Hybrid Quantum-Classical Benders' Decomposition for Federated Learning Scheduling in Distributed Networks

Xinliang Wei, *Member, IEEE*, Lei Fan, *Senior Member, IEEE*, Yuanxiong Guo, *Senior Member, IEEE*, Yanmin Gong, *Senior Member, IEEE*, Zhu Han, *Fellow, IEEE* and Yu Wang, *Fellow, IEEE*

Abstract—Scheduling multiple federated learning (FL) models within a distributed network, especially in large-scale scenarios, poses significant challenges since it involves solving NP-hard mixed-integer nonlinear programming (MINLP) problems. However, it's imperative to optimize participant selection and learning rate determination for these FL models to avoid excessive training costs and prevent resource contention. While some existing methods focus solely on optimizing a single global FL model, others struggle to achieve optimal solutions as the problem grows more complex. In this paper, exploiting the potential of quantum computing, we introduce the Hybrid Quantum-Classical Benders' Decomposition (HQCBD) algorithm to effectively tackle the joint MINLP optimization problem for multi-model FL training. HQCBD combines quantum and classical computing to solve the joint participant selection and learning scheduling problem. It decomposes the optimization problem into a master problem with binary variables and small subproblems with continuous variables, then leverages the strengths of both quantum and classical computing to solve them respectively and iteratively. Furthermore, we propose the Hybrid Quantum-Classical Multiple-cuts Benders' Decomposition (MBD) algorithm, which utilizes the inherent capabilities of quantum algorithms to produce multiple cuts in each round, to speed up the proposed HQCBD algorithm. Extensive simulation on the commercial quantum annealing machine demonstrates the effectiveness and robustness of the proposed methods (both HQCBD and MBD), with improvements of up to 70.3% in iterations and 81% in computation time over the classical Benders' decomposition algorithm on classical CPUs, even at modest scales.

Index Terms—Federated Learning, Participant Selection, Learning Scheduling, Hybrid Quantum-Classical Optimization

1 Introduction

With the advancement of technology, quantum computing (QC) has gained widespread attention due to the realization of speedups offered by quantum techniques for complex computational problems. This has resulted in transformational breakthroughs on specific tasks accomplished with near-term quantum computers. QC has more computational power than classical computers and may be faster at solving complex optimization problems, e.g., random quantum circuit sampling [1], Gaussian boson sampling [2], and combinatorial optimization [3]–[5]. In this paper, by leveraging the parallel computing capability of QC, we focus on designing a new quantum-inspired scheduling algorithm to solve a complex joint participant selection and learning scheduling problem for federated learning (FL) in distributed networks.

FL is a distributed artificial intelligence (AI) approach that allows for the training of high-quality AI models by aggregating local updates from multiple FL clients (or workers), such as IoT devices, without direct access to the local data [6]–[10]. This potentially prevents the disclosure of sensitive user information and preferences, reducing the risk of privacy leakage. Nevertheless, when deploying the FL framework in distributed networks, there are two challenges. *First*, the computing power and network resources of servers, as well as their data distributions, are diverse. Some low-performance servers may cause the convergence process to slow down and reduce training performance. Furthermore, dispersed computing resources and high network latency may result in high training costs. *Second*, in the practical scenario, concurrently training multiple models in the shared distributed network creates competition for computing and communication resources. As shown in Fig. 1, two FL models are trained concurrently and each FL model requires one parameter server (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.

As a result, we concentrate primarily on the problem of joint participant selection and learning scheduling in multi-model FL training scenarios. It should be noted that in

• *X. Wei and Y. Wang are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19112. Email: {xinliang.wei, wangyu}@temple.edu. L. Fan is with the Department of Engineering Technology and Department of Electrical and Computer Engineering at the University of Houston, Houston, TX 77004. Email: lfan8@central.uh.edu. Z. Han is with the Department of Electrical and Computer Engineering at the University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea, 446-701. Email: hanzhu22@gmail.com. Y. Guo and Y. Gong are with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, respectively. Email: {yuanxiong.guo, yanmin.gong}@utsa.edu. X. Wei and Y. Wang are the co-corresponding authors.* *The work is partially supported by the US NSF (Grant No. CCF-1908843, CNS-2006604, CNS-2107216, CNS-2128368, CMMI-2222810, ECCS-2302469, CNS-2106761, and CMMI-2222670), the US Department of Transportation, Toyota, Amazon and Japan Science and Technology Agency (JST) Adopting Sustainable Partnerships for Innovative Research Ecosystem (ASPIRE) JPMJAP2326.*

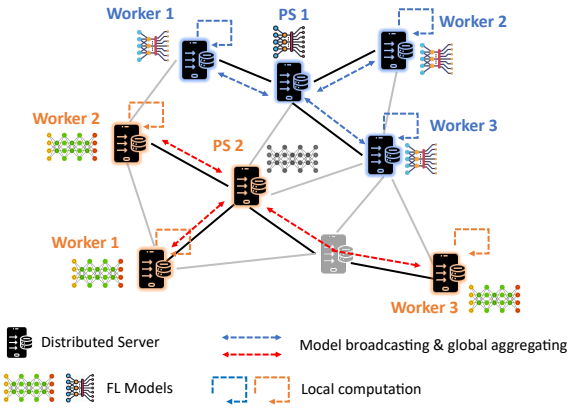


Fig. 1: The training process of distributed federated learning.

distributed networks, any server can serve as either a PS or a client, and that participant selection includes selecting both the PS and clients for each FL model. For clarity, we refer to a client as an FL worker. It is worth noting that both participant (client) selection and learning scheduling problems have been studied in FL using classical computers recently [11]–[14]. However, most existing works focus 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] formulated a MINLP model for the joint participant selection and learning scheduling problem in multi-model federated edge learning, and proposed multi-stage methods to solve the joint optimization problem. Nevertheless, due to the nature of the formulated optimization as a MINLP problem, the proposed methods may not lead to optimal solutions and may not scale well when the problem grows more complex.

To address the aforementioned issue, quantum computing has recently emerged as a powerful optimization tool [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). Still, 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, commercial quantum annealers can provide much more qubits, compared with gated-based quantum computers at the current stage. The quantum annealer can solve the quadratic unconstrained binary optimization (QUBO) problem using the Ising model. The constrained integer programming model of the master problem can be reformulated as the QUBO model. As a result, we attempt to develop novel hybrid quantum-classical algorithms on the quantum annealer.

Three research challenges exist in developing efficient hybrid quantum-classical techniques with decomposition schemes. *First, how to convert our original MINLP problem into an integer program (IP) problem and even further convert it into a QUBO model as an input to the quantum annealer? Second, how to design a novel hybrid quantum-classical strategy that solves the corresponding problem in fewer iterations? Last, how to derive an efficient number of integer cuts that iteratively reduce the search space and accelerate the convergence of the hybrid quantum-classical methods?*

To handle the above challenges, we develop two novel hybrid quantum-classical algorithms to demonstrate the potential of such hybrid approaches. To address the first challenge, we leverage the linearization and Benders' decomposition (BD) technique which is widely employed for solving MILP problems to convert our MINLP problem into an integer programming (IP) master problem and linear programming (LP) subproblems and then present a HQCBD algorithm. The master problem will be solved by the quantum annealer while subproblems will be solved by the classical computer. For the second and third challenges, we further design a multiple-cut version of HQCBD (MBD), by employing quantum computers in solving the master problem which can provide multiple feasible solutions. Following that, multiple subproblems are constructed based on these feasible solutions and each subproblem returns a Benders' cut to the master problem. By doing so, multiple cuts will be added to the master problem as constraints and further hasten the convergence speed. At last, we investigate the impact of different numbers of cuts of the MBD strategy by setting up various cases. The major 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 a MINLP problem, aiming to minimize the total learning cost.
- We then propose a novel HQCBD algorithm to tackle the joint optimization problem. By leveraging the combination of quantum computing and classical optimization techniques, our HQCBD algorithm can quickly converge to the desired solution as the classical BD algorithm does but with much fewer iterations and faster speeds.
- We further present a multiple-cuts version of HQCBD (MBD), to accelerate the convergence speed by taking multiple outputs from the quantum annealer to generate multiple cuts in each round. By selecting various numbers of cuts, MBD can achieve varying levels of performance improvement.
- We conduct extensive simulations with real FL tasks as well as the commercial quantum computer to evaluate our proposed algorithms. Numerous experiments have demonstrated that our proposed HQCBD and MBD can achieve significant advancement (up to 70.3% saving of iterations and 81% reduction of computation time) compared to the BD algorithm on classical CPUs even at small scales.

The remainder of this article is organized as follows. Section 2 presents an overview of the related works. In Section 3, the system model and problem formulation are introduced. Section 4 presents the Benders' decomposition, quantum formulation, and our proposed HQCBD and MBD algorithms. Performance evaluation is discussed in Section 5 and conclusions are presented in Section 6. A preliminary version of this paper appears as [21].

2 Related Work

2.1 Federated Learning

Federated learning emerges as an efficient distributed machine learning approach to exploit distributed data and computing resources, so as to collaboratively train machine learning models. Currently, the efforts of FL have focused on the communication and energy efficiency [6], [22], [23], the convergence and adaptive control [13], [24], the resource allocation and model aggregation [25]–[27]. For example, Yang *et al.* [22] studied the joint computation and transmission optimization problem aiming to minimize the total energy consumption for FL over wireless communication networks, then proposed an iterative algorithm to derive a near-optimal solution. Li *et al.* [23] formulated a compression control problem and proposed a convergence-guaranteed FL algorithm with flexible communication compression that allows participants to compress their gradients to different levels before uploading to the central server. 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. In addition, various studies related to hierarchical federated learning (HFL) were discussed, which focus on the use of edge computing and mobile devices. Liu *et al.* [24] proved that HFL can achieve convergence by using cloud and edge servers as two-tier parameter servers to aggregate the partial models from mobile clients. Luo *et al.* [25] investigated the optimization problem of resource allocation and edge association to minimize global costs for device users under HFL. Wang *et al.* [26] examined the formation of cluster structures in HFL, where edge servers are clustered for model aggregation. Also, Meng *et al.* [27] studied federated edge learning by using decentralized P2P methods. While some of these works also consider learning control of FL, they either consider different FL topologies (e.g. HFL) or optimize different objectives.

2.2 Client Selection and Learning Scheduling

Client selection and learning scheduling are critical problems, particularly in distributed FL where it is inevitable to communicate among servers. Hence, client selection or client sampling has been well studied in FL recently [11], [12], [28]–[32]. For example, Nishio and Yonetani [11] studied a client selection problem in edge computing where the edge server acts as a PS and numerous mobile clients are selected as workers. Their client selection aimed to maximize the number of selected workers under time constraints. Cho *et al.* [28] conducted a convergence analysis of FL using biased client selection and found that selecting clients with higher local losses leads to faster convergence than using unbiased client selection. Ribero and Vikalo [29] proposed a modified FedAvg algorithm for updating the global model in communication-constrained settings based on collecting models from clients and only clients whose model difference exceeds the threshold will be sampled for global updates. Marnissi *et al.* [30] further designed a client selection strategy based on the gradient norms importance to improve the communication efficiency of FL. Similarly, Balakrishnan *et al.* [32] also introduced diversity in the client selection problem by leveraging submodular maximization. Lai *et al.* [31] proposed a framework to guide participant selection in FL aiming to improve the training performance and indicated that clients with the greatest utility can improve model accuracy and hasten the convergence speed. Furthermore, Jin *et al.* [12] examined both the learning control of FL and the edge provisioning problem in distributed networks. Although their work is similar to ours, they did not take into account the selection of PS, and in their scenario, the remote cloud center always served as the PS. In addition, all aforementioned works do not take the concurrent multiple FL models training case into account which significantly affects the total training performance of all FL models.

Recently, we have considered participant selection to minimize the total learning cost in multi-model federated edge learning [9], [15]. In [9], we studied a participant selection problem for multiple hierarchical FL sessions with a fixed learning rate, which is a completely different setting from this paper. In [15], we formulated a joint participant selection and learning scheduling problem in multi-model FL and proposed both two/three-stage optimization methods and greedy heuristics to solve the joint optimization problem. The studied optimization problem in [15] is similar but different to the problem we formulate here since we introduce a new learning cost item - participant cost which the FL model owner needs to pay the participants of FL training. In addition, in this paper, we use a quantum-assisted approach to tackle the problem, while in [15] we use classical optimization approaches. In Section 5, we will compare the performance of a modified version of the two-stage method in [15] with our hybrid quantum approach for our problem.

2.3 Hybrid Quantum Optimization

Quantum computing (QC) [33] has been proven to be superior to solving many challenging computationally intensive problems [3]–[5], [34]–[36]. However, the application of QC is limited by the current state of a quantum computer (such as availability or cost). To address this, a hybrid quantum-classical computing framework has been developed for solving a complex optimization problem where both quantum and classical computers are used.

Such hybrid quantum optimization has been newly applied in different areas including machine learning, mobile computing, network communication, task scheduling, and classification [16]–[20], [37]. For instance, Tran *et al.* [16] first proposed a hybrid quantum-classical approach to solve the complete tree search problem. They decomposed the original problem into the master problem and subproblems where both master problem and subproblems were solved by quantum annealer, and the global search tree was maintained by the classical computer. Ajagekar *et al.* [19] proposed two hybrid QC-based optimization techniques

for solving large-scale mixed-integer linear programming (MILP) and mixed-integer fractional programming (MIFP) scheduling problems. Similarly, both [18] and [20] introduced a hybrid quantum-classical algorithm by leveraging a different decomposition technique (Benders' Decomposition (BD)) to solve the MILP optimization problem. Paterakis [37] also provided a hybrid quantum-classical optimization algorithm for unit commitment problems and further introduced a method for employing various cut selection criteria in order to control the size of the master problem.

Inspired by the aforementioned works, we apply the hybrid quantum-classical framework proposed by [18], [20] to tackle a specific real-world optimization problem that jointly optimizes the participant selection and learning schedule in multi-model FL. For this particular problem, we propose a distinct solving process where the binary master problem is solved by quantum annealer and subproblems with continuous variables however are addressed by the classical computer. Different from [18], [20], we also consider the multiple-cuts strategy to hasten the convergence speed.

3 System Model and Problem Formulation

In this section, we first introduce our system model, federated learning model, and associated cost model. Then we formulate the studied learning scheduling problem.

3.1 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 sc_i and CPU frequency sf_i while each link e_j has an available bandwidth b_j. Each server holds a distinct set of datasets for local training. We assume that each server can keep multiple types of datasets (e.g., text, image) 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).

