Hybrid Quantum-Classical Graph Neural Networks for Solving Unit Commitment Problem

Xuanzi Ma[1], Junchao Cui[1], Wenxia Wang[1], Yizhen Huang[1], Bei Zhou[1], Jinchen Xu[1*]
[1]School of Cyberspace Security, Information Engineering University, Zhengzhou 450001, China

***Abstract*—The Unit Commitment Problem (UCP) is a core optimization task for ensuring secure and economically efficient operation of power grids. However, existing methodologies face two major limitations: first, there is a lack of a general and efficient way to represent the complex relationships between UCP variables and constraints; second, exponential parameter growth with problem scale impedes characterization of complex dependencies in UCP, thereby limiting predictive accuracy. To address these issues, we propose a hybrid quantum-classical graph neural network framework (HQGNN) that establishes a novel paradigm for UCP by: (1) modeling UCP as a unified bipartite graph format, and (2) integrating quantum feature extraction modules to reduce the parameter size and enhance classical GNNs' capability in capturing high-dimensional structural characteristics. Extensive experiments on IEEE and RTS benchmark datasets demonstrate that HQGNN achieves improvements in UCP solution accuracy while validating the effectiveness of quantum modules in enhancing graph-structured feature learning. The quantum module in HQGNN is proven to be the key enabler of performance gains. Its integration cuts parameters by 16.8% and reduces mean squared error (MSE) in optimal solution prediction by 24.4% across all benchmarks, while achieving full-task superiority on the IEEE-9 dataset.These results substantiate the proposed method's capacity to simultaneously enhance solution quality and computational efficiency through quantum-enhanced graph representation learning.**

***Index Terms*—Unit Commitment Problem, Hybrid Quantum-Classical Framework, Quantum Graph Neural Networks**

I. Introduction

The Unit Commitment Problem (UCP) is a fundamental optimization task in power system operations. It entails determining the on/off commitment states and output schedules of generators, with the objective of minimizing operational costs while satisfying a set of technical and operational constraints, such as load balance, ramping limits, and minimum uptime/downtime requirements [1], [2]. This problem holds dual significance: theoretically, it is a canonical NP-hard mixed-integer linear programming (MILP) problem. It has high dimensionality, nonconvexity, and nondifferentiability, and encompasses both continuous variables (generator outputs) and discrete variables (commitment states); Practically, traditional solvers exhibit exponential computational complexity growth with increasing generator numbers and time horizons, necessitating efficient methodologies for large-scale systems and long-term planning scenarios.

With escalating power demand, UCP complexity and scale have substantially increased, driving continuous evolution of solution methodologies [3], [4]. Emerging data-driven approaches learn implicit dispatch patterns from historical records, circumventing complex mathematical programming while accelerating computation and improving predictive accuracy. These can be categorized into two paradigms as follows.

The first paradigm employs traditional machine learning models such as Random Forest (RF) and Support Vector Machines (SVM) [5]–[7]. These methods utilize diversified technical pathways to predict unit commitment states, thereby reducing subsequent search space. Zhang et al. [8] developed an RF-based ensemble approach predicting UCP states from historical MILP solutions, achieving significant acceleration. Schmitt et al. [9] applied RF to predict binary commitment decisions in network-constrained UC, using predictions as warm-start inputs for accelerated solving. Pan et al. [10] constructed a heterogeneous ensemble model combining KNN, SVM, and CART for sub-hourly UC, fixing partial binary variables to substantially reduce MILP problem scale while maintaining near-optimal solutions. However, these approaches lack a unified UCP representation framework, relying heavily on manually engineered heterogeneous features (e.g., startup costs, load profiles). Despite efficiency improvements through ensemble or improved models, no standardized modeling paradigm exists, limiting adaptability across varying scales and constraint conditions while failing to capture complex variable-constraint interactions.

The second paradigm utilizes graph neural networks (GNNs) to explicitly model grid topology and operational constraints [11], [12]. Conventional approaches represent generators and buses as nodes, transmission lines as edges, employing graph convolutional networks (GCN) or graph attention networks (GAT) to capture spatial interdependencies [13]. Ostrowski et al. [14] applied GAT with attention mechanisms to weight adjacent nodes, achieving superior accuracy over LSTM in commitment state prediction. Yang et al. [11] used GNN-predicted high-quality initial solutions to significantly reduce search spaces.

Despite progress in solution acceleration, both paradigms face two critical challenges: (1) Absence of universal graph representation frameworks for UCP. Traditional ML methods relies on manual feature engineering while existing GNNs, though embedding grid topology, lack standardized modeling paradigms; (2) Inadequate characterization of complex variable-constraint relationships with exponential parameter growth impeding high-order dependency modeling in power system graphs. Increasing GNN depth/width risks parameter inflation, training overhead, and overfitting, severely limiting accuracy and scalability in large-scale, constraint-intensive scenarios.

To address these limitations, we propose a hybrid quantum-

*Corresponding author:atao728208@126.com

classical graph neural network (HQGNN) that: (1) establishes a universal bipartite graph modeling paradigm for UCP; and (2) integrates quantum modules into classical GNNs to suppress exponential parameter growth while enhancing feature extraction capabilities from UCP bipartite graphs. Our key contributions include:

- Development of the HQGNN framework resolving UCP's complex relationship characterization and parameter inflation challenges. Quantum feature extraction modules enhance classical GNNs' capacity to learn high-order dependencies in power system graphs, overcoming traditional GNN expressiveness limitations while avoiding parameter inflation and training overhead associated with increased depth/width.
- Introduction of a standardized bipartite graph representation paradigm for UCP, eliminating reliance on manual feature engineering. This paradigm systematically models UCP's generator parameters and variable-constraint relationships within bipartite graph structures, enhancing model generalization and transferability.
- Comprehensive experiments validate HQGNN's superiority. On IEEE and RTS benchmarks, HQGNN outperforms classical and quantum-graph baselines in UCP accuracy. Its quantum module reduces parameters by 16.8% and optimal solution prediction MSE by 24.4% across benchmarks, with full-task superiority on IEEE-9. These results confirm the quantum module's key role in graph feature learning and HQGNN's efficiency for constraint-heavy UCP tasks.

II. PRELIMINARIES

This section focuses on the fundamental formulation of UCP based on MILP. We elaborate on the variable definitions, objective function, and constraints of the model, which establishes a rigorous mathematical basis for the subsequent bipartite graph conversion in our proposed framework.

The UCP objective function typically minimizes the total system operational cost, which primarily consists of generation cost, startup cost, and shutdown cost. Assuming generators $i \in I$ and time periods $t \in T$, the final UCP objective function is formulated as:

$$min \sum_{i \in I} \sum_{t \in T} \left(C_i^{on} y_{i,t} + C_i^{off} z_{i,t} + C_i^{fuel} p_{i,t} \right) \tag{1}$$

where C_i^{on} denotes generator i's startup cost, C_i^{off} represents generator i's shutdown cost, C_i^{fuel} is the fuel cost coefficient for generator i , and $p_{i,t}$ indicates the output power of generator i during time interval t . Binary variables $y_{i,t}$ and $z_{i,t}$ mark startup and shutdown actions for generator i at time t . The following constraints are introduced:

- Power Balance Constraint: The total generator output must satisfy system load demand D_t at each time interval:

$$\sum_{i=1}^{I} p_{i,t} = D_t, \forall t \in T \tag{2}$$

- Generator Output Limits: Each generator's power output must remain between its minimum and maximum technical capacities:

$$u_{i,t} \cdot P_i^{min} \leq p_{i,t} \leq u_{i,t} \cdot P_i^{max}, \forall i, t \tag{3}$$

where P_i^{min} and P_i^{max} denote generator i 's minimum and maximum technical outputs, and $u_{i,t} \in \{0, 1\}$ represents the operational status of generator i during period t .

- Ramp Rate Constraints: Power adjustments between consecutive periods are limited by ramping rates:

$$|p_{i,t} - p_{i,t-1}| \leq \Delta P_i, \forall i, t > 1 \tag{4}$$

- Minimum On/Off Time Constraints: Linearized operational constraints require minimum continuous operation or shutdown durations:

$$\begin{cases} y_{i,t} \geq u_{i,t} - u_{i,t-1} \\ z_{i,t} \geq u_{i,t-1} - u_{i,t} \\ y_{i,t} + z_{i,t} \leq 1 \end{cases}, t \geq 2 \tag{5}$$

where $y_{i,t} = 1$ denotes the startup action of generator i at time t, $z_{i,t} = 1$ denotes the shutdown action, and $y_{i,t} + z_{i,t} \leq 1$ ensures that a generator cannot start and shut down simultaneously.

III. METHOD

This section presents the overall architecture and implementation details of key components in the HQGNN, including the UCP bipartite graph representation methodology, the classical graph embedding module, and the quantum feature extraction module. Subsequently, the design of the output layer tailored for UCP solving is elaborated. HQGNN takes as input the bipartite graph representation of UCP and the corresponding set of labeled tuples, and generates output results of varying dimensions depending on the task requirements.

The overall architecture of HQGNN is illustrated in Fig. 1(a), comprising four sequential components: (1) bipartite graph input layer, (2) classical graph embedding layer, (3) quantum feature extraction module, and (4) output module. The bipartite graph representation methodology for UCP is introduced in Section III-A, the implementation of the classical graph embedding layer is detailed in Section III-B, while the design of the quantum feature extraction module is presented in Section III-C, and the implementation of the output layer is described in Section III-D.

A. Bipartite graph format of UCP

We model the UCP as a bipartite graph. Based on the previous chapter, we have provided the MILP modeling for UCP (Fig. 1(b)), the core basis for UCP-to-bipartite graph conversion. We then transform it into an MILP-graph structure, retaining all core UCP constraints (integer/continuous variables, linear constraints) and complex variable-constraint associations. Using the method in [16] for converting MILPs to bipartite graphs, the construction workflow is as follows.

Each UCP bipartite graph sample $G = (V \cup C, E)$ consists of variable nodes $V = \{v_1, v_2, \ldots, v_n\}$ and constraint nodes

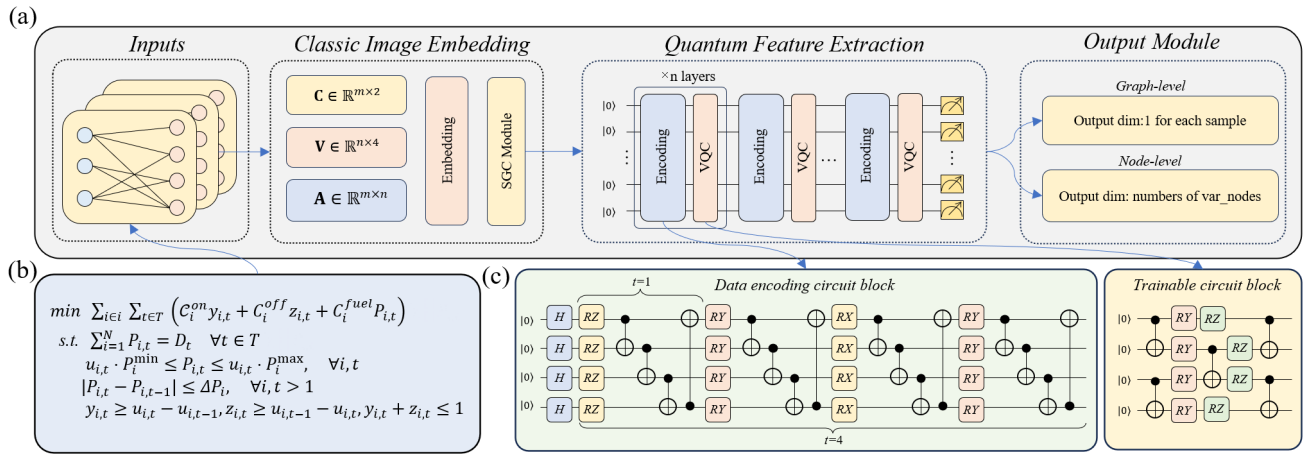

Fig. 1. Architectural overview of HQGNN. (a) Overall architectural structure of the model; (b) MILP formulation of UCP; (c) Circuit implementations of the two key blocks in the quantum feature extraction module.

$C = \{c_1, c_2, \ldots, c_m\}$. Variable nodes and constraint nodes carry their own feature vectors: the features of variable nodes include the lower and upper bounds of decision variables, objective function coefficients, etc.; the features of constraint nodes include constraint types, constant terms, etc. The edge set $E \subseteq V \times C$ represents the association relationship between variables and constraints, and the weight w_{ij} of each edge is determined by the coefficient of the variable in the constraint. These features together form the multi-dimensional attribute representation of the input graph.

First, we model integer variables and continuous variables as variable nodes. It is known that there are four kinds of variable nodes in UCP: three integer variables $u_{i,t}$, $y_{i,t}$, $z_{i,t}$ ($u_{i,t}, y_{i,t}, z_{i,t} \in \{0, 1\}$) and one continuous variable $p_{i,t}$ ($p_{i,t} \geq 0$): $u_{i,t}$ represents the operating state of unit i in time period t (1 means operation, and 0 means shutdown), $y_{i,t}$ and $z_{i,t}$ mark the start-stop actions of unit i in time period t, $p_{i,t}$ represents the actual output of unit i in time period t. We construct the feature of the variable node as $h_j^W = (c_j, l_j, a_j, \tau_j)$, where c_j is the objective function coefficient (fuel cost coefficient C_i^{fuel} or start up cost C_i^{on}), l_j and a_j are the lower and upper bounds of the variable, and τ_j is the variable type (1 for integer variable, 0 for continuous variable). Furthermore, the feature vectors of variable nodes are given as follows:

$$\begin{cases} p_{i,t} & = & \left(c_i, P_i^{\min}, P_i^{\max}, 0\right) \\ u_{i,t} & = & (0, 0, 1, 1) \\ y_{i,t} & = & \left(C_i^{\text{on}}, 0, 1, 1\right) \\ z_{i,t} & = & \left(C_i^{\text{off}}, 0, 1, 1\right) \end{cases} \tag{6}$$

Second, we transform constraints into constraint nodes. In this paper, we focus on key constraints of UCP, including the power balance constraint, ramping constraint, and start-stop time constraint of UCP. Based on this, the feature of the constraint node corresponding to each constraint is constructed as $h_i^V = (b_i, \circ_i)$, where b_i represents the right-hand side of each constraint (such as load D_t or ramping rate ΔP_t), and $\circ_i$ represents the constraint type ($\leq, =, \geq$).

Then, we construct the edge connections in the UCP bipartite graph. In the UCP bipartite graph, edges represent whether a variable appears in a constraint. Therefore, edges are only established between constraint nodes and variable nodes, with the edge weight corresponding to the coefficient of the variable within that constraint. For example, in the power balance constraint, the coefficient of variable $p_{i,t}$ is 1, so the weight of this edge is 1. The construction of each constraint node and its associated edges is shown in Table 1, where the values in the "Edge" column $(v_{i,t}, w)$ indicate that the constraint node is connected to the variable node $v_{i,t}$ with edge weight w.

TABLE I
CONSTRUCTION OF CONSTRAINT NODES AND EDGE CONNECTIONS.

ID	Constraint	Feature	Edge
1	Power Balance	$(D_t, 0)$	$(p_{i,t}, 1)$
2	Lower Limit	$(0, 2)$	$(p_{i,t}, 1)(u_{i,t}, -P_i^{\min})$
3	Upper Limit	$(0, 1)$	$(p_{i,t}, 1)(u_{i,t}, -P_i^{\max})$
4	Ramping Speed	$(\Delta P_i, 1)$	$(p_{i,t}, 1)(p_{i,t-1}, -1)$
5	Startup Logic	$(0, 2)$	$(y_{i,t}, 1)(u_{i,t}, -1)(u_{i,t-1}, 1)$
6	Startup Logic	$(0, 2)$	$(y_{i,t}, 1)(u_{i,t}, 1)(u_{i,t-1}, -1)$
7	Startup Logic	$(1, 1)$	$(z_{i,t}, 1)(y_{i,t}, 1)$

B. Classical Graph Embedding Module

The bipartite graph is processed by a classical embedding layer to generate low-dimensional node embeddings, which are then adapted for quantum encoding.

We extract three features: constraint node matrix $\mathbf{C} \in \mathbb{R}^{m \times 2}$, variable node matrix $\mathbf{V} \in \mathbb{R}^{n \times 4}$, and adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where m and n denote the number of constraint nodes and variable node. These are processed by Simplified Graph Convolution (SGC) [17] with $k = 2$ to capture local topology. For quantum encoding compatibility (mapping classical features to Q qubits), we first project the input data to d_q-dimensional space (where d_q is the feature dimension per qubit), then apply global average pooling to aggregate graph-level information, and finally map to n-dimensional vector matching the number of qubits.

C. Quantum Feature Extraction Module

To address the limitations of classical Graph Neural Networks (GNNs) in expressive power due to the Weisfeiler-Lehman (WL) test and exponential growth of parameter scale, this paper proposes a quantum graph feature extraction module. This module utilizes the spectral expansion capability and non-local entanglement properties of quantum computation to encode graph information into high-dimensional quantum features, which are then mapped into a classical GNN-compatible vector space. The architecture consists of three components: data encoding layer, trainable layer, and measurement layer, with the specific circuit implementation shown in Fig. 1(c).

1) **Data Encoding Circuit Block:** The core objective of this layer is to generate high-order frequency spectra through r rounds of repeated encoding operations, while controlling the number of independent parameters to avoid over-parameterization issues. Based on Schuld et al.'s perspective [19] that the representational capacity of quantum models is constrained by frequency bandwidth and spectral flexibility, we adopt multi-round repeated encoding to broaden the accessible frequency bandwidth, and further enhance the quantum module's expressive power by coupling feature dimensionality to encoding rounds and assigning independent parameters to each round.

Specifically, for each input feature of a qubit, we perform r rounds of repeated encoding (in this work, $r = 4$), where each round uses a different type of Pauli rotation gate. Entanglement between quantum states across rounds is achieved via CNOT gates, ensuring fusion of multi-round encoded information. Moreover, we adopt an independently parametrized strategy, assigning independent training parameters θ_t $(t = 1, 2, \ldots, r)$ to each encoding round.

The quantum gate operations for r rounds of encoding applied to the feature vector $\mathbf{x}$ output from the classical graph embedding module can be expressed as:

$$U_{\text{seq},i,j} = \prod_{t=1}^{r} \left(R_{G_t}(\theta_{t,j} + x_{i,j}) \cdot \text{CNOT}(q_{t-1,j}, q_{t,j}) \right) \quad (7)$$

where G_t denotes the Pauli gate type in the t-th round, satisfying $G_t \neq G_{t+1}$ to prevent information confusion caused by consecutive identical gate types. Here, $q_{t,j}$ denotes the j-th feature's t-th encoding qubit (with $q_{0,j}$ being the initial qubit in state $|0\rangle$). The CNOT gate $\text{CNOT}(q_a, q_b) = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \sigma_x$ enables entanglement propagation between successive rounds. In this work, we select the gate sequence $[Z, Y, X, Y]$ $(r = 4)$, where $R_G(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_G\right)$, and σ_G represents the corresponding Pauli matrix $(\sigma_x, \sigma_y, \sigma_z)$, as illustrated in Fig. 1(c).

Frequency Spectrum Analysis: According to the repeated encoding frequency theory proposed in [19], the frequency spectrum generated by r rounds of repeated encoding is determined by the eigenvalues of the generator Hamiltonian associated with the encoding gates. For a Pauli rotation gate $R_G(\theta)$, its generator Hamiltonian has eigenvalues $\pm 1/2$. The eigenvalue sum for r-round encoding is given by:

$$\Lambda_j = \sum_{t=1}^{r} \lambda_{j,t}, \quad \lambda_{j,t} \in \{-1/2, 1/2\} \quad (8)$$

Therefore, the resulting frequency spectrum is:

$$\Omega = \left\{ \Lambda_k - \Lambda_l \,\middle|\, k, l \in \{1, 2, \ldots, 2^r\} \right\} \quad (9)$$

It can be observed that the number of encoding rounds r directly determines the order of the frequency spectrum. When $r = 4$, the method covers all integer frequencies in the interval $[-4, 4]$, achieving a frequency bandwidth four times wider than single-round encoding [19].

2) **Trainable Circuit Block** This layer enhances feature associations within quantum states through trainable quantum circuits, leveraging the non-local entanglement property of quantum systems to capture high-order nonlinear interactions among node features, thereby further improving the expressive power of the feature representation.

Specifically, we employ an RY parameterized gate and a CNOT entanglement layer to construct the trainable quantum circuit structure. The RY parameters are assigned as $\phi = [\phi_1, \phi_2, \ldots, \phi_q]^\top$, used to adjust the phase of each qubit state. And the CNOT entanglement layer adopts a ring-shaped entanglement architecture, the final qubit is connected back to the first qubit, ensuring global entanglement across all qubits and capturing long-range feature correlations.

The overall operation of the quantum feature extraction module can be expressed as:

$$|\psi_{\text{ext}}\rangle = U_{\text{extract}} \cdot |\psi_{\text{enc}}\rangle, \quad U_{\text{extract}} = U_{\text{CNOT}} \cdot \prod_{q=1}^{Q} R_Y(\phi_q) \quad (10)$$

where $|\psi_{\text{enc}}\rangle$ and $|\psi_{\text{ext}}\rangle$ denote the input and output quantum states of this layer, respectively. Here, $R_Y(\phi_q) = \exp\left(-i\frac{\phi_q}{2}\sigma_y\right)$ represents the RY rotation gate applied to the q-th qubit, and $U_{\text{CNOT}} = \prod_{q=1}^{q} \text{CNOT}(q, (q \mod q) + 1)$ denotes the ring-shaped CNOT entanglement layer, which realizes global entanglement among all qubits.

3) **Measurement Block** This layer performs Pauli-Z measurements on all qubits to obtain the expected values of each quantum bit. We apply Pauli-Z measurements to the output state $|\psi_{\text{ext}}\rangle$ from the trainable circuit block, obtaining the expectation value for each qubit as:

$$M_i = \langle \psi_{\text{ext}} | \sigma_Z^{(i)} | \psi_{\text{ext}} \rangle \quad (11)$$

D. Output Module

Based on output dimensionