.

Randomized Rounding: After obtaining the fractional decisions $\widehat{\boldsymbol{x}}_t$ and $\widehat{y}_t$ for the epoch t, we need to convert them to integers. We employ randomized rounding:

$$u = \widehat{x}_{i,t} \text{ or } \widehat{y}_t,\ \ u = \begin{cases}\lfloor u \rfloor, & \textit{with prob. } \lceil u \rceil - u\\ \lceil u \rceil, & \textit{with prob. } u - \lfloor u \rfloor\end{cases}.$$

This ensures $E[\bar{y}_t] = \widehat{y}_t$ and $E[\bar{\boldsymbol{x}}_t] = \widehat{\boldsymbol{x}}_t$. Then, as shown in Line 4 of Algorithm 1, we use $\bar{\boldsymbol{x}}_t$ to place the blockchain nodes and use $\bar{y}_t$ to set the number of training iterations for decentralized federated learning in the time epoch t. After such implementations of these control decisions, the results revealed (i.e., $||G_t||_F$) will help the decision update for the next epoch (i.e., for constructing $\mathcal{C}_{t+1,1}(\cdot)$).

Input Estimation: As stated earlier, we need to obtain the estimated inputs $\{\widehat{\alpha}_t, \widehat{\beta}_t\}$ on the fly at t as our algorithm runs. Unlike previous works that estimate inputs via solving additional optimizations [6, 32], we adopt a lightweight approach here for reducing the overall algorithmic runtime overhead. In fact, in the next section, we will show a non-trivial theorem that we actually only require

$$\left|\widehat{\mathcal{P}}_{t,1}(\cdot) - \mathcal{P}_{t,1}(\cdot)\right| \le \nu \text{ and } \left|\widehat{\mathcal{P}}_{t,2}(\cdot) - \mathcal{P}_{t,2}(\cdot)\right| \le \nu,$$

where ν is a constant, in order to upper-bound the regret incurred by our algorithm. Note that this constant ν can always exist. For example, if for any t the above differences are $\nu_{t,1}$ and $\nu_{t,2}$, then we can set $\nu = \max_t\big\{\max\{\nu_{t,1}, \nu_{t,2}\}\big\}$. In practice, we may want to make ν as small as possible. In our realization, we choose to calculate the estimations at t as $\widehat{\alpha}_t = \frac{1}{t-1}\sum_{\tau=1}^{t-1}\alpha_\tau$ and $\widehat{\beta}_t = \frac{1}{t-1}\sum_{\tau=1}^{t-1}\beta_\tau$.

IV. Performance Analysis

We upper-bound the regret sublinearly based on the design of our algorithms. We introduce and recap some notations, based on which we present Theorem 1 and derive its proof.

(i) The fractional decisions produced by our online approach are $\{\widehat{\boldsymbol{x}}_t, \widehat{y}_t, \forall t\}$; the integer decisions produced by our online approach are $\{\bar{\boldsymbol{x}}_t, \bar{y}_t, \forall t\}$.

(ii) For problems with estimated inputs, given $\widehat{\boldsymbol{x}}_t$ at t, the fractional optimal solution for $\widehat{\mathbb{P}}_1$ at t is $\widehat{y}_t^*$; given $\widehat{\boldsymbol{y}}_t$ at t, the fractional optimal solution for $\widehat{\mathbb{P}}_2$ at t is $\widehat{\boldsymbol{x}}_t$.

(iii) For problems with actual inputs, the fractional offline optimal solution for $\mathbb{P}_1$ is $\{\widetilde{\boldsymbol{x}}_t^{*\circ}, \widetilde{y}_t^{*\circ}, \forall t\}$; the fractional offline optimal solution for $\mathbb{P}_2$ is $\{\widetilde{\boldsymbol{x}}_t^{*\diamond}, \widetilde{\boldsymbol{y}}_t^{*\diamond}, \forall t\}$.

(iv) For problems with actual inputs, the fractional offline optimal solution for $\mathbb{P}$ is $\{\widetilde{\boldsymbol{x}}_t^*, \widetilde{y}_t^*, \forall t\}$; the integer offline optimal solution for $\mathbb{P}$ is $\{\boldsymbol{x}_t^*, y_t^*, \forall t\}$.