3.2 Federated Learning Model

We assume that parallel FL was conducted where multiple models are being trained concurrently in the network. We consider a classical FL process that consists of a PS and multiple workers. Instead of using a single centralized server as the PS of all models, we select a group of servers distributed in the network with enough capacity as its participants to jointly train the FL model. Assume that W FL models ($M = \{m_1, \cdots, m_W\}$) are trained concurrently and each FL model has certain requirements for the training task, i.e.,

1) κ_j FL workers and one PS, with the minimum required CPU frequency χ_j and model size μ_j, respectively.
2) the minimum required global convergence rate 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 model parameters of model m_j on the 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

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

Generally, FL will perform round by round and we denote the total number of global aggregation, 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 same local update rule as

$$\omega_{i,j}^{\alpha,\beta} = \omega_{i,j}^{\alpha,\beta-1} - \eta \nabla F_{i,j}(\omega_{i,j}^{\alpha,\beta-1}), \tag{2}$$

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

$$F_{i,j}(\omega_{i,j}^{\alpha,\hat{\beta}}) - F_{i,j}^* \leq \varrho_j [F_{i,j}(\omega_{i,j}^{\alpha,0}) - F_{i,j}^*]. \tag{3}$$

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 the global model parameters as

$$\omega_j^{\alpha} = \sum_{i \in S_j} \frac{D_{i,j}}{D_j} \omega_{i,j}^{\alpha-1,\hat{\beta}}, \tag{4}$$

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

$$\mathcal{G}_j(\omega_j^{\hat{\alpha}}) - \mathcal{G}_j^* \leq \varsigma_j [\mathcal{G}_j(\omega_j^0) - \mathcal{G}_j^*], \tag{5}$$

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

To achieve the desired local convergence rate ϱ_j and global convergence rate ς_j, we must determine the number of local updates $\hat{\beta} = \varphi_j$ and global iterations $\hat{\alpha} = \vartheta_j$ based on equations (3) and (5). As per the above observation, we can predefine the global convergence rate ς_j for each FL model and perform local updates and global iterations accordingly. This leads to a relationship between the convergence rate and the number of local updates and global iterations [12], [22], [38]–[40].

$$\vartheta_j \geq \frac{2\lambda^2}{\gamma^2 \xi} ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1-\varrho_j} \triangleq \vartheta_0 \ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1-\varrho_j}, \tag{6}$$

$$\varphi_j \geq \frac{2}{(2-\lambda\delta)\delta\gamma} log_2\left(\frac{1}{\varrho_j}\right) \triangleq \varphi_0 log_2\left(\frac{1}{\varrho_j}\right), \tag{7}$$

where ξ and δ are two variables in ranges (0, $\frac{\gamma}{\lambda}$] and (0, $\frac{2}{L}$) related to the λ-Lipschitz parameter and γ-strongly convex parameter, respectively. ϑ_0 and φ_0 are two constants with the definition $\vartheta_0 = \frac{2\lambda^2}{\gamma^2\xi}$ and $\varphi_0 = \frac{2}{(2-\lambda\delta)\delta\gamma}$.

3.3 Cost Model

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

Transmission Cost: The primary component of the transmission cost is attributed to the costs of uploading and downloading the FL model. We use the model size μ_j of FL model m_j to represent the amount of data uploaded and downloaded. To determine the transmission cost for downloading models from the PS or uploading models to the PS, we utilize the shortest path within the distributed network. 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. Then, the total transmission cost is

$$C_j^{trans} = 2 \cdot \vartheta_j \sum_{k=1}^{N} \sum_{i=1}^{N} x_{k,j} \cdot y_{i,j} \cdot \rho_j(v_i, v_k). \tag{8}$$

Here, v_i and v_k represent the candidates for j-th FL model' worker and PS, respectively. The decision variables, $x_{k,j}$ and $y_{i,j}$, determine whether server v_k is selected as a PS and v_i is selected as an FL worker for the j-th FL model. Note that in our model any server can only be assigned to train one FL model and perform one role at a time, which will be a constraint (12d) on $x_{k,j}$ and $y_{i,j}$ in our formulated problem.

Local Training Cost: The cost of local training for the j-th FL model is defined below, where the function $\psi(\cdot)$ is used to define the CPU cycles required to process the sample data $D_{j,i}$.

$$C_j^{local} = \vartheta_j \cdot \varphi_j \cdot \sum_{i=1}^{N} y_{i,j} \cdot \frac{\psi(D_{j,i})}{sf_i}. \tag{9}$$

Global Aggregation Cost: The cost of global aggregation of all uploaded models uses the similar $\psi(\cdot)$ function and is defined as follows.

$$C_j^{global} = \vartheta_j \cdot \sum_{i=1}^{N} x_{i,j} \cdot \frac{\psi(\mu_j)}{sf_i}. \tag{10}$$

Participant Cost: Each participant of FL model m_j will be paid a basic rental cost for utility management which is related to their CPU frequency. Let p_j be the unit price for a CPU unit, accordingly, the participant cost for jth FL model is defined as

$$C_j^{rent} = \sum_{i=1}^{N} (x_{i,j} + y_{i,j}) \cdot p_j \cdot sf_i. \tag{11}$$

3.4 Problem Formulation

In the multi-model FL scenario introduced earlier, we aim to determine the participants for each model and schedule their local and global updates. We assume that each model has only one parameter server and κ_j workers, i.e., $\sum_{i=1}^{M} x_{i,j} = 1$ and $\sum_{i=1}^{M} y_{i,j} = \kappa_j$. The maximal local convergence rate of model m_j is represented by $\varrho_j \in [0.01, 0.99]$, while ς_j is the predetermined requirement for the number of global iterations and local updates for model m_j. The decision variables for our optimization are $x_{i,j}$, $y_{i,j}$, and ϱ_j.

We will now present the formulation of our problem, which involves selecting the optimal PS and workers for each model in the distributed network while achieving the desired local convergence rate. The objective is to minimize the total learning cost of all FL models, given by the following expression.

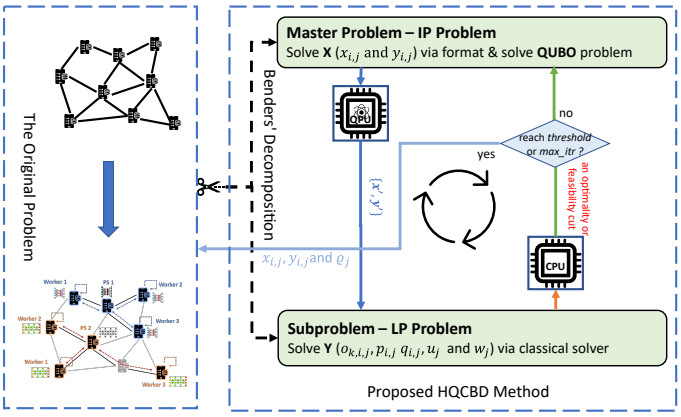


Fig. 2: The proposed HQCBD framework.

$$\min_{x,y,\rho} \quad \sum_{j=1}^{W} (C_j^{trans} + C_j^{local} + C_j^{global} + C_j^{rent}) \tag{12}$$

$$\text{s.t.} \quad x_{i,j}\mu_j\kappa_j \le sc_i, \quad x_{i,j}\chi_j \le sf_i, \quad \forall i, j, \tag{12a}$$

$$y_{i,j}\mu_j \le sc_i, \quad y_{i,j}\chi_j \le sf_i, \quad \forall i, j, \tag{12b}$$

$$\sum_{i=1}^{N} x_{i,j} = 1, \quad \sum_{i=1}^{N} y_{i,j} = \kappa_j, \quad \forall j, \tag{12c}$$

$$\sum_{j=1}^{W} (x_{i,j} + y_{i,j}) \le 1, \quad \forall i, \tag{12d}$$

$$i \in (1, \ldots, N), j \in (1, \ldots, W), \tag{12e}$$

$$x_{i,j} \in \{0, 1\}, y_{i,j} \in \{0, 1\}, \tag{12f}$$

$$\varrho_j \in [0.01, 0.99]. \tag{12g}$$

Constraints (12a) and (12b) ensure that the CPU and storage capacity of each FL model are satisfied. Constraint (12c) guarantees the number of PS and FL workers of each model is 1 and κ_j, respectively. Constraint (12d) states that each server can only be assigned to train one FL model and perform one role at a time. The decision variables and their ranges are given in (12e)-(12g). Note that the formulated problem (12) is a MINLP problem, which is NP-hard in general and challenging to solve with classical computing.

4 Hybrid Quantum Assisted Benders' Decomposition Methods

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.

We first briefly introduce the basic idea of BD. BD 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]. As can be seen in the right part of Fig. 2, 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 it is shown that no finite optimal solution exists or when the gap between the upper and lower bound is sufficiently small [41].

