Quantum Assisted Scheduling Algorithm for Federated Learning in Distributed Networks

Xinliang Wei*, Lei Fan†, Yuanxiong Guo‡, Yanmin Gong§, Zhu Han¶, Yu Wang*
*Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA
†Department of Engineering Technology, University of Houston, Houston, TX, USA
‡Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, USA
§Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, USA
¶Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
{xinliang.wei, wangyu}@temple.edu, lfan8@central.uh.edu, {yuanxiong.guo, yanmin.gong}@utsa.edu, zhan2@uh.edu.

***Abstract*—The scheduling problem for federated learning (FL) with multiple models in a distributed network is challenging, as it involves NP-hard mixed-integer nonlinear programming. Moreover, it requires optimal participant selection and learning rate determination among multiple FL models to avoid high training costs and resource competition. To overcome those challenges, in literature the Benders' decomposition algorithm (BD) can deal with mixed integer problems, however, it still suffers from limited scalability. To address this issue, in this paper, we present the Hybrid Quantum-Classical Benders' Decomposition (HQCBD) algorithm, which combines the power of quantum and classical computing to solve the joint participant selection and learning scheduling problem in multi-model FL. HQCBD decomposes the optimization problem into a master problem with binary variables and small subproblems with continuous variables. This collaboration maximizes the potential of both quantum and classical computing, and optimizes the complex joint optimization problem. Simulation on the commercial D-Wave quantum annealing machine demonstrates the effectiveness and robustness of the proposed method, with up to 18% improvement of iterations and 81% improvement of computation time over BD algorithm on classical CPUs even at small scales.**

***Index Terms*—Federated learning, participant selection, learning scheduling, hybrid quantum-classical optimization**

I. Introduction

With the use of quantum superposition and entanglement, quantum computing (QC) has demonstrated a quantum advantage over classical computing in random quantum circuit sampling [1], Gaussian boson sampling [2], and combinatorial optimization [3]–[5]. In this paper, by leveraging the parallel computing capability of quantum computing, we focus on designing a new quantum-assisted scheduling algorithm to solve a complex joint participant selection and learning scheduling problem for federated learning (FL) in distributed networks.

Federated learning is emerging as an effective and privacy-preserving machine learning (ML) paradigm [6]–[9], which leverages both the computing capabilities and local datasets available at the distributed clients to collaboratively train an ML model and exchange model parameters periodically among the parameter server (PS) and FL clients (or workers). FL can not only prevent the leakage of personal privacy, but also make full use of massive computing resources on distributed clients. However, there are two obstacles when deploying the FL framework in distributed networks. First, the computing capability and network resources of servers and their data distribution are heterogeneous. Some low-performance servers may decelerate the convergence process and diminish the training performance. Also, the dispersed computing resources and large network latency may lead to high training costs. Second, for the practical scenario, training multiple different models in the shared distributed network simultaneously leads to competition for computing and communication resources. As shown in Fig. 1, two FL models are trained concurrently and each FL model requires one PS and three workers for model training. In this case, which FL model is preferentially served at which server directly affects the total training cost of all FL models. To this end, appropriate participant selection and learning schedules are fairly crucial for multi-model FL training.

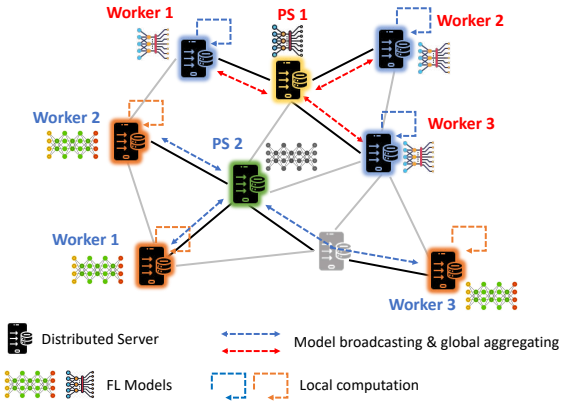


Fig. 1: The training process of multi-model federated learning.

Therefore, we mainly concentrate on the joint participant selection and learning scheduling problem in multi-model FL training scenarios. It should be emphasized that each server in distributed networks can serve as a PS or client and the participant selection includes the selection of both the PS and clients

The work is partially supported by the US National Science Foundation (under Grant No. CCF-1908843, CNS-2006604, CNS-2107216, CNS-2128368, CNS-2047761, CNS-2106761, CMMI-2222670, CMMI-2222810, and EPCN-2045978), the US Department of Transportation, Toyota, and Amazon.

979-8-3503-3618-4/23/$31.00 ©2023 IEEE

for each FL model. We denote a client as an FL worker in our work for simplicity. It is worth noting that both participant (client) selection and learning scheduling problems have been studied in FL using classical computers recently. For instance, Nishio and Yonetani [10] studied a client selection problem in the decentralized FL where a set of mobile clients are chosen to act as workers for FL and their aim is to maximize the number of selected clients under time constraints. Zhu *et al.* [11] proposed an asynchronous FL framework with adaptive client selection to minimize the total training latency by leveraging client availability and long-term fairness. Li *et al.* [12] also considered client scheduling in FL to overcome client uncertainties or stragglers via learning-based task replication. Wang *et al.* [13] focused on FL training convergence and adaptive control in edge computing without client selection. They proposed a control algorithm to determine the trade-off between local update and global parameter aggregation so as to minimize the loss function. Jin *et al.* [14] studied the joint control of local learning rate and edge provisioning in FL to minimize the long-term cumulative cost. However, those works concentrate on optimizing a single global FL model rather than multiple FL models. More importantly, none of these works take into account the PS selection for multiple FL models. Recently, Wei *et al.* [15] considered a joint participant selection and learning scheduling problem in multi-model federated edge learning, and proposed multi-stage methods to solve the joint optimization problem. However, due to the nature of the formulated optimization as a mixed-integer non-linear program (MINLP), the proposed methods may not lead to optimal solutions and do not scale well when the problem grows more complex.

To overcome the above problem, quantum computing has recently gained prominence as a powerful tool for optimization [3]–[5]. Such approaches, however, may not be competitive until the shortcomings of QC, such as the limited number of qubits, are overcome by further technological advancements. To that end, several hybrid quantum-classical solutions [16], [17] have been proposed to tackle optimization problems by leveraging the complementary strengths of quantum and classical computers. For example, Ajagekar *et al.* [17] proposed a hybrid solution strategy for optimization problems that uses quantum annealing (QA), but it may result in longer computational times with no guarantee of feasibility for large-scale scheduling problems due to the inefficient use of quantum solution techniques. Subsequently, some researchers in [18]–[20] presented the novel hybrid quantum-classical optimization technique through the decomposition of the problem into smaller tractable master problems and subproblems. Inspired by the pioneers, we attempt to solve our joint participant selection and learning scheduling problem by the hybrid quantum-classical optimization approach combined with decomposition techniques. Such an approach enables us to fully utilize the capabilities of both quantum and classical computers. In addition, in the quantum computing market, D-wave stands out because it offers the quantum annealer computer with the most qubits of all the candidates. With D-wave's quantum annealer computer, one can solve an integer linear programming (ILP) problem by converting it into a quadratic unconstrained binary optimization (QUBO) model, which is inspired by the Ising model. As a result, we attempt to develop novel hybrid quantum-classical algorithms on the D-Wave's quantum computer.