Theorem 1. *The regret is upper-bounded as*

$$r = E[\mathcal{P}(\{\bar{\boldsymbol{x}}_t, \bar{y}_t, \forall t\})] - \mathcal{P}(\{\boldsymbol{x}_t^*, y_t^*, \forall t\}) \le \mathcal{O}(T^\tau) + \Omega,$$

where $\tau < 1$ and Ω are constants.

Proof. First, we note the following set of inequalities:

$$\textstyle\sum_t\{\mathcal{P}_{t,1}(\widetilde{\boldsymbol{x}}_t^*, \widetilde{y}_t^*) - \mathcal{P}_{t,1}(\boldsymbol{x}_t^*, y_t^*)\} \le 0, \tag{9}$$
$$\textstyle\sum_t\{\mathcal{P}_{t,1}(\widetilde{\boldsymbol{x}}_t^{*\circ}, \widetilde{y}_t^{*\circ}) - \mathcal{P}_{t,1}(\widetilde{\boldsymbol{x}}_t^*, \widetilde{y}_t^*)\} \le 0, \tag{10}$$
$$\textstyle\sum_t\{\mathcal{P}_{t,1}(\widehat{\boldsymbol{x}}_t, \widetilde{y}_t^{*\circ}) - \mathcal{P}_{t,1}(\widetilde{\boldsymbol{x}}_t^{*\circ}, \widetilde{y}_t^{*\circ})\} = \Omega_1', \tag{11}$$
$$\textstyle\sum_t\{\widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widetilde{y}_t^{*\circ}) - \mathcal{P}_{t,1}(\widehat{\boldsymbol{x}}_t, \widetilde{y}_t^{*\circ})\} \le \nu, \tag{12}$$
$$\textstyle\sum_t\{\widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t^*) - \widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widetilde{y}_t^{*\circ})\} \le 0, \tag{13}$$
$$\textstyle\sum_t\{\widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t) - \widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t^*)\} \le \mathcal{O}(T^\tau). \tag{14}$$

(9) follows from (iv); (10) follows from (iii); (11) defines the constant Ω_1'; (12) follows from the input estimation, i.e., Line 5 of Algorithm 1; (13) follows from (ii); and (14) follows from Line 6 of Algorithm 1 with Proposition 2, where $\tau < 1$.

Second, summing up (9)∼(14) and replacing $\widehat{\mathcal{P}}_{t,1}(\widehat{\boldsymbol{x}}_t, \widehat{y}_t)$ by $E[\widehat{\mathcal{P}}_{t,1}(\bar{\boldsymbol{x}}_t, \bar{y}_t)]$, $\forall t$ in (14), we have

$$\sum_t \{E[\widehat{\mathcal{P}}_{t,1}(\bar{\boldsymbol{x}}_t, \bar{y}_t)] - \mathcal{P}_{t,1}(\boldsymbol{x}_t^*, y_t^*)\} \leq \Omega_1 + \mathcal{O}(T^\tau), \quad (15)$$

where $\Omega_1 = \Omega_1' + \nu$. Here, note that the first term on the left-hand side of (9) cancels the second term on the left-hand side of (10); the first term on the left-hand side of (10) cancels the second term on the left-hand side of (11); ... and so on. The replacement mentioned above and the adoption of the expectation is due to randomized rounding, following from Line 3 of Algorithm 1.

Third, analogously, following steps similar to the above, we can actually also get the following:

$$\sum_t \{\widehat{\mathcal{P}}_{t,2}(\bar{\boldsymbol{x}}_t, \bar{y}_t) - \mathcal{P}_{t,2}(\boldsymbol{x}_t^*, y_t^*)\} \leq \Omega_2. \quad (16)$$

Different from (15), when deriving (16), we note that in the inequality corresponding to (14) regarding $\widehat{\mathcal{P}}_{t,2}$, the left-hand side is actually ≤ 0, following from Line 7 of Algorithm 1.