4.1 Problem Linearization and Reformulation

We now convert our original problem (12) to a form where BD can be applied. We first reformulate it by extracting all constant variables and further introducing additional continuous variables u_j and w_j to replace ϱ_j as below

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

$$\text{s.t.} \quad (12a)-(12g),$$

$$b_1 \le u_j \le b_2, \tag{13a}$$

$$b_3 \le w_j \le b_4, \tag{13b}$$

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 $a_{4,i} = \delta f_i$. Also, $u_j = \frac{1}{1-\varrho_j}$, $w_j = u_j log_2(\frac{u_j}{u_j-1})$, $b_1 = 1.01$, $b_2 = 100$, $b_3 = 1.435$ and $b_4 = 6.725$.

Note that Problem (13) consists of several terms that are the products of integer and continuous variables, e.g. $u_j\cdot x_{k,j}\cdot y_{i,j}, w_j\cdot y_{i,j}$. 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

$$\min_{x,y,u,w,o,p,q} \sum_{j=1}^{W}[\sum_{k=1}^{N}\sum_{i=1}^{N}a_{1,i,j,k}\cdot o_{k,i,j}+\sum_{i=1}^{N}a_{2,i,j}\cdot p_{i,j} + \sum_{i=1}^{N}a_{3,i,j}\cdot q_{i,j}+\sum_{i=1}^{N}a_{4,i}\cdot(x_{i,j}+y_{i,j})] \tag{14}$$

$$\text{s.t.} \quad (12a)-(12g),(13a),(13b),$$

$$b_1x_{k,j}y_{i,j} \le o_{k,i,j} \le b_2x_{k,j}y_{i,j}, \tag{14a}$$

$$u_j - o_{k,i,j} \le b_2(1-x_{k,j}y_{i,j}), \tag{14b}$$

$$u_j - o_{k,i,j} \ge b_1(1-x_{k,j}y_{i,j}), \tag{14c}$$

$$b_3y_{i,j} \le p_{i,j} \le b_4y_{i,j}, \tag{14d}$$

$$w_j - p_{i,j} \le b_4(1-y_{i,j}), \tag{14e}$$

$$w_j - p_{i,j} \ge b_3(1-y_{i,j}), \tag{14f}$$

$$b_1x_{i,j} \le q_{i,j} \le b_2x_{i,j}, \tag{14g}$$

$$u_j - q_{i,j} \le b_4(1-x_{i,j}), \tag{14h}$$

$$u_j - q_{i,j} \ge b_3(1-x_{i,j}). \tag{14i}$$

So far, we have linearized the product of binary and continuous variables as (u,w,o,p,q), and therefore we can apply BD. In problem (14), for each possible choice $\bar{x}$ and $\bar{y}$, we find the best choices for u,w,o,p,q by solving a linear programming. So we regard u,w,o,p,q as a function of x,y. 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 $\bar{x}$ and $\bar{y}$. 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. In addition, the problem (14) can be rewritten as a general form as follows.

$$\min_{\mathbf{X},\mathbf{Y}} \quad \mathbf{c}^\intercal\mathbf{X}+\mathbf{h}^\intercal\mathbf{Y} \tag{15}$$

$$\text{s.t.} \quad \mathbf{A}_1\mathbf{X} = \mathbf{a}_1, \tag{15a}$$

$$\mathbf{A}_2\mathbf{X} \le \mathbf{a}_2, \tag{15b}$$

$$\mathbf{X}^\intercal\mathbf{B}\mathbf{X}+\mathbf{G}\mathbf{Y} \le \mathbf{a}_3, \tag{15c}$$

$$\mathbf{X} = [x,y]^\intercal, \quad \mathbf{X}\in\mathbb{X}, \tag{15d}$$

$$\mathbf{Y} = [u,w,o,p,q]^\intercal, \quad \mathbf{Y}\in\mathbb{Y}, \tag{15e}$$

where $\mathbf{c}$ and $\mathbf{h}$ are coefficient vectors for binary and continuous variables in the objective function, respectively. $\mathbf{A}_1,\mathbf{A}_2,\mathbf{B},\mathbf{G}$ are coefficient matrices in the constraints while $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are constant vectors. Note $\mathbf{X}$ and $\mathbf{Y}$ are binary and continuous decision variables, respectively. Next, we will detail the formulation of the subproblem (LP problems) and master problem (an integer program (IP)) after the BD.

4.2 Classical Optimization for Subproblem

Based on the structure of BD in Section 4.1, the subproblem is defined as follows.

$$\min_{u,w,o,p,q} \sum_{i=1}^{N}\sum_{j=1}^{W}(\sum_{k=1}^{N}a_{1,i,j,k}\cdot o_{k,i,j}+a_{2,i,j}\cdot p_{i,j}+a_{3,i,j}\cdot q_{i,j}) \tag{16}$$

$$\text{s.t.} \quad (13a),(13b),(14a)-(14i).$$

The general form of the subproblem can be further represented as follows.

$$\text{Subproblem:} \quad \min_{\mathbf{Y}} \quad \mathbf{h}^\intercal\mathbf{Y} \tag{17}$$

$$\text{s.t.} \quad -\mathbf{G}\mathbf{Y} \ge \mathbf{X}^\intercal\mathbf{B}\mathbf{X}-\mathbf{a}_3, \tag{17a}$$

$$\mathbf{Y} = [u,w,o,p,q]^\intercal, \quad \mathbf{Y}\in\mathbb{Y}. \tag{17b}$$

In addition, the dual problem of the subproblem is defined below and $\boldsymbol{\pi}$ is the dual variable, i.e.,

$$\max_{\boldsymbol{\pi}} \quad (\mathbf{X}^\intercal\mathbf{B}\mathbf{X}-\mathbf{a}_3)^\intercal\boldsymbol{\pi} \tag{18}$$

$$\text{s.t.} \quad -\mathbf{G}^\intercal\boldsymbol{\pi} \le \mathbf{h}, \tag{18a}$$

$$\boldsymbol{\pi} \ge 0. \tag{18b}$$

This problem can be directly solved by a classical LP solver in a classical computer, e.g. Scipy [42] or Gurobi [43].

4.3 Quantum Formulation for Master Problem

Based on the dual problem of the subproblem, the master problem in a general form can be defined below.

$$\text{Master:} \quad \min_{\mathbf{X}} \quad \mathbf{c}^\intercal\mathbf{X}+\lambda \tag{19}$$

$$\text{s.t.} \quad \mathbf{A}_1\mathbf{X} = \mathbf{a}_1, \tag{19a}$$

$$\mathbf{A}_2\mathbf{X} \le \mathbf{a}_2, \tag{19b}$$

$$\lambda \ge \lambda^{down}, \tag{19c}$$

$$\lambda \ge (\mathbf{X}^\intercal\mathbf{B}\mathbf{X}-\mathbf{a}_3)^\intercal\boldsymbol{\pi}^k, \quad \forall k\in\hat{K}, \tag{19d}$$

$$\mathbf{X} = [x,y]^\intercal, \quad \mathbf{X}\in\mathbb{X}, \tag{19e}$$

where λ is the optimal value of the subproblem at the current iteration. Constraints (19c) is the feasible lower bound of the subproblem and (19d) is the corresponding Benders' cut, where $\hat{K}$ is the stored index set of optimality cuts from the previous iterations.