Two research challenges exist in developing efficient hybrid quantum-classical techniques with decomposition schemes. ***First****, how to convert our original MINLP problem into an ILP problem that can be recognized by the quantum computer?* ***Second****, how to further convert the reformulated ILP problem into a QUBO model as the input to the D-Wave's quantum computer?* To handle the above challenges, we develop a novel hybrid quantum-classical algorithm to demonstrate the potential of such hybrid approaches. Specifically, we leverage the linearization and Benders' decomposition (BD) technique which is widely employed for solving mixed integer linear programming (MILP) problems to convert our MINLP problem into the ILP master problem and linear programming (LP) subproblems and then present a Hybrid Quantum-Classical Benders' Decomposition (HQCBD) algorithm. The master problem will be solved by the quantum computer while subproblems will be solved by distributed classical computers. The contributions of this paper are summarised as follows.

- We first formulate a joint participant selection (both PS and workers) and learning scheduling problem for multi-model FL in a distributed network as an MINLP, with the objective to minimize the total learning costs of all FL models. (Section II)
- We then propose a novel HQCBD algorithm to tackle the joint optimization problem. By leveraging the combination of quantum computing and classical optimization technique, our HQCBD algorithm can quickly converge to the desired solution as the classical BD does but with much fewer iterations and faster speeds. (Section III)
- We conduct extensive simulations with real FL tasks as well as the commercial D-Wave quantum computer to evaluate our proposed algorithms. Numerous experiments have demonstrated that our proposed HQCBD can achieve significant advancement (up to 18% saving of iterations and 81% reduction of computation time) compared to the BD algorithm on classical CPUs even at small scales. (Section IV)

II. System Model and Problem Formulation

A. System Model

The distributed network connecting all computing servers is modeled as a graph $G(V, E)$, where $V = \{v_1, \cdots, v_N\}$ and $E = \{e_1, \cdots, e_L\}$ are the sets of N servers and L direct connection links, respectively. Generally, each server v_i owns a specific storage capacity c_i and CPU frequency f_i while each link e_j has an available bandwidth b_j. Each server holds a distinct set of datasets and can be used for local model training. We assume that each server can hold multiple types of datasets for FL training and the dataset used by the j-th FL model in the i-th server is denoted by $D_{i,j}$. In this paper, we focus on

the participant selection based on computing/communication resources in the distributed network, and do not consider training data distributions (which is another important research topic and orthogonal to our research).

B. Federated Learning Model

We assume that parallel FL was conducted where multiple models are trained concurrently in the network. We consider a classical FL process that consists of a PS and multiple workers. W FL models ($M = \{m_1, \cdots, m_W\}$) are trained concurrently and each FL model will request certain requirements for the training task, i.e., (1) $\kappa_j + 1$ servers as participants including one PS and κ_j workers, whose CPU and storage capacity should be larger than its required minimal CPU frequency χ_j and model size μ_j, respectively; and (2) the achieved global convergence rate needs to be larger than ς_j. We further assume that each server can only play a role as either the PS or the worker for any FL model at one time. The training process of each FL model includes three stages: (a) initializing and broadcasting the global model of m_j to each participant; (b) each worker performs the local model computation using its own dataset; and (c) aggregating the local models from workers, as illustrated in Fig. 1 and detailed in the sequel.

***Stage 1**: Global Model Initialization.* In Stage 1, we initialize the global model parameter for each FL model as ω_j and send the global model parameter to each selected participant.

***Stage 2**: Local Model Computation.* Let the local parameters of model m_j on server v_i be $\omega_{i,j}$ and the loss function on a training data sample s be $f_{i,j}(\omega_{i,j}, dx_s, dy_s)$, where dx_s is the input feature and dy_s is the required label. Then the loss function on the whole local dataset of v_i is defined as

$$F_{i,j}(\omega_{i,j}) = \frac{1}{|D_{i,j}|} \sum_{s \in D_{i,j}} f_{i,j}(\omega_{i,j}, dx_s, dy_s).$$

Generally, FL will perform round by round and we denote the total number of global aggregations, and local updates as $\hat{\alpha}$ and $\hat{\beta}$, where α and β are their indexes, respectively. In the α-th round, each worker runs a number of local updates to achieve a *local convergence accuracy* $\varrho_j \in (0, 1)$. At the β-th local iteration, each worker follows the local update rule as

$$\omega_{i,j}^{\alpha,\beta} = \omega_{i,j}^{\alpha,\beta-1} - \eta \nabla F_{i,j}(\omega_{i,j}^{\alpha,\beta-1}),$$

where η is the learning rate of the loss function. This process will run until

$$F_{i,j}(\omega_{i,j}^{\alpha,\hat{\beta}}) - F_{i,j}^* \leq \varrho_j [F_{i,j}(\omega_{i,j}^{\alpha,0}) - F_{i,j}^*]. \tag{1}$$

Here, we set $\omega_{i,j}^{\alpha,0} = \omega_j$.

***Stage 3**: Global Aggregation.* At this stage, one participant has to be chosen as the PS. After $\hat{\beta}$ local updates, all workers send their local model parameter $\omega_{i,j}^{\alpha,\hat{\beta}}$ to the PS. The PS performs FedAvg to aggregate global model parameters as

$$\omega_j^{\alpha} = \sum_{i \in S_j} \frac{D_{i,j}}{D_j} \omega_{i,j}^{\alpha-1,\hat{\beta}},$$

where $D_j = \bigcup_{i \in S_j} D_{i,j}$ is the total number of data sample from κ_j workers and S_j is the set of selected workers. The global convergence of the global model is defined as

$$\mathcal{G}_j(\omega_j^{\hat{\alpha}}) - \mathcal{G}_j^* \leq \varsigma_j [\mathcal{G}_j(\omega_j^0) - \mathcal{G}_j^*], \tag{2}$$

where $\mathcal{G}_j^*$ is the global optimum of FL model m_j.

Finally, from (1) and (2), in order to achieve the desired local convergence rate ϱ_j and global convergence rate ς_j, we need to calculate the number of local updates $\hat{\beta} = \varphi_j$ and the number of global rounds $\hat{\alpha} = \vartheta_j$. From the above observation, we can find that the global convergence rate ς_j for each FL model can be predefined and we have to conduct enough local updates and global aggregations to achieve that. Then we have the following relationship between the convergence rate and the local update as well as global iterations [14], [21]–[25].

$$\vartheta_j \geq \frac{2\lambda^2}{\gamma^2 \xi} ln(\frac{1}{\varsigma_j}) \frac{1}{1 - \varrho_j} \triangleq \vartheta_0 \ln(\frac{1}{\varsigma_j}) \frac{1}{1 - \varrho_j}, \tag{3}$$

$$\varphi_j \geq \frac{2}{(2 - \lambda\delta)\delta\gamma} log_2(\frac{1}{\varrho_j}) \triangleq \varphi_0 log_2(\frac{1}{\varrho_j}), \tag{4}$$

where ξ and δ are two variables in ranges (0, $\frac{\gamma}{\lambda}$] and (0, $\frac{2}{L}$), respectively. λ is the λ-Lipschitz parameter and γ is the γ-strongly convex parameter. Both the values of λ and γ are determined by the loss function. ϑ_0 and φ_0 are two constants where $\vartheta_0 = \frac{2\lambda^2}{\gamma^2\xi}$ and $\varphi_0 = \frac{2}{(2-\lambda\delta)\delta\gamma}$.

C. Cost Model

Our cost model consists of four parts: *transmission cost, local update cost, global aggregation cost, and participant cost*, defined as follows.