Finally, summing up (15) and (16), we complete the proof, where $\Omega = \Omega_1 + \Omega_2$. □

V. Experimental Evaluations

A. Evaluation Settings

System Implementation: We implement our control algorithms as part of a prototype system, as in Fig. 3. We adopt FedML [4] for decentralized federated learning, and create a blockchain on top of the distributed InterPlanetary File System (IPFS) [25]. Data Sender and Data Receiver communicate with each other as described in Section II-A. Hash is done via AES256-CBC. Proof-of-Work is via calculating a specific target hash value (e.g., with the prefix "0000"). We mainly use Python for our prototype implementation. For the proposed algorithms, we implement them by AMPL [26], invoking the IPOPT [27] optimization solver. To realize large-scale experiments, we conduct emulations on our lab testbed consisting of three servers, and use threads to emulate edges. Each thread runs either the FedML codes, or both the FedML and the blockchain codes. Inter-edge communication is therefore implemented as inter-thread communication.

Edge Networks and Training Data: We get real-world ISP edge networks data [28], containing four ISP edge networks of 49, 18, 16 and 4 edges, respectively. 20 out of these edges are on the borders of the edge networks, connecting to other edge networks via WANs. WAN transmission pricing is from TraceRoute [6]. We use the Occupancy Detection [29] dataset, which contains 20,000 data samples for binary classification of room occupancy from temperature, humidity, light, etc. We train a logistic regression model on 10 edges via decentralized federated learning. For inputs such as resource consumption for encryption and decryption and for competing for the new block, we measure the real-time CPU and memory usage and conduct estimation on the fly based on historical measurements. We consider 100 time epochs, each of 15 minutes.

Algorithms: We compare our approach with the following alternatives, each of which selects edges to serve as blockchain nodes and ensures one blockchain node in each edge network:

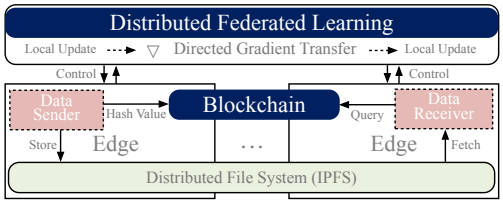
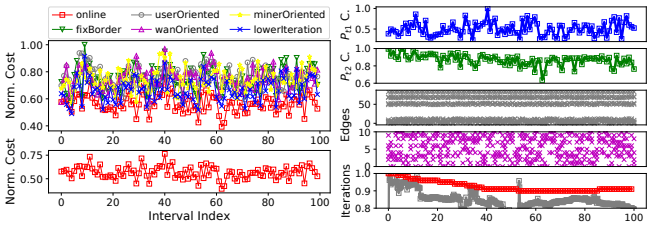


Fig. 3: Implementation architecture



(a) Real-time cost (b) More algorithmic details

Fig. 4: Cost with algorithmic details

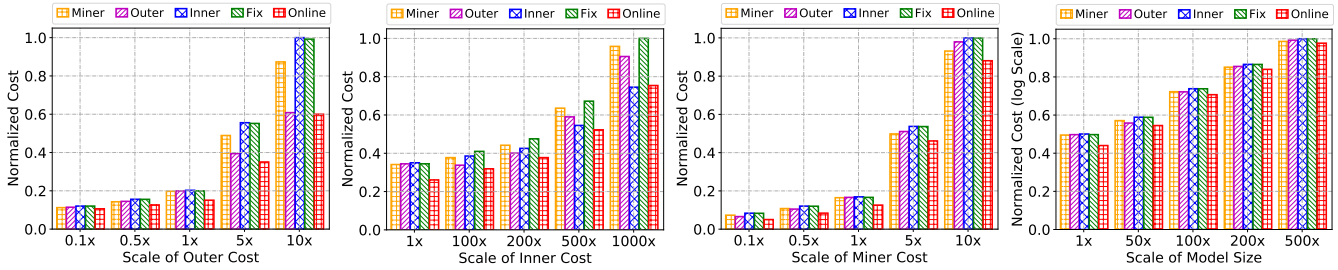
- *wanOriented (or "outer")* selects edges with the minimum across-WAN communication cost;
- *userOriented (or "inner")* selects edges with the minimum intra-edge-network communication cost;
- *minerOriented (or "miner")* selects edges with the minimum computation cost of encryption and decryption and running blockchain nodes;
- *fixBorder (or "fix")* selects edges randomly in each edge network;
- *lowerIteration (or "lower")* conducts training with 80% iterations involved upon the edges selected as in "*fix*".