QUBO Formulation. Quantum annealers are capable of solving optimization problems that are formulated as QUBO. To make use of advanced quantum annealers, the master problem must be transformed into its corresponding QUBO formulation. A QUBO problem commonly involves a vector of binary variables $\mathbf{x}$ and an upper-diagonal matrix $\mathbf{Q}$, a $N' \times N'$ matrix with upper-triangular properties. The objective of QUBO is to minimize the following function:

$$f(\mathbf{x}) = \sum_{i \in N'} \mathbf{Q}_{i,i}\mathbf{x}_i + \sum_{i<j} \mathbf{Q}_{i,j}\mathbf{x}_i\mathbf{x}_j. \tag{20}$$

where $\mathbf{Q}_{i,i}$ is the diagonal terms with linear coefficients and $\mathbf{Q}_{i,j}$ is the nonzero off-diagonal terms with quadratic coefficients. Furthermore, (20) can be expressed as a general form defined below.

$$\min_{\mathbf{x} \in \{0,1\}^{N'}} \mathbf{x}^\intercal \mathbf{Q} \mathbf{x}. \tag{21}$$

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 [44], the constraints are converted as follows,

$$\begin{aligned}
(19a) \Rightarrow \quad & \xi_1 : P^1(\mathbf{A}_1\mathbf{X} - \mathbf{a}_1)^2, \\
(19b) \Rightarrow \quad & \xi_2 : P^2(\mathbf{A}_2\mathbf{X} - \mathbf{a}_2 + \sum_{l=0}^{\bar{l}^2} 2^l s_l^2)^2, \\
& \text{where} \quad \bar{l}^2 = \lceil \log_2(\mathbf{a}_2 - \mathbf{A}_2\mathbf{X}) \rceil. \\
(19c) \Rightarrow \quad & \xi_3 : P^3(\lambda^{down} - \lambda + \sum_{l=0}^{\bar{l}^3} 2^l s_l^3)^2, \\
& \text{where} \quad \bar{l}^3 = \lceil \log_2(\lambda - \lambda^{down}) \rceil. \\
(19d) \Rightarrow \quad & \xi_4 : P^4((\mathbf{X}^\intercal\mathbf{B}\mathbf{X} - \mathbf{a}_3)^\intercal \pi^l - \lambda + \sum_{l=0}^{\bar{l}^4} 2^l s_l^4)^2, \\
& \text{where} \quad \bar{l}^4 = \lceil \log_2[\lambda - \min_{\mathbf{X},\pi}(\mathbf{X}^\intercal\mathbf{B}\mathbf{X} - \mathbf{a}_3)^\intercal \pi^l] \rceil.
\end{aligned}$$

Here, P^* is the predefined penalty vector 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 as

$$\max_{\mathbf{X}} \quad \mathbf{c}^\intercal\mathbf{X} + \lambda + \xi_1 + \xi_2 + \xi_3 + \xi_4. \tag{22}$$

Variable Representation. Problem (22) is still not 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 number $\hat{\lambda} \in \mathbb{Q}$. 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

$$\lambda = \sum_{ii=-\underline{m}}^{\bar{m}_+} 2^{ii}\mathbf{w}_{ii+\underline{m}} - \sum_{jj=0}^{\bar{m}_-} 2^{jj}\mathbf{w}_{jj+1+\underline{m}+\bar{m}_+} = \hat{\lambda}(\mathbf{w}). \tag{23}$$

In (23), $\bar{m}_+ + 1$ 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.

$$\max_{\mathbf{X},\mathbf{w}} \quad \mathbf{c}^\intercal\mathbf{X} + \hat{\lambda}(\mathbf{w}) + \xi_1 + \xi_2 + \xi_3 + \xi_4. \tag{24}$$

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

Input: Distributed network with N servers V, W FL models M, coefficient of the objective function and constraints in master problem and subproblem
Output: All decision variables $\mathbf{X}$ and $\mathbf{Y}$

1: Initialize upper/lower bound of λ, $\overline{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
2: Initialize threshold $\epsilon = 0.001$, $max\_itr = 100$, $itr = 1$
3: **while** $|\overline{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max\_itr$ **do**
4: $\quad$ $\mathbf{P} \leftarrow$ Appropriate penalty numbers or arrays
5: $\quad$ $\mathbf{Q} \leftarrow$ Reformulate both objective and constraints in (14) and construct QUBO formulation as (24)
6: $\quad$ $\mathbf{X}' \leftarrow$ Solve problem (24) by quantum computer
7: $\quad$ $\underline{\lambda} \leftarrow$ Extract $\mathbf{w}$ and replace $\underline{\lambda}$ with $\hat{\lambda}(\mathbf{w})$ as (23)
8: $\quad$ $SUP(\mathbf{X}) \leftarrow$ Solve problem (18) with fixed $\mathbf{X}'$
9: $\quad$ $\overline{\lambda} \leftarrow SUP(\mathbf{X})$
10: $\quad$ Add a Benders' cut to the master problem as (19d)
11: $\quad$ $itr+ = 1$
12: **end while**
13: **return X, Y**

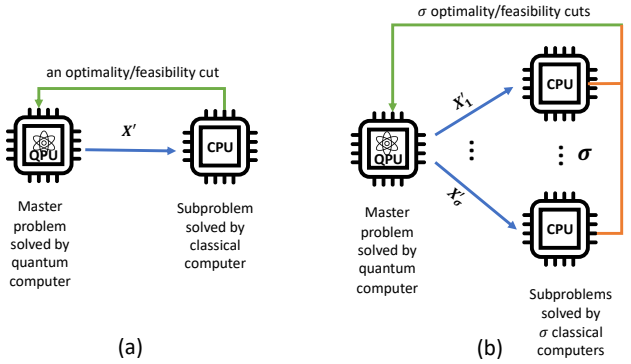


Fig. 3: Flow of HQCBD: (a) single cut vs (b) multi cuts.

4.4 HQCBD Algorithm

Our proposed HQCBD is described by Algorithm 1. Fig. 2 shows the overall flow of HQCBD, while Fig. 3(a) shows the detailed interaction between the master problem and subproblem. The master problem is solved by a quantum computer and generates a binary solution ($\mathbf{X}'$), 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 bounds 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 (14) in the QUBO format and solve the QUBO problem with a quantum annealer and update the lower bound of the problem $\underline{\lambda}$ (Lines 5-7). Given $\mathbf{X}'$ from the master problem, we solve the subproblem (18) and update the upper bound of the problem $\overline{\lambda}$ (Lines 8-9). We finally add the Benders' cut

to the master problem and continue the next iteration (Lines 10-11) until it converges (Line 3).

We use the quantum annealer to execute our proposed algorithm for solving the QUBO master problem. Moreover, it is crucial to adjust the penalties appropriately for a suitable QUBO model. Typically, a high penalty may cause a coefficient explosion, which could result in the malfunctioning of the quantum annealer. Conversely, a low penalty might cause the quantum annealer to ignore the constraints. Here, given the lower and upper bound of a penalty, we leverage the binary search method to iteratively determine a well-tuned penalty for each constraint.

4.5 Multiple Cuts Version

In Algorithm 1 (Line 11), we only consider one single Benders' cut imported to the master problem in each round. This cut is computed from the subproblem based on an optimal feasible solution ($\mathbf{X}'$) returned by the quantum annealer (Line 6 of Algorithm 1). However, one of the advances of the quantum algorithm is that it can generate multiple feasible solutions simultaneously. Therefore, to accelerate the convergence of the master problem, we further introduce a hybrid quantum-classical multiple-cuts optimization method. In the multiple-cuts version of the HQCBD algorithm (MBD), we leverage the multiple feasible solutions generated by the quantum annealer and select the top σ feasible solutions to further generate multiple cuts. Then multiple cuts are inserted into the master problem per iteration. Fig. 3(b) illustrates this idea.

The detailed MBD algorithm is given by Algorithm 2. Compared with the single-cut version of the HQCBD algorithm, first, these top σ feasible solutions are sent to σ subproblems and all subproblems execute in parallel (Lines 6 and 8). Second, each subproblem generates a Benders' cut and sends it back to the master problem (Line 9). Finally, the master problem collects all Benders' cuts, adds to the constraints (Line 10), and continues the next iteration. Note that if one of these subproblems reaches the threshold, the iteration will be stopped since the upper bound and lower bound converge to the predefined threshold.