Transmission Cost: The transmission cost consists of the FL model downloading and uploading costs. Denote μ_j by the model size of FL model m_j. We leverage the shortest path in the distributed network to calculate the transmission cost when downloading models from the PS or uploading models to the PS. Let $\rho_j(v_i, v_k)$ be the transmission cost of model m_j from server v_i to v_k, and it can be calculated by $\rho_j(v_i, v_k) = \sum_{e_l \in P_{i,k}} \frac{\mu_j}{b_l}$, where $P_{i,k}$ is the shortest path connecting v_i to v_k. For model m_j, the total transmission cost is

$$C_j^{trans} = 2 \cdot \vartheta_j \sum_{k=1}^{N} \sum_{i=1}^{N} x_{k,j} \cdot y_{i,j} \cdot \rho_j(v_i, v_k).$$

Here, v_i and v_k are a worker and the PS of m_j, respectively. In addition, $x_{i,j}$ or $y_{i,j}$ are the decision variables indicating whether to select server v_i as a parameter server or an FL worker for the j-th FL model.

Local Update Cost: Let $\psi(\cdot)$ be the function to define CPU cycles to process the sample data $D_{j,i}$ used by the jth FL model and stored in server v_i. So the local update cost for the j-th FL model is defined as

$$C_j^{local} = \vartheta_j \cdot \varphi_j \cdot \sum_{i=1}^{N} y_{i,j} \cdot \frac{\psi(D_{j,i})}{f_i}.$$

Global Aggregation Cost: Similarly, we use $\psi(\cdot)$ function to define the number of CPU cycles to process the aggregation step for the uploaded FL models.

$$C_j^{global} = \vartheta_j \cdot \sum_{i=1}^{N} x_{i,j} \cdot \frac{\psi(\mu_j)}{f_i}.$$

Participant Cost: Each participant of FL model m_j will be paid a basic rental cost for utility management which is

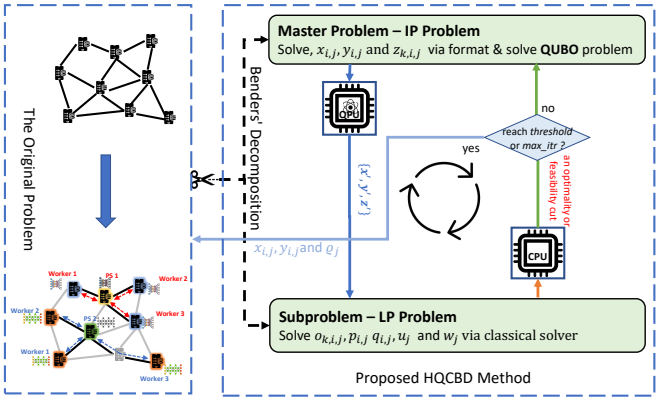


Fig. 2: The proposed HQCBD framework.

related to their CPU frequency. Let p_j be the unit price for a CPU unit, accordingly, the participant cost for the j-th FL model is defined as

$$C_j^{rent} = \sum_{i=1}^{N}(x_{i,j} + y_{i,j}) \cdot p_j \cdot f_i.$$

D. Problem Formulation

Under the previously introduced multi-model federated learning scenario, we consider how to choose participants for each of the models and how to schedule their local/global updates. Recall that we assume that only one PS and κ_j workers selected for one model, i.e., $\sum_{i=1}^{M} x_{i,j} = 1$ and $\sum_{i=1}^{M} y_{i,j} = \kappa_j$. We use $\varrho_j \in [0.01, 0.99]$ to represent the maximal local convergence rate of m_j. We will use ϱ_j and ς_j to control the number of local updates and global iterations for model m_j. Note that ς_j is given by the model m_j as a requirement. Overall, $x_{i,j}$, $y_{i,j}$ and ϱ_j are the decision variables of our optimization.

We now formulate our participant selection and learning scheduling problem for FL in the distributed network where we need to select the parameter server and workers for each model as well as achieve the desired local convergence rate. The objective of our problem is to minimize the total learning costs of all FL models as follows.

$$\min_{x,y,\rho} \quad \sum_{j=1}^{W}(C_j^{trans} + C_j^{local} + C_j^{global} + C_j^{rent}) \tag{5}$$

$$\text{s.t.} \quad x_{i,j}\mu_j\kappa_j \le c_i, \quad x_{i,j}\chi_j \le f_i, \qquad \forall i,j, \tag{6}$$

$$y_{i,j}\mu_j \le c_i, \quad y_{i,j}\chi_j \le f_i, \qquad \forall i,j, \tag{7}$$

$$\sum_{i=1}^{N} x_{i,j} = 1, \quad \sum_{i=1}^{N} y_{i,j} = \kappa_j, \qquad \forall j, \tag{8}$$

$$\sum_{j=1}^{W}(x_{i,j} + y_{i,j}) \le 1, \qquad \forall i, \tag{9}$$

$$i \in (1, \ldots, N), j \in (1, \ldots, W), \tag{10}$$

$$x_{i,j}, y_{i,j} \in \{0, 1\}, \varrho_j \in [0.01, 0.99]. \tag{11}$$

Constraints (6) and (7) make sure that the storage and CPU satisfy the requirements from the FL model. Constraint (8) guarantees the number of the parameter server and FL workers of each model is 1 and κ_j, respectively. Constraint (9) ensures that each server only trains one FL model and can only play one role at one time. The decision variables and their ranges are given in (10) and (11). Note that the formulated problem (5) is a non-linear mixed-integer program, which is NP-hard in general and challenging to solve with classical computing.

III. Hybrid Quantum Assisted Benders' Decomposition (HQCBD) Method

Motivated by the advances in QC, we decouple the original problem into a master problem and a subproblem by leveraging Benders' decomposition [18], [20] and solving them using quantum and classical methods, respectively. Fig. 2 shows the framework of our proposed HQCBD.

A. Benders' Decomposition

We first briefly introduce the basic idea of Benders' decomposition. Benders' decomposition is a useful algorithm for solving convex optimization problems with a large number of variables. It works best when a large problem can be decomposed into two (or more) smaller problems that are individually much easier to solve [18]. At a high level, the procedure will iteratively solve the master problem and subproblem. Each iteration provides an updated upper and lower bound on the optimal objective value. The result of the subproblem either provides a new constraint to add to the master problem or a certificate that no finite optimal solution exists for the problem. The procedure terminates when no finite optimal solution exists or when the gap between the upper and lower bound is sufficiently small [26].

We reformulate our original problem (5) by extracting all constant variables as below.

$$\begin{aligned}\min_{x,y,\rho} \quad & \sum_{j=1}^{W}[\frac{1}{1-\varrho_j}\sum_{k=1}^{N}\sum_{i=1}^{N} a_{1,i,j,k} \cdot x_{k,j} \cdot y_{i,j} \\ & + \frac{1}{1-\varrho_j} \cdot log_2(\frac{1}{\varrho_j}) \cdot \sum_{i=1}^{N} a_{2,i,j} \cdot y_{i,j} \\ & + \frac{1}{1-\varrho_j} \cdot \sum_{i=1}^{N} a_{3,i,j} \cdot x_{i,j} + \sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j})]\end{aligned} \tag{12}$$

$$\text{s.t.} \quad (6) - (11),$$

where the four sets of constant variables are $a_{1,i,j,k} = 2\vartheta_0 ln(\frac{1}{\varsigma_j}) \cdot \rho_j(v_i, v_k)$, $a_{2,i,j} = \varphi_0\vartheta_0 ln(\frac{1}{\varsigma_j}) \cdot \frac{\psi(D_{j,i})}{f_i}$, $a_{3,i,j} = \vartheta_0 ln(\frac{1}{\varsigma_j}) \cdot \frac{\psi(\mu_j)}{f_i}$, and $a_{4,i} = \delta f_i$.