All algorithms execute in an online manner using the same estimated inputs. Also, all of them take the number of the training iterations determined by our approach in each epoch.

B. Evaluation Results

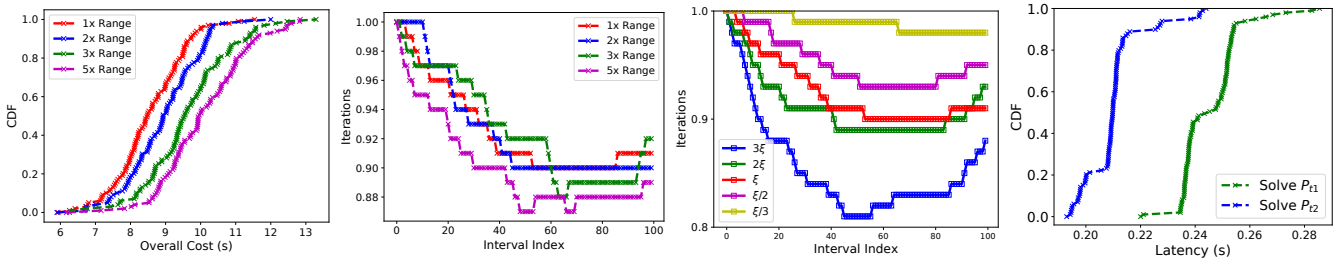
Blockchain Cost: Fig. 4(a) compares the real-time cost of all approaches. Our approach decreases the average cost per time epoch by at least 31%, compared to others. The peak cost of our approach is only 75% of the maximum of all algorithms, shown in the bottom sub-figure of Fig. 4(a). In Fig. 4(b), the first two sub-figures show the cost incurred by our two decoupled subproblems. The third and the fourth sub-figures exhibit the dynamic selections of the edges to serve as blockchain nodes, where the selection for the first 10 edges in the same edge network is displayed in detail in the latter. The fifth sub-figure shows the number of training iterations (in red) and the norm of the gradients (in grey) per time epoch.

Impact of Different Factors: Fig. 5(a) illustrates the impact of WAN pricing on the total cost. As WAN price increases, the gap between our approach and "outer" becomes smaller, because the WAN cost gradually becomes the dominant cost. The cost reduction of our approach over others is 5%∼46%. Fig. 5(b) shows the impact of the intra-ISP communication overhead, including collecting user contributions information, on the total cost. With the increase of the number of the end users, the gap between our approach and "inner" becomes



(a) Impact of WAN pricing (b) Impact of intra-ISP comm. (c) Impact of blockchain (d) Impact of model size

Fig. 5: Impact of different factors on total cost



(a) Impact on total cost (b) Impact on training iterations (c) Impact of ξ (d) Execution time

Fig. 6: More evaluation results

smaller due to similar reasons as stated for Fig. 5(a). The cost reduction of our approach over others is at least 15%. Fig. 5(c) depicts the impact of the blockchain operations (i.e., encryption/decryption and running blockchain nodes) overhead on the total cost. With the increase of the blockchain operations overhead, the gap between our approach and "miner" becomes smaller. The cost reduction of our approach over others is 11%∼32%. Fig. 5(d) reveals the impact of the model size on the total cost. Such model size also impacts the size of the gradients. As the model size grows, the gap between our approach and others increases (the vertical axis of this figure is in the logarithmic scale). The cost reduction is up to 210%.