Algorithm 2 Multiple-cuts Benders' Decomposition (MBD)

Input: Distributed network with N servers V, W FL models M, coefficient of the objective function and constraints in master problem and subproblem, number of cuts σ
Output: All decision variables **X** and **Y**

1: Initialize upper/lower bound of λ, $\overline{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
2: Initialize threshold $\epsilon = 0.001$, $max\_itr = 100$, $itr = 1$
3: **while** $|\overline{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max\_itr$ **do**
4: $\quad$ **P** $\leftarrow$ Appropriate penalty numbers or arrays
5: $\quad$ **Q** $\leftarrow$ Reformulate both objective and constraints in (14) and construct QUBO formulation as (24)
6: $\quad$ $\{\mathbf{X}'\}_\sigma \leftarrow$ Solve problem (24) by quantum computer and return σ feasible solutions
7: $\quad$ $\underline{\lambda} \leftarrow$ Extract **w** with highest value and replace $\underline{\lambda}$ with $\tilde{\lambda}(\mathbf{w})$ as (23)
8: $\quad$ $\{SUP(\mathbf{X})\}_\sigma \leftarrow$ Solve σ subproblems (18) with fixed $\mathbf{X}'$ in parallel
9: $\quad$ $\overline{\lambda} \leftarrow \{SUP(\mathbf{X})\}_\sigma$ with lowest value
10: $\quad$ Add all σ benders' cut to the master problem as (19d)
11: $\quad$ $itr += 1$
12: **end while**
13: **return X**, **Y**

5 Performance Evaluation

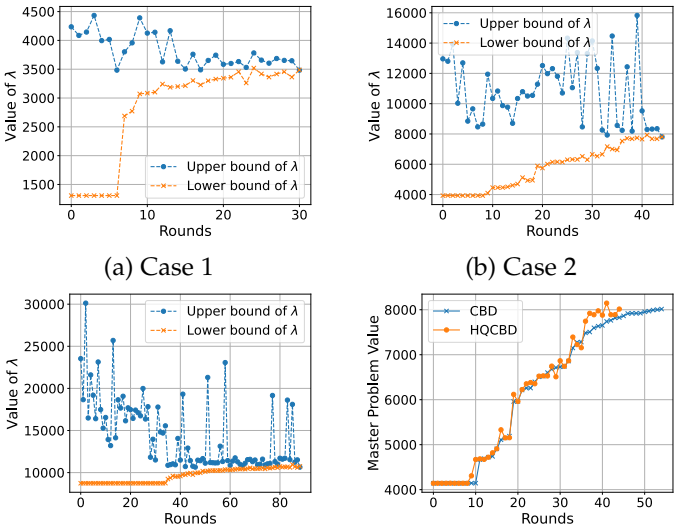
In this section, we simulated a distributed network environment and conducted experiments of realistic FL tasks using publicly available datasets. To confirm the practicality of our hybrid quantum-classical optimization algorithm, we implemented the proposed algorithms on a hybrid D-Wave quantum processing unit (QPU). We utilized the D-Wave system that is available via the Leap quantum cloud service [45]. Based on the Pegasus topology, the D-Wave system also has over 5k qubits and 35k couplers, which can solve complex problems of up to 1M variables and 100k 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.

5.1 Simulation Setup

Network Setting: Our distributed computing environment consists of 100 servers where the topology depends on the real-world EUA-Dataset [46] and the Internet topology zoo [47]. EUA-Dataset, which contains the geographical locations of 125 cellular base stations in the Melbourne CBD area, is a widely-used dataset in mobile computing, 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 sc_i, CPU frequency sf_i and link bandwidth b_j belonging to the ranges of $1,024-2,048$GB, $2-5$GHz, and $512-1,024$Mbps, respectively.

Datasets and FL models: We conduct extensive experiments on the following real-world datasets: California Housing dataset [48], MNIST [49], Fashion-MNIST (FMNIST) [50], and CIFAR-10 [51]. 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 datasets for performance evaluation: (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. Furthermore, Each FL task has a specific model size μ_j and CPU requirement χ_j in ranges of $10-100$MB and $1-3$GHz, respectively. The global convergence requirement and the two constant variables are set based on [12]: $\varsigma_j = 0.001$, $\vartheta_0 = 15$ and $\varphi_0 = 4$.

Benchmarks and Metrics: We compare our proposed HQCBD and MBD algorithms 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 [43] or Scipy [42]) to solve the master problem and subproblems. RAND randomly generates the random decisions on the



(a) Case 1 (b) Case 2

(c) Case 3 (d) Master problem value

Fig. 4: Performance of HQCBD: its convergence.

model's parameter server, FL workers, and local convergence rate under certain constraints. TWSO is a previous algorithm [15] that decomposes the original problem into two subproblems (participant selection and learning scheduling) and solves them iteratively. 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*, *the solver accessing time* and *the gain or advancement* of our proposed algorithms over CBD.

5.2 Simulation Results

5.2.1 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 1, 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. 4 and Table 1 show the related results of their performances.

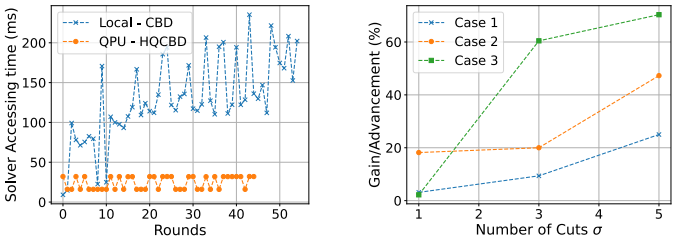
In Figs. 4(a)-(c), the blue dashed line denotes the upper bound of value λ used in HQCBD, and the orange dashed line denotes the lower bound of λ in HQCBD. As we can see, the upper bound and lower bound finally converge and we obtain the non-negative lower bound at 31st, 45th, and 89th round for each case, respectively. This result proves that our proposed algorithm is mathematically consistent with the classical BD algorithm. Fig. 4(d) shows the trend of the master problem value of case 2 calculated by (22) 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 1: Iteration comparison of CBD and HQCBD over three different cases. 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

TABLE 2: Solver accessing time (ms) comparison of CBD and HQCBD.

Case	CBD		HQCBD	
	Max. / Min.	Avg. / Std.	Max. / Min.	Avg. / Std.
1	190.5 / 6.7	117.1 / 50.1	32.1 / 15.9	31.5 / 2.8
2	235.3 / 9.1	129.6 / 50.1	32.1 / 15.9	24.2 / 7.9
3	395.5 / 14.5	120.3 / 63.2	32.1 / 16.1	25.5 / 8.0



(a) CBD vs HQCBD (b) MBD gains over CBD

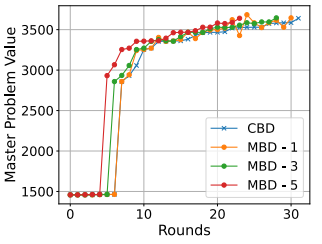
Fig. 5: (a) Comparison of the real solver accessing time of CBD and HQCBD in Case 2. (b) Gains of MBD over CBD with the different number of cuts σ in different cases.

Table 1 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%).

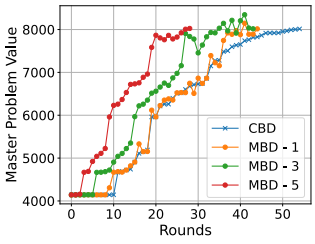
Moreover, we show the comparison of real solver accessing time (i.e., the computation time of the solvers) for CBD and HQCBD in Table 2 and plot the detailed accessing time of Case 2 in Fig. 5. The solver accessing time is the real accessing time of the QPU solver and local solver without considering other overheads, such as variables setting time, parameters transmission time, and so on. As we can see in Table 2, 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 are 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. 5(a) 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.