Due to the non-linear term $x_{k,j}y_{i,j}$, it is challenging to solve the problem directly. Thus, we introduce an additional binary variable $z_{k,i,j}$ to linearize the product term and additional continuous variables u_j and w_j to replace ϱ_j.

$$\min_{x,y,z,u,w} \quad \sum_{j=1}^{W}[u_j \cdot \sum_{k=1}^{N}\sum_{i=1}^{N} a_{1,i,j,k} \cdot z_{k,i,j} + w_j \cdot \sum_{i=1}^{N} a_{2,i,j} \cdot y_{i,j} + u_j \cdot \sum_{i=1}^{N} a_{3,i,j} \cdot x_{i,j} + \sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j})] \tag{13}$$

$$\text{s.t.} \quad (6) - (11),$$

$$z_{k,i,j} \leq y_{i,j}, \quad z_{k,i,j} \leq x_{k,j}, \tag{14}$$

$$z_{k,i,j} \geq x_{k,j} + y_{i,j} - 1, \tag{15}$$

$$b_1 \leq u_j \leq b_2, \quad b_3 \leq w_j \leq b_4, \tag{16}$$

where $u_j = \frac{1}{1-\varrho_j}$, $w_j = u_j log_2(\frac{u_j}{u_j - 1})$, $b_1 = 1.01$, $b_2 = 100$, $b_3 = 1.435$ and $b_4 = 6.725$.

Note that Problem (13) consists of several terms that are the products of integer and continuous variables, e.g. $u_j \cdot z_{k,i,j}, w_j \cdot y_{i,j}$. Hence, we further introduce variables $o_{k,i,j}$, $p_{i,j}$ and $q_{i,j}$ to represent the product of an integer variable and a continuous variable as below.

$$\min_{x,y,z,u,w,o,p,q} \sum_{j=1}^{W}[\sum_{k=1}^{N}\sum_{i=1}^{N} a_{1,i,j,k} \cdot o_{k,i,j} + \sum_{i=1}^{N} a_{2,i,j} \cdot p_{i,j} + \sum_{i=1}^{N} a_{3,i,j} \cdot q_{i,j} + \sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j})] \tag{17}$$

$$\text{s.t.} \quad (6) - (11), (14) - (16),$$

$$b_1 z_{k,i,j} \leq o_{k,i,j} \leq b_2 z_{k,i,j}, \tag{18}$$

$$u_j - o_{k,i,j} \leq b_2(1 - z_{k,i,j}), \tag{19}$$

$$u_j - o_{k,i,j} \geq b_1(1 - z_{k,i,j}), \tag{20}$$

$$b_3 y_{i,j} \leq p_{i,j} \leq b_4 y_{i,j}, \tag{21}$$

$$w_j - p_{i,j} \leq b_4(1 - y_{i,j}), \tag{22}$$

$$w_j - p_{i,j} \geq b_3(1 - y_{i,j}), \tag{23}$$

$$b_1 x_{i,j} \leq q_{i,j} \leq b_2 x_{i,j}, \tag{24}$$

$$u_j - q_{i,j} \leq b_4(1 - x_{i,j}), \tag{25}$$

$$u_j - q_{i,j} \geq b_3(1 - x_{i,j}). \tag{26}$$

So far, we have linearized the products of binary variables (x, y, and z) as well as the products of binary and continuous variables (u, w, o, p, q), and therefore can apply Benders' decomposition. In problem (17), for each possible choice $\bar{x}$, $\bar{y}$ and $\bar{z}$, we find the best choices for u, w, o, p, q by solving a linear program. So we regard u, w, o, p, q as a function of x, y, z. Then we replace the contribution of u, w, o, p, q to the objective with a scalar variable representing the value of the best choice for a given $\bar{x}$, $\bar{y}$ and $\bar{z}$. We start with a crude approximation to the contribution of u, w, o, p, q and then generate a sequence of dual solutions to tighten up the approximation.

Next, we will detail the formulation of the corresponding subproblem (LP problems) and master problem (an integer programming (IP) problem) after the Benders' decomposition.

B. Classical Optimization for Subproblem

For the subproblem, we consider a distributed optimization where each server can solve its own optimal subproblem. Based on the decomposition in Section III-A, the subproblem for each server is defined as follows.

$$\min_{u,w,o,p,q} \sum_{j=1}^{W}(\sum_{k=1}^{N} a_{1,i,j,k} \cdot o_{k,i,j} + a_{2,i,j} \cdot p_{i,j} + a_{3,i,j} \cdot q_{i,j}) \tag{27}$$

$$\text{s.t.} \quad (16), (18) - (26). \tag{28}$$

Let $f_i(x, y, z)$ represent the subproblem value of i-th server, then the overall subproblem is

$$SUP(x, y, z) = \sum_{i=1}^{N} f_i(x, y, z)$$

The subproblem for each server can be further represented in a general form as follows.

$$f_i(x, y, z) = \min \quad \mathbf{d}_i^\intercal \mathcal{Y}_i$$
$$\text{s.t.} \quad \mathcal{A}\mathcal{Y}_i \geq \mathcal{B}\mathcal{X}_i + \mathcal{C},$$

where $\mathcal{Y}_i = [\mathbf{o}_{k,i,j}, \mathbf{p}_{i,j}, \mathbf{q}_{i,j}, \mathbf{u}_j, \mathbf{w}_j]^\intercal$, $\mathcal{X}_i = [\mathbf{x}_{i,j}, \mathbf{y}_{i,j}, \mathbf{z}_{k,i,j}]^\intercal$, and $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are coefficients in the constraints. In addition, the dual problem of the subproblem is defined below and π_i is the dual variable.

$$\max \quad (\mathcal{B}\mathcal{X}_i + \mathcal{C})^\intercal \pi_i \tag{29}$$

$$\text{s.t.} \quad \mathcal{A}^T \pi_i \leq \mathbf{d}_i, \tag{30}$$

$$\pi_i \geq 0, \tag{31}$$

where $\mathbf{d}_i = \begin{bmatrix}\mathcal{E}_i & \mathcal{F}_i & \mathcal{G}_i & 0 & \cdots & 0\end{bmatrix}^\intercal$, $\mathcal{E}_i = \begin{bmatrix}a_{1,i,1,1} & \ldots & a_{1,i,1,N} & \ldots & a_{1,i,W,1} & \ldots & a_{1,i,W,N}\end{bmatrix}$, $\mathcal{F}_i = \begin{bmatrix}a_{2,i,1} & \ldots & a_{2,i,W}\end{bmatrix}$, $\mathcal{G}_i = \begin{bmatrix}a_{3,i,1} & \ldots & a_{3,i,W}\end{bmatrix}$. This problem can be solved by a classical LP solver.