Impact of Input Estimations: In Fig. 6(a) and Fig. 6(b), "1 ×Range" refers to the range of variations of the actual inputs as in the original dataset; "2×Range" means that we manually enlarge each actual input to 2 times its original value; and so on. Estimation is always based on averaging the historical values; thus, as such range increases, the absolute gap between the actual input and the estimated input becomes larger. Fig. 6(a) shows that, as such range for increases, the total cost increases. This is because the control decisions are made based on more inaccurate estimations. Fig. 6(b) shows the impact on the number of training iterations by decisions made based on estimated inputs. Although the minimum number of the training iterations decreases as the estimations become less accurate, such minimum number of the training iterations is still 88% of the original value, ensuring model convergence.

Impact of Algorithmic Parameter: Fig. 6(c) investigates the impact of ξ on the number of training iterations. We set $\xi_1 = \xi_2 = \xi$, where ξ_1 and ξ_2 are the algorithmic parameters as in Section III-B. As in the formulation (4), as ξ increases, the number of training iterations at one time epoch does not have to follow that at the previous time epoch; thus, the number of training iterations can change vastly as time goes. When ξ is small, it is harder to change the number of training iterations as time goes. The minimum number of training iterations is 80% of its value at the beginning of the time horizon.

Execution Time: Fig. 6(d) shows the execution time, where the solving process takes only hundreds of milliseconds.

VI. Related Work

Federated Learning at Edge: Wang *et al.* [10] studied model aggregations under a given resource budget. Zhou *et al.* [11] controlled the throughout of data training for cost-efficient federated learning in edge networks. Feng *et al.* [12] optimized model compression, sample selection, and user selection in wireless mobile edges. Lu *et al.* [13] clustered clients based on training data distribution and selected participants via auctions for energy efficiency. Jin *et al.* [14] jointly managed data transference from user devices, resource provisioning at edge servers, and the federated learning process.

These works focus on federated learning at the network edge, but do not typically explore consensus mechanisms over distrustful networks such as WANs. They lack consideration of blockchains and cannot capture our work.

Blockchain at Edge: Qiu *et al.* [15] used reinforcement learning for computation offloading in blockchain-empowered edge computing. Feng *et al.* [8] optimized the performance of blockchain and mobile edge computing via the Markov decision process. Guo *et al.* [9] studied adaptive resource allocation and block generation to improve blockchain throughput. Chen *et al.* [16] presented a resource authentication approach based on blockchain group key management. Du *et al.* [17] designed edge resource markets with the Proof-of-Stake consensus for transaction verification and award allocation.

These works investigate the optimization and the application of blockchains in edge computing. They do not exploit blockchains for federated learning, and thus are inapplicable to our work tailored to decentralized federated learning.

Decentralized Federated Learning with Blockchain: Li *et al.* [18] proposed to let clients join blockchain and federated learning simultaneously and broadcast gradients for aggregation. Li *et al.* [19] employed blockchain for storage to replace central federated learning servers. Hu *et al.* [20] decentralized federated learning using a ring topology and used blockchain for security. Wilhelmi *et al.* [21] used blockchain to store model updates and studied ledger inconsistencies and age of information. Feng *et al.* [22] explored smart contracts with the blockchain to authenticate participants and aggregate models.

These works often study a different type of decentralized federated learning, leverage a specific aspect of blockchain, or loosely couple these two. None has captured the cost trade-offs with online resource optimization under uncertainties.

VII. Conclusion

Orchestrating the blockchain dynamically and continuously in a cost-minimizing manner to facilitate decentralized federated learning across distrustful networks has been less studied, and this paper fills this gap. Our proposed online optimization algorithm innovatively decomposes the problem and solves subproblems alternately on the fly only based on estimated inputs. We theoretically prove the sublinear regret of our online approach, practically implement the prototype system, and experimentally exhibit the cost benefits from multiple angles. For future work, we will continue to explore the interaction and intersection of blockchain and federated learning techniques.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation (CNS-2047719 and CNS-2225949), by the National Natural Science Foundation of China (No. 61832005, No. 62072344, No. U20A20177, No. 62232004, and No. 62172241), and by the China University Industry Research Innovation Foundation (No. 2021FNA04005). The corresponding authors are Lei Jiao (jiao@cs.uoregon.edu) and Zhuzhong Qian (qzz@nju.edu.cn).