TABLE 3: Iteration of CBD and MBD with different σ.

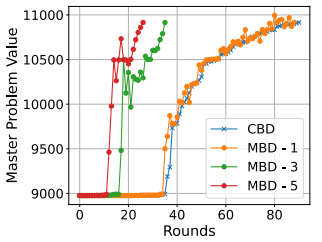
Case	# of Binary var.	Itr. of CBD	Itr. of MBD ($\sigma = 1/3/5$)
1	63	32	31 / 29 / 24
2	126	55	45 / 44 / 29
3	198	91	89 / 36 / 27



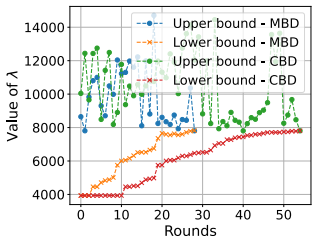
(a) Case 1



(b) Case 2



(c) Case 3



(d) Convergence comparison

Fig. 6: Performance of MBD: its convergence.

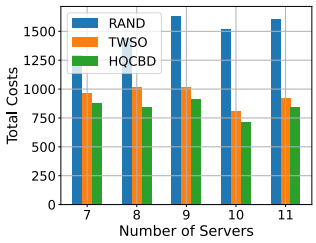
5.2.2 *Performance of MBD*

We now evaluate the efficiency of our proposed MBD algorithm. Similarly, we consider three different cases with different numbers of servers, FL models, and workers. We study the impact of the number of cuts σ used in MBD and we select the value from 1, 3, and 5. Recall that when $\sigma = 1$, MBD is our standard HQCBD. Table 3 and Fig. 6 show the result of multiple cuts and convergence comparison with CBD. In Fig. 6(a)-(c), MBD-1 is our proposed HQCBD algorithm where only a single cut is added to the master problem, while MBD-3 or MBD-5 means 3 or 5 cuts are added to the master problem. In this scenario, we can find that our MBD-1 (HQCBD) converges faster than the CBD. But with more cuts (larger σ), the convergence speed of MBD-σ becomes faster. Table 3 lists the detailed comparison between CBD and MBD for different cases. Fig. 6(d) further demonstrates the upper and lower bound detailed convergence comparison between our proposed algorithm MBD with $\sigma = 5$ and the CBD in Case 2. We can see that our proposed methods use fewer rounds (29) to converge the optimal value compared with the classical one (55).

We also plot the gain or advancement of MBD over CBD in terms of iteration reduction for different numbers of cuts in Fig. 5(b). Obviously, different numbers of cuts have achieved different positive gains or advancements in different cases. The largest improvement is up to 70.3% for Case 3 with $\sigma = 5$. This further proves the efficiency of both proposed algorithms HQCBD and MBD.

5.2.3 *Comparison with Existing Methods*

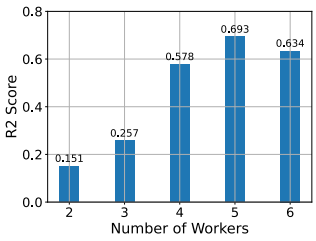
We now compare our proposed method HQCBD with the random method (RAND) and a two-stage iterative optimization method (TWSO) [15] in terms of solving the joint optimization problem.



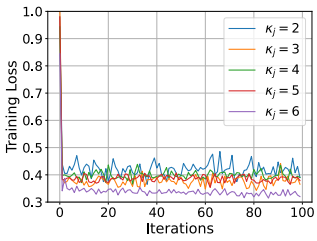
(a) Impact of server number

(b) Impact of worker number

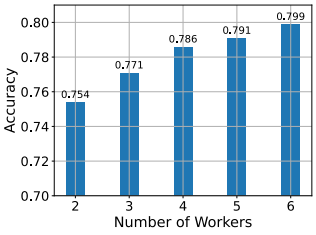
Fig. 7: Performance comparison with existing methods: with different numbers of servers or FL workers.



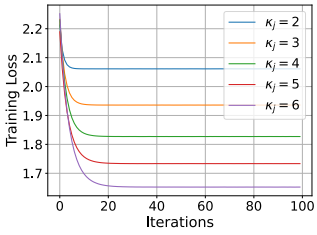
(a) R2 score



(b) LR over CA housing



(c) Accuracy

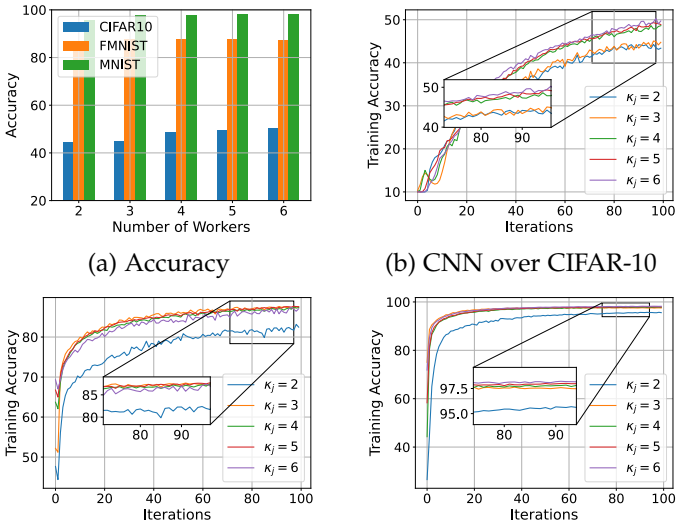


(d) LR over MNIST

Fig. 8: 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.

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. 7(a) shows the results. Obviously, RAND has the worst performance due to its randomness. Our HQCBD algorithm gets further improvements compared with our proposed 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 examine how varying numbers of FL workers affect the 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. 7(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. With more qubits supporting, we expect that the speed of HQCBD will have a more significant advantage over TWSO on large-scale optimization problems.



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

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

Fig. 9: Training accuracy of CNN models (non-convex) and impact from the number of workers: (a) accuracy of all CNN models; (b)-(d) accuracy convergence with different number of workers over different datasets.

5.2.4 *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). In Fig. 8(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. 8(b). Similarly, the accuracy of the logistic regression model also increases with more FL workers while the training loss declines with the rise of the number of workers as shown in Figs. 8(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. Figs. 9(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. 9(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. 9(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.

6 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. In order to accelerate the convergence speed of our proposed algorithm, we further introduced a multiple-cuts version of HQCBD (MBD) to hasten the solving process. Extensive simulations on the D-Wave quantum annealing machine demonstrated the efficiency and robustness of our proposed HQCBD and MBD algorithms which not only achieved the same result as the classical algorithm but also took much fewer iterations (up to 70.3% improvement) and less accessing time (up to 81% reduction) to obtain the desired solution even at relevantly small scales.

This work is our first attempt toward quantum-assisted optimization for distributed computing. There are a few possible directions for further study. (1) We plan to further study and improve the performance of the proposed HQCBD with more diverse FL datasets and applications. (2) We plan to extend our general hybrid quantum-classical optimization framework so that it can be applied to more types of joint optimization problems (beyond FL) in distributed systems. (3) We are also interested in designing a new hybrid quantum-classical optimization framework on other types of quantum machines, e.g., gate-based quantum computers. With the new development of robust and diverse quantum computers with more available qubits, we believe the hybrid quantum-classical optimization can play a more important role in the next generation distributed intelligent systems.

References

[1] F. Arute, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, v.574,no.7779,pp.505-510, 2019.

[2] H.-S. Zhong, *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 Trans. 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 Info. Process.*, vol. 21, no. 2, pp. 67, 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 *Proc. of AISTATS*, Ft. Lauderdale, 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, et al., "Robust and communication-efficient federated learning from non-iid data," *IEEE Trans. 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 *Proc. of IEEE NAS*, Philadelphia, PA, Oct. 2022.

[10] J. Liu, X. Wei, X. Liu, H. Gao, and Y. Wang, "Group-based hierarchical federated learning: convergence, group formation, and sampling," in *Proc. of ICPP*, 2023.

[11] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. of IEEE ICC*, Shanghai, China, May 2019.

[12] Y. Jin, et al., "Learning for learning: Predictive online control of federated learning with edge provisioning," in *Proc. of IEEE INFOCOM*, Virtual, May 2021.

[13] S. Wang, et al., "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. 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*, vol. 22, no. 9, pp. 5449–5462.

[15] X. Wei, J. Liu, and Y. Wang, "Joint participant selection and learning scheduling for multi-model federated edge learning," in *Proc. of IEEE MASS*, Denver, CO, Oct. 2022.

[16] T. Tran, et al., "A hybrid quantum-classical approach to solving scheduling problems," in *Proc. of Int'l Symp. on Combinatorial Search*, New York, Jul. 2016.

[17] A. Ajagekar, et al., "Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems," *Computers & Chemical Eng.*, vol. 132, pp. 106630, Jan. 2020.

[18] Z. Zhao, L. Fan, and Z. Han, "Hybrid quantum Benders' decomposition for mixed-integer linear programming," in *Proc. of IEEE 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 Trans. 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 Net.*,vol.36,no.5,pp.72-76, 2022.

[21] X. Wei, et al., "Quantum assisted scheduling algorithm for federated learning in distributed networks," in *Proc. of ICCCN*, 2023.

[22] Z. Yang, et al., "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, Nov. 2020.

[23] L. Li, et al., "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *Proc. of IEEE INFOCOM*, May 2021.

[24] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. of IEEE ICC*, Jun. 2020.

[25] S. Luo, et al., "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. on Wireless Commu.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[26] Z. Wang, et al., "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. INFOCOM*, 2021.

[27] Z. Meng, et al., "Learning-driven decentralized machine learning in resource-constrained wireless edge computing," in *Proc. of IEEE INFOCOM*, Virtual, May 2021.

[28] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, Oct. 2020.

[29] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *arXiv preprint arXiv:2007.15197*, Oct. 2020.

[30] O. Marnissi, et al., "Client selection in federated learning based on gradients importance," *arXiv preprint arXiv:2111.11204*, Nov. 2021.

[31] F. Lai, et al., "Oort: Efficient federated learning via guided participant selection." in *Proc. of USENIX OSDI*, Virtual, Jul. 2021.

[32] R. Balakrishnan, et al., "Diverse client selection for federated learning via submodular maximization," in *Proc. of ICLR*, 2022.

[33] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, pp. 79, Aug. 2018.

[34] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, Dec. 1992.

[35] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. of ACM STOC*, New York, NY, May 1996.

[36] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, p. 1484–1509, Oct. 1997.

[37] N. G. Paterakis, "Hybrid quantum-classical multi-cut benders approach with a power system application," *arXiv preprint arXiv:2112.05643*, Dec. 2021.

[38] Y. Jin, et al., "Resource-efficient and convergence-preserving online participant selection in federated learning," in *Proc. of IEEE ICDCS*, Singapore, Feb. 2020.

[39] M. Chen, et al., "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. on Wireless Communications*, vol. 20, no. 1, pp. 269–283, Oct. 2020.

[40] C. Ma, et al., "Distributed optimization with arbitrary local solvers," *Optimi. Meth. & Software*, vol.32, no.4, pp.813-848, 2017.

[41] R. Rahmaniani, et al., "The benders decomposition algorithm: A literature review," *European J. of Operational Research*, vol.259, no.3, pp.801-817, Jun 2017.

[42] P. Virtanen, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Meth.*, vol.17, pp.261-272, 2020.

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

[44] 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.

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

[46] P. Lai, et al., "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *International Conference on Service-Oriented Computing*, Hangzhou, China, Nov. 2018.

[47] S. Knight, et al., "The internet topology zoo," *IEEE J. on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[48] F. Pedregosa, et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[49] Y. LeCun, et al., "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp.2278-2324, 1998.

[50] 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.

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

Xinliang Wei (S'21-M'23) holds a Ph.D. in Computer and Information Sciences from Temple University, Philadelphia, USA in 2023. He received his M.S. and B.E. degrees both in Software Engineering from SUN Yat-sen University, Guangzhou, China in 2016 and 2014, respectively. His research interests include edge computing, federated learning, reinforcement learning, and Internet of Things. He is a recipient of *Outstanding Research Assistant* from College of Science and Technology (2022) and *Scott Hibbs Future of Computing Award* from Department of Computer & Information Sciences (2023) at Temple University. He is currently an Assistant Professor in Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences.

Lei Fan (M'15-SM'20) is an Assistant Professor in the Department of Engineering Technology as well as in the Department of Electrical and Computer Engineering at University of Houston. Before this position, he worked in the electricity energy industry for several years. He received the Ph.D. degree in operations research from the Industrial and System Engineering Department at University of Florida. His research includes quantum computing, optimization methods, complex system operations, power system operations and planning.

Yuanxiong Guo (M'14-SM'19) received the B.Eng. degree in electronics and information engineering from the Huazhong University of Science and Technology, China, in 2009, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Florida, in 2012 and 2014, respectively. He is currently an Associate Professor in the Department of Information Systems and Cyber Security at the University of Texas at San Antonio. His current research interests include distributed machine learning, applied data science, and trustworthy AI with application to digital health, energy sustainability, and human-robot collaboration. He is on the Editorial Board of IEEE Transactions on Vehicular Technology and servers as the track co-chair for IEEE VTC 2021-Fall. He is a recipient of the Best Paper Award in the IEEE GLBOECOM 2011.

Yanmin Gong (SM, IEEE) received the B.Eng. degree in electronics and information engineering from Huazhong University of Science and Technology in 2009, the M.S. degree in electrical engineering from Tsinghua University in 2012, and the Ph.D. degree in electrical and computer engineering from University of Florida in 2016. She is currently an Associate Professor of Electrical and Computer Engineering with the University of Texas at San Antonio. Her research interests lie at the intersection of machine learning, cybersecurity, and networking systems. Dr. Gong is a recipient of the NSF CAREER Award, the NSF CRII Award, the IEEE Computer Society TCSC Early Career Researchers Award for Excellence in Scalable Computing, the Rising Star in Networking and Communications Award by IEEE ComSoc N2Women, and the Best Paper Award at IEEE GLOBECOM. She is currently an Editor of the ACM COMPUTING SURVEYS and IEEE WIRELESS COMMUNICATIONS.

Zhu Han (S'01-M'04-SM'09-F'14) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. Currently, he is a John and Rebecca Moores Professor in the Electrical and Computer Engineering Department as well as in the Computer Science Department at the University of Houston. His research targets on the novel game-theory related concepts critical to enabling efficient and distributive use of wireless networks with limited resources. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, and the 2021 IEEE Kiyo Tomiyasu Award. He was an IEEE Communications Society Distinguished Lecturer from 2015-2018, AAAS fellow since 2019, and ACM fellow since 2024. He is a 1% highly cited researcher since 2017 according to Web of Science.

Yu Wang (S'02-M'04-SM'10-F'18) is a Professor in the Department of Computer and Information Sciences at Temple University. He holds a Ph.D. from Illinois Institute of Technology, an MEng and a BEng from Tsinghua University, all in Computer Science. His research interest includes wireless networks, smart sensing, and distributed computing. He has published over 300 papers in peer reviewed journals and conferences. He is a recipient of *Ralph E. Powe Junior Faculty Enhancement Awards* from Oak Ridge Associated Universities (2006), *Outstanding Faculty Research Award* from College of Computing and Informatics at the University of North Carolina at Charlotte (2008), *Fellow of IEEE* (2018), *ACM Distinguished Member* (2020), and *IEEE Benjamin Franklin Key Award* (2024). He has served as Associate Editor for *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, among others.