C. Quantum Formulation for Master Problem

Based on the dual problem of the subproblem, the master problem can be defined as follows.

$$\min_{x,y,z} \quad \sum_{j=1}^{W}[\sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j}) + \lambda] \tag{32}$$

$$\text{s.t.} \quad (6) - (10),$$

$$x_{i,j} \in \{0, 1\}, y_{i,j} \in \{0, 1\}, \quad \forall i, j, \tag{33}$$

$$z_{k,i,j} \leq y_{i,j}, \quad \forall i, j, k, \tag{34}$$

$$z_{k,i,j} \leq x_{k,j}, \quad \forall i, j, k, \tag{35}$$

$$z_{k,i,j} \geq x_{k,j} + y_{i,j} - 1, \quad \forall i, j, k, \tag{36}$$

$$\lambda \geq \lambda^{down}, \tag{37}$$

$$\lambda \geq (\mathcal{B}\mathcal{X} + \mathcal{C})^\intercal \pi^l, \tag{38}$$

where λ is the optimal value of the subproblem aggregated from all servers at the current iteration. Constraints (37) and (38) are the corresponding Benders' cuts.

QUBO Formulation. Quantum annealers are able to solve the optimization problem in a QUBO formulation. To leverage the state-of-art quantum annealers provided by D-Wave, the master problem has to be converted to the corresponding QUBO formulation. Due to the rule of QUBO setup, we have to reformulate our constrained master problem as the

unconstrained QUBO by using penalties. The basic idea is to find the best penalty coefficients of the constraints. Following the principle of constraint-penalty pairs in [27], the constraints are converted as follows:

$$(6) \Rightarrow \quad \xi_1 : P^1_{i,j}(x_{i,j}\mu_j\kappa_j - c_i + \sum_{l=0}^{\bar{l}^1} 2^l s^1_l)^2,$$
$$\text{where} \quad \bar{l}^1 = \lceil \log_2[c_i - \min_x(x_{i,j}\mu_j\kappa_j)] \rceil.$$

$$(6) \Rightarrow \quad \xi_2 : P^2_{i,j}(x_{i,j}\chi_j - f_i + \sum_{l=0}^{\bar{l}^2} 2^l s^2_l)^2,$$
$$\text{where} \quad \bar{l}^2 = \lceil \log_2[f_i - \min_x(x_{i,j}\chi_j)] \rceil.$$

$$(7) \Rightarrow \quad \xi_3 : P^3_{i,j}(y_{i,j}\mu_j - c_i + \sum_{l=0}^{\bar{l}^3} 2^l s^3_l)^2,$$
$$\text{where} \quad \bar{l}^3 = \lceil \log_2[c_i - \min_y(y_{i,j}\mu_j)] \rceil.$$

$$(7) \Rightarrow \quad \xi_4 : P^4_{i,j}(y_{i,j}\chi_j - f_i + \sum_{l=0}^{\bar{l}^4} 2^l s^4_l)^2,$$
$$\text{where} \quad \bar{l}^4 = \lceil \log_2[f_i - \min_y(y_{i,j}\chi_j)] \rceil.$$

$$(8) \Rightarrow \quad \xi_5 : P^5_{i,j}(\sum_{i=1}^{N} x_{i,j} - 1)^2,$$

$$(8) \Rightarrow \quad \xi_6 : P^6_{i,j}(\sum_{i=1}^{N} y_{i,j} - \kappa_j)^2,$$

$$(9) \Rightarrow \quad \xi_7 : P^7_{i,j}[\sum_{j=1}^{W}(x_{i,j} + y_{i,j}) - 1 + \sum_{l=0}^{\bar{l}^7} 2^l s^7_l]^2,$$
$$\text{where} \quad \bar{l}^{10} = \lceil \log_2[1 - \min_{x,y} \sum_{j=1}^{W}(x_{i,j} + y_{i,j})] \rceil.$$

$$(34) \Rightarrow \quad \xi_8 : P^8_{i,j}(z_{k,i,j} - y_{i,j}z_{k,i,j}).$$

$$(35) \Rightarrow \quad \xi_9 : P^9_{i,j}(z_{k,i,j} - x_{k,j}z_{k,i,j}).$$

$$(36) \Rightarrow \quad \xi_{10} : P^{10}_{k,i,j}(x_{k,j} + y_{i,j} - 1 - z_{k,i,j} + \sum_{l=0}^{\bar{l}^{10}} 2^l s^{10}_l)^2,$$
$$\text{where} \quad \bar{l}^{10} = \lceil \log_2[\min_{x,y,z}(z_{k,i,j} - x_{k,j} - y_{i,j} + 1)] \rceil.$$

$$(37) \Rightarrow \quad \xi_{11} : P^{11}(\lambda^{down} - \lambda + \sum_{l=0}^{\bar{l}^{11}} 2^l s^{11}_l)^2,$$
$$\text{where} \quad \bar{l}^{11} = \lceil \log_2(\lambda - \lambda^{down}) \rceil.$$

$$(38) \Rightarrow \quad \xi_{12} : P^{12}((\mathcal{B}\mathcal{X} + \mathcal{C})^T\pi^l - \lambda + \sum_{l=0}^{\bar{l}^{12}} 2^l s^{12}_l)^2,$$
$$\text{where} \quad \bar{l}^{12} = \lceil \log_2[\lambda - \min_{x,y,z,\pi}(\mathcal{B}\mathcal{X} + \mathcal{C})^T\pi^l] \rceil.$$

Here, P^*, $P^*_{i,j}$ and $P^*_{k,i,j}$ are the predefined penalty coefficients when the corresponding constraint is violated. s^*_l is a binary slack variable and $\bar{l}^*$ is the upper bound of the number of slack variables. Then, the reformulated unconstrained master problem is defined below.

$$\min_{x,y,z} \quad \sum_{j=1}^{W}[\sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j}) + \lambda + \xi_1 + \xi_2 + \xi_3 + \xi_4 + \xi_5 + \xi_6 + \xi_7 + \xi_8 + \xi_9 + \xi_{10} + \xi_{11} + \xi_{12}] \tag{39}$$

Variable Representation. Now consider the problem in (39), it is still not in the QUBO formation due to the existence of the continuous variable λ. Thus, we need to represent the continuous variable λ using binary bits. We use a binary vector $\mathbf{w}$ with the length of M bits to replace continuous variable λ and denote it as a new discrete variable $\hat{\lambda} \in \mathbb{Q}$. In general, $\hat{\lambda}$ requires the binary numeric system assigning M bits to replace continuous variable λ. Then we can recover the $\hat{\lambda}$ by

$$\lambda = \sum_{ii=-\underline{m}}^{\bar{m}_+} 2^{ii} w_{ii+\underline{m}} - \sum_{jj=0}^{\bar{m}_-} 2^{jj} w_{jj+1+\underline{m}+\bar{m}_+} = \hat{\lambda}(w) \tag{40}$$

In (40), $\bar{m}_+ + 1$ is the number of bits for the positive integer part $\mathbb{Z}_+$, $\underline{m}$ is the number of bits for the positive decimal part and $\bar{m}_- + 1$ is the number of bits for the negative integer part $\mathbb{Z}_-$. Then, the final QUBO formulation of the master problem is defined as follows.

$$\min_{x,y,z,w} \quad \sum_{j=1}^{W}[\sum_{i=1}^{N} a_{4,i} \cdot (x_{i,j} + y_{i,j}) + \hat{\lambda}(w) + \xi_1 + \xi_2 + \xi_3 + \xi_4 + \xi_5 + \xi_6 + \xi_7 + \xi_8 + \xi_9 + \xi_{10} + \xi_{11} + \xi_{12}] \tag{41}$$

D. HQCBD Algorithm

Our proposed HQCBD is described by Algorithm 1. Fig. 2 shows the overall flow of HQCBD and the detailed interaction between the master problem and subproblem. The master problem is solved by a quantum computer and generates a binary solution (x', y', z'), and then sends it to general devices for distributed computation of subproblems by a classical solver (e.g. Scipy). After subproblems are solved, an optimality or feasibility cut is sent to the master problem and it continues to the next round.