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *MLSys*, 2019.

[3] C. He, C. Tan, H. Tang, S. Qiu, and J. Liu, "Central server free federated learning over single-sided trust social networks," *arXiv:1910.04956*, 2019.

[4] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," *arXiv:2007.13518*, 2020.

[5] C.-C. Hung, G. Ananthanarayanan, L. Golubchik, M. Yu, and M. Zhang, "Wide-area analytics with multiple resources," in *ACM EuroSys*, 2018.

[6] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *ACM SIGCOMM*, 2016.

[7] J. Zou, D. He, S. Zeadally, N. Kumar, H. Wang, and K. R. Choo, "Integrated blockchain and cloud computing systems: A systematic survey, solutions, and challenges," *ACM Computing Surveys*, vol. 54, no. 8, pp. 160:1–160:36, 2022.

[8] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2019.

[9] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1689–1703, 2019.

[10] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[11] Z. Zhou, S. Yang, L. J. Pu, and S. Yu, "Cefl: Online admission control, data scheduling and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[12] C. Feng, Z. Zhao, Y. Wang, T. Q. Quek, and M. Peng, "On the design of federated learning in the mobile edge computing systems," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 5902–5916, 2021.

[13] R. Lu, W. Zhang, Y. Wang, Q. Li, X. Zhong, H. Yang, and D. Wang, "Auction-based cluster federated learning in mobile edge computing systems," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[14] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, "Learning for learning: predictive online control of federated learning with edge provisioning," in *IEEE INFOCOM*, 2021.

[15] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050–8062, 2019.

[16] T. Chen, L. Zhang, K.-K. R. Choo, R. Zhang, and X. Meng, "Blockchain-based key management scheme in fog-enabled iot systems," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 766–10 778, 2021.

[17] Y. Du, Z. Wang, J. Li, L. Shi, D. N. K. Jayakody, Q. Chen, W. Chen, and Z. Han, "Blockchain-aided edge computing market: Smart contract and consensus mechanisms," *IEEE Transactions on Mobile Computing*, 2022.

[18] J. Li, Y. Shao, K. Wei, M. Ding, C. Ma, L. Shi, Z. Han, and H. V. Poor, "Blockchain assisted decentralized federated learning (blade-fl): Performance analysis and resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2401–2415, 2021.

[19] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[20] Y. Hu, Y. Zhou, J. Xiao, and C. Wu, "Gfl: A decentralized federated learning framework based on blockchain," *arXiv:2010.10996*, 2020.

[21] F. Wilhelmi, E. Guerra, and P. Dini, "On the decentralization of blockchain-enabled asynchronous federated learning," *arXiv:2205.10201*, 2022.

[22] C. Feng, B. Liu, K. Yu, S. K. Goudos, and S. Wan, "Blockchain-empowered decentralized horizontal federated learning for 5g-enabled uavs," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3582–3592, 2021.

[23] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.

[24] Ronald H.W. Hoppe, "Optimization Theory: Null-space Approach," https://www.math.uh.edu/∼rohop/fall_06/, 2006.

[25] Ahansaha Souvik, "Blockchain based Decentralized File System over IPFS," https://github.com/ruchi26/Blockchain-based-Decentralized-File-Sharing-System-using-IPFS, 2020.

[26] "AMPL," https://ampl.com/, 2023.

[27] "IPOPT," https://github.com/coin-or/Ipopt/, 2023.

[28] "Links from Edges to Aliyun," www.aliyun.com/, 2023.

[29] UCI Machine Learning Repository, "Occupancy Detection Data Set," https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+, 2016.

[30] D. Kumar, J. Li, A. Chandra, and R. Sitaraman, "A TTL-based approach for data aggregation in geo-distributed streaming analytics," in *ACM SIGMETRICS*, 2019.

[31] V. Jain, "Zero-norm optimization: Models and applications," Ph.D. dissertation, 2010.

[32] H. Wang, L. Wan, M. Dong, K. Ota, and X. Wang, "Assistant vehicle localization based on three collaborative base stations via sbl-based robust doa estimation," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5766–5777, 2019.