Specifically, as shown in Algorithm 1, we first initialize the upper and lower bound of the problem as well as other parameters, e.g. convergence threshold ϵ and the number of maximal iterations max_itr (Lines 1-2). Then appropriate penalty numbers or arrays will be generated (Line 4). After that, we reformulate the master problem in (32) in the QUBO format and solve the QUBO problem with a quantum computer and update the lower bound of the problem $\underline{\lambda}$ (Lines 5-7). Given x', y', z' from the master problem, we solve the subproblem (29) and extract ϱ' as well as update the upper bound of the problem $\overline{\lambda}$ (Lines 8-10). We finally add the Benders' cut to the master problem and continue the next iteration (Lines 11-12) until it converges (Line 3).

We leverage the D-Wave solver to implement our proposed algorithm to solve the QUBO master problem. In addition, the penalties also need to be carefully tuned for a decent QUBO model. In general, a large penalty can cause the

Algorithm 1 Hybrid Quantum-Classical Benders' Decomposition (HQCBD) Method

Input: Distributed network with N servers V, W FL models M, Coefficient of the objective function and constraints in master problem and subproblem

Output: PS selection x', worker selection y', and local convergence rate ϱ'

Initialize upper/lower bound of λ, $\overline{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
Initialize threshold $\epsilon = 0.001$, $max\_itr = 100$, $itr = 1$
while $|\overline{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max\_itr$ **do**
  $\mathbf{P} \leftarrow$ Appropriate penalty numbers or arrays
  $\mathbf{Q} \leftarrow$ Reformulate both objective and constraints in (32) and construct QUBO formulation as (41)
  x', y', $z' \leftarrow$ Solve problem (41) by quantum computer
  $\underline{\lambda} \leftarrow$ Extract $\mathbf{w}$ and replace $\underline{\lambda}$ with $\hat{\lambda}(w)$ as (40)
  $SUP(x, y, z) \leftarrow$ Solve problem (29) with fixed x', y', z'
  Extract ϱ' from $SUP(x, y, z)$
  $\overline{\lambda} \leftarrow SUP(x, y, z)$
  Add a new benders' cut to the master problem as (38)
  $itr += 1$
end while
return x', y', ϱ'

quantum annealer to malfunction due to coefficient explosion. In contrast, a small penalty can make the quantum annealer ignore the constraints. A well-tuned penalty will lead to a fairly high probability of the quantum solver giving the correct answer.

IV. Performance Evaluation

In this section, we simulated a distributed network environment and conducted experiments of realistic FL tasks using publicly available datasets. To validate the feasibility of our hybrid quantum-classical optimization algorithm, we run the proposed algorithms on a hybrid D-Wave quantum processing unit (QPU). We accessed the D-Wave system provided by Leap quantum cloud service [28]. Based on the Pegasus topology, the D-Wave system also has over $5,000$ qubits and $35,000$ couplers, which can solve complex problems of up to $1,000,000$ variables and $100,000$ constraints. We performed a number of test cases that can be resolved in under 100 iterations, but only due to the high cost of QPU utilization and the developer's time constraints.

A. Simulation Setup

Network Setting: Our distributed computing environment consists of 100 servers where the topology depends on the real-world EUA-Dataset [29] and the Internet topology zoo [30]. EUA-Dataset is widely used in mobile computing and contains the geographical locations of 125 cellular base stations in the Melbourne central business district area, while the Internet topology zoo is a popular network topology dataset that includes a number of historical network maps all over the world. We randomly select a set of servers from these topology datasets to conduct simulations. In each simulation, each server has a maximal storage capacity c_i, CPU frequency f_i and link bandwidth b_j belonging to the ranges of $1,024 \sim 2,048$GB, $2 \sim 5$GHz, and $512 \sim 1,024$Mbps, respectively.

Datasets and FL models: We conduct extensive experiments on the following real-world datasets: California Housing dataset [31], MNIST [32], Fashion-MNIST (FMNIST) [33], and CIFAR-10 [34]. These are well-known ML datasets for linear regression, logistic regression, or image classification tasks. Two models with convex loss functions are implemented on the above real-world datasets for performance evaluation, which are (i) Linear Regression with MES loss on the California Housing dataset and (ii) Logistic Regression with the cross-entropy loss on MNIST. We are also interested in the performance of our proposed methods on FL models with non-convex loss functions. Thus, three datasets, MNIST, FMNIST, and CIFAR-10, are used to train convolutional neural network (CNN) models with different structures.

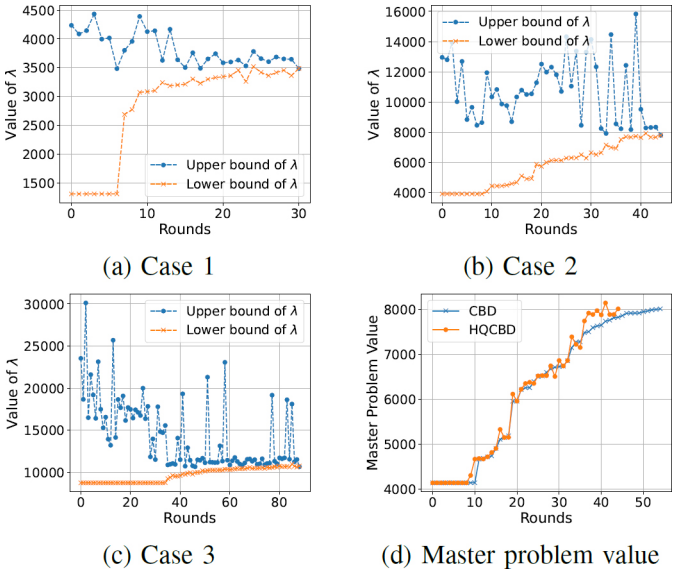
Benchmarks and Metrics: We compare our proposed HQCBD with three baseline strategies: classical Benders' decomposition (CBD), random algorithm (RAND), and two-stage iterative optimization algorithm (TWSO) [15]. CBD uses a classical LP solver (Gurobi [35] or Scipy [36]) to solve the master problem and subproblems. RAND randomly generates the random decisions on the model's parameter server, FL workers, and local convergence rate under certain constraints. TWSO is a previous algorithm from [15] that decomposes the original problem into two subproblems (participant selection and learning scheduling) and solves them iteratively. Since the optimization problem in [15] has a different learning cost function, we adjust their method to our optimization problem for fairness. The following metrics are adopted to compare the performances of our proposed methods and the baselines: *the total cost* of FL training, *the loss or accuracy* of FL models, *the number of iterations*, and *the solver accessing time*.

TABLE I: Iteration of CBD and HQCBD over three different cases. Here, the set up column shows {# of servers, # of models, # of workers per model} used in each case.

Case	Set up	# of Variables	Itr. of CBD	Itr. of HQCBD
1	$\{7, 1, 3\}$	63	32	31
2	$\{7, 2, 2\}$	126	55	45
3	$\{9, 2, 3\}$	198	91	89

B. Performance of HQCBD

To demonstrate the feasibility and performance of our proposed HQCBD, we conduct three sets of small-scale experiments with different case settings (servers are selected from 100 servers). As shown in Table I, there are three cases. The first case includes 7 servers, 1 FL model, and 3 workers per model with a total of 63 binary variables. The second case has 7 servers, 2 FL models, and 2 workers per model with a total of 126 binary variables. The third case consists of 9 servers, 2 FL models, and 3 workers per model with a total of 198 binary variables. For each case, we perform both CBD and HQCBD. Fig. 3 and Table I show the related results of their performances.



(a) Case 1 (b) Case 2

(c) Case 3 (d) Master problem value

Fig. 3: Performance of HQCBD: its convergence.

In Figs. 3(a)-(c), the blue dashed line denotes the upper bound of value λ used in HQCBD, and the orange dashed line denotes the lower bound of λ in HQCBD. As we can see, the upper bound and lower bound finally converge and we obtain the non-negative lower bound at the 31st, 45th and 89th round for each case, respectively. This result proves that our proposed algorithm is mathematically consistent with the classical Benders' decomposition algorithm. In addition, Fig. 3(d) shows the trend of the master problem value of Case 2 calculated by (39) compared with the solution of CBD. We can see that the value of the master problem keeps increasing until it converges. Specifically, the master problem value keeps static in the first few rounds since only an unbounded ray is found in the subproblem and a feasibility cut is added to the master problem. As we run more iterations, the optimality cut is found and added to the master problem. Once the difference between the upper bound and lower bound reaches a threshold, the problem is solved. The solution from HQCBD is similar to the one from CBD. Table I further demonstrates the detailed comparison between CBD and HQCBD in terms of the number of iterations used to solve the problem. We can find that HQCBD takes fewer iterations to converge to the optimal solution compared with CBD (for example, for Case 2, the improvement of iterations is around 18%).

TABLE II: Solver accessing time (ms) of CBD and HQCBD.

Case	CBD		HQCBD	
	Max./Min.	Mean/Std.	Max/Min.	Mean/Std.
1	190.47/6.71	117.14/50.12	32.10/15.93	31.49/2.79
2	235.29/9.11	129.56/50.04	32.11/15.92	24.10/7.98
3	395.48/14.45	120.25/63.19	32.11/16.00	25.53/7.85

Furthermore, we show the comparison of real solver accessing time (i.e., computation time of the solvers) for CBD and HQCBD in Table II and plot the detailed accessing time of Case 2 in Fig. 4. The solver accessing time is the

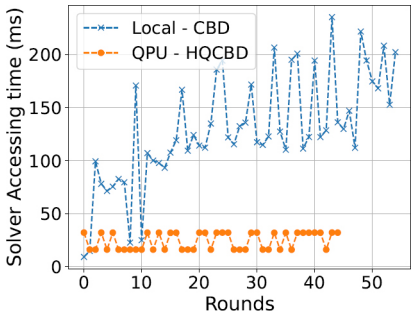


Fig. 4: Solver accessing time of CBD/HQCBD in Case 2.

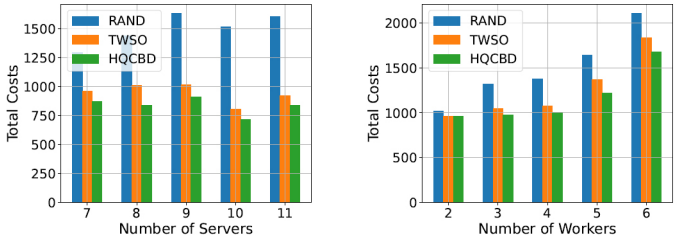
real accessing time of QPU solver and local solver without considering other overheads, such as variables setting time, parameters transmission time, and so on. As we can see in Table II, the minimal accessing time of CBD is relatively lower than that of HQCBD. However, the maximal and average accessing time as well as the standard deviation value of CBD is significantly higher than HQCBD. For example, for Case 2, the mean accessing time of HQCBD is 81% less than the one of CBD, and more significantly the standard deviation of accessing time of HQCBD is 84% less than the one of CBD. We also confirm via Fig. 4 that the solver accessing time of CBD in each round/iteration varies significantly while the solver accessing time of HQCBD in each round keeps stable and is even smaller than that of CBD. This finding proves the efficiency and robustness of leveraging the hybrid quantum-classical technique to solve the optimization problem in terms of either the convergence iteration or the solver accessing time.

C. Comparison with Existing Methods

We now compare our proposed method HQCBD with the random method (RAND) and a two-stage iterative optimization method (TWSO) [15] in terms of solving the joint optimization problem.

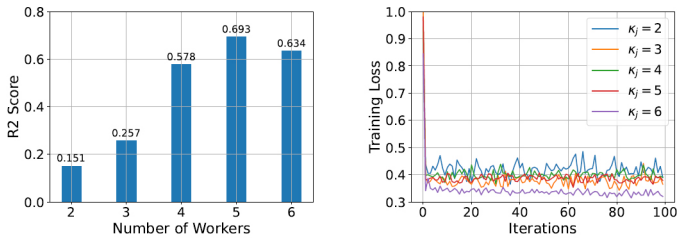
Firstly, we focus on the necessity of the optimization problem and study the impact of different numbers of servers. We concurrently train 2 FL models with 2 workers per model and the number of servers varies from 7 to 11. Fig. 5(a) shows the results. Obviously, RAND has the worst performance due to its randomness. Our HQCBD algorithm gets further improvements compared with TWSO and demonstrates the effectiveness of the HQCBD algorithm. In addition, as the number of servers increases, the total cost of HQCBD first decreases and increases then decreases again. This is because the topology may change when the server number varies and lead to the change of selection decision as well as the total cost.

Next, we investigate the impact of different numbers of FL workers on total costs. We set the number of servers and FL models to 15 and 2, respectively. The number of FL workers is in the range of $[2, 6]$. As shown in Fig. 5(b), the total costs increase as the number of workers increases. This is obvious since the more workers, the more total costs consumed. Our proposed HQCBD still outperforms RAND and TWSO algorithms. With more qubits supporting, we

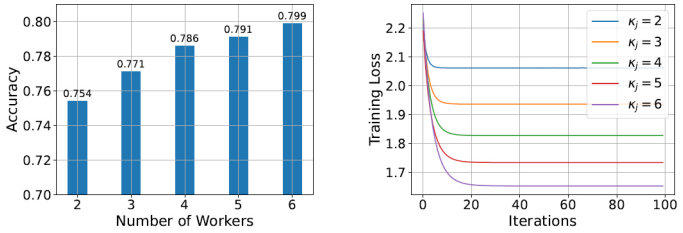


(a) Impact of server number (b) Impact of worker number

Fig. 5: Performance comparison with existing methods given different numbers of servers or FL workers.



(a) R2 score (b) LR over CA housing



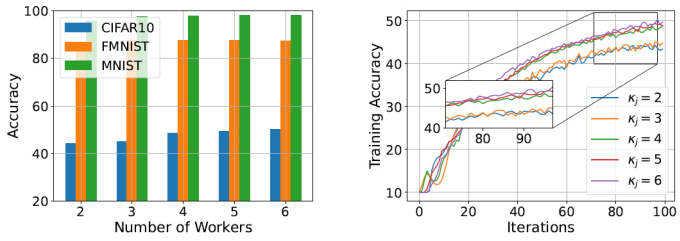
(c) Accuracy (d) LR over MNIST

Fig. 6: Training loss of linear and logistic regression models and impact from worker numbers: (a)(b) R2 scores and loss of linear regression model over California housing dataset; (c)(d) accuracy and loss of logistic regression model over MNIST dataset.

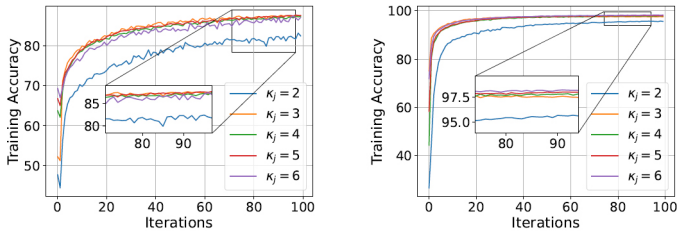
expect that the speed of HQCBD will have a more significant advantage over TWSO on large-scale optimization problems.

D. Performance of FL Model

Now, we look into the performance of our proposed methods in the real FL training process. We concurrently train 2 FL models with convex loss functions on the non-IID dataset settings: (i) Linear Regression with MSE loss over the California Housing dataset, (ii) Logistic Regression with cross-entropy loss over the MNIST dataset. Each dataset is split into 15 servers unequally and the number of global training rounds is set to 100 for clear comparison. We further introduce the R2 score metric to evaluate the performance of linear regression model training. R2 score is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). As shown in Fig. 6(a), the R2 score of the linear regression model rises as the number of workers increases. Also, with more FL workers, the training loss of the linear regression model decreases as illustrated in Fig. 6(b). Similarly, the accuracy of the logistic regression



(a) Accuracy (b) CNN over CIFAR-10



(c) CNN over FMNIST (d) CNN over MNIST

Fig. 7: Training accuracy of CNN models (non-convex) and impact from the number of workers: (a) accuracy of all CNN models; (b) accuracy of CNN over CIFAR-10 dataset; (c) accuracy of CNN over FMNIST dataset; (d) accuracy of CNN over MNIST dataset.

model also increases with more FL workers while the training loss declines with the rise of the number of workers as shown in Figs. 6(c)-(d). These results further indicate the feasibility and effectiveness of our proposed algorithm.

Finally, to demonstrate the performance of our proposed algorithms on FL models with non-convex loss functions, we conduct two sets of FL training with CNN models: (i) CNN models over CIFAR-10 and FMNIST datasets, and (ii) CNN models over CIFAR-10 and MNIST datasets. The experimental setting is similar to that in the convex experiment. Fig. 7(a)-(d) show the training accuracy of all CNN models under different numbers of workers. Obviously, with more workers, the training accuracy also increases, especially for the CIFAR-10 dataset as illustrated in Figs. 7(a) and (b). However, the training accuracy of FMNIST and MNIST datasets keep similar when the number of workers is larger than 2 as shown in Figs. 7(c) and (d). This is mainly due to the non-IID setting and simplicity of FMNIST and MNIST datasets since MNIST and FMNIST are both grayscale images from 10 categories.

V. Conclusion

In this paper, a joint participant selection and learning scheduling problem for multi-model FL has been studied. Motivated by the powerful parallel computing capabilities of quantum computers, we proposed a quantum-assisted HQCBD algorithm by employing the complementary strengths of classical optimization and quantum annealing to optimally select participants (both PS and FL workers) and determined the learning schedule to minimize the total cost of all FL models. Extensive simulations on the D-Wave quantum annealing machine demonstrated the efficiency and robustness of our proposed HQCBD algorithm which not only achieved the same

result as the classical algorithm but also took much fewer iterations (up to 18% improvement) and less accessing time (up to 81% reduction) to obtain the desired solution even at relevantly small scales. With the new development of robust quantum computers with more qubits, we believe that the proposed HQCBD-based method will have great applications in the joint learning scheduling of distributed machine learning in the near future.

REFERENCES

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[2] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, Dec. 2020.

[3] S. Niu and A. Todri-Sanial, "Effects of dynamical decoupling and pulse-level optimizations on IBM quantum computers," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–10, Aug. 2022.

[4] Ö. Salehi, A. Glos, and J. A. Miszczak, "Unconstrained binary models of the travelling salesman problem variants for quantum optimization," *Quantum Information Processing*, vol. 21, no. 2, p. 67, Jan. 2022.

[5] D. An and L. Lin, "Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm," *ACM Transactions on Quantum Computing*, vol. 3, no. 2, pp. 1–28, Jun. 2022.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, Apr. 2017.

[7] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," *IEEE Intelligent Systems*, vol. 37, no. 02, pp. 27–34, Mar. 2022.

[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, Nov. 2019.

[9] X. Wei, J. Liu, X. Shi, and Y. Wang, "Participant selection for hierarchical federated learning in edge clouds," in *IEEE International Conference on Networking, Architecture, and Storage (NAS)*, Philadelphia, PA, Oct. 2022.

[10] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE International Conference on Communications*, Shanghai, China, May 2019.

[11] H. Zhu, Y. Zhou, H. Qian, Y. Shi, X. Chen, and Y. Yang, "Online client selection for asynchronous federated learning with fairness consideration," *IEEE Transactions on Wireless Communications*, Oct. 2022, doi: 10.1109/TWC.2022.3211998.

[12] Y. Li, F. Li, L. Chen, L. Zhu, P. Zhou, and Y. Wang, "Power of redundancy: Surplus client scheduling for federated learning against user uncertainties," *IEEE Transactions on Mobile Computing*, May 2022, doi: 10.1109/TMC.2022.3178167.

[13] 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, Mar. 2019.

[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 Conference on Computer Communications (INFOCOM)*, Virtual, May 2021.

[15] X. Wei, J. Liu, and Y. Wang, "Joint participant selection and learning scheduling for multi-model federated edge learning," in *IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, Denver, CO, Oct. 2022.

[16] T. Tran, M. Do, E. Rieffel, J. Frank, Z. Wang, B. O'Gorman, D. Venturelli, and J. Beck, "A hybrid quantum-classical approach to solving scheduling problems," in *Proceedings of the International Symposium on Combinatorial Search*, New York, USA, Jul. 2016.

[17] A. Ajagekar, T. Humble, and F. You, "Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems," *Computers & Chemical Engineering*, vol. 132, p. 106630, Jan. 2020.

[18] Z. Zhao, L. Fan, and Z. Han, "Hybrid quantum benders' decomposition for mixed-integer linear programming," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, TX, Apr. 2022.

[19] A. Ajagekar, K. Al Hamoud, and F. You, "Hybrid classical-quantum optimization techniques for solving mixed-integer programming problems in production scheduling," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–16, Jun. 2022.

[20] L. Fan and Z. Han, "Hybrid quantum-classical computing for future network optimization," *IEEE Network*, vol. 36, no. 5, pp. 72–76, Nov. 2022.

[21] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, and X. Wang, "Resource-efficient and convergence-preserving online participant selection in federated learning," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, Singapore, Feb. 2020.

[22] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, Oct. 2020.

[23] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, Nov. 2020.

[24] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, Feb. 2017.

[25] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning (ICML)*, Beijing, China, Jun. 2014.

[26] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The benders decomposition algorithm: A literature review," *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, Jun 2017.

[27] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: a tutorial on formulating and using qubo models," *4OR-Q J Oper Res*, vol. 17, pp. 335–371, Nov. 2019.

[28] D-wave hybrid solver service: An overview. [Online]. Available: https://www.dwavesys.com/resources/white-paper/d-wave-hybrid-solver-service-an-overview/

[29] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *International Conference on Service-Oriented Computing*, Hangzhou, China, Nov. 2018.

[30] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Feb. 2011.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[33] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *ArXiv*, vol. abs/1708.07747, Aug. 2017.

[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., Apr. 2009.

[35] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," Jan. 2023. [Online]. Available: https://www.gurobi.com

[36] P. Virtanen, R. Gommers *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Feb. 2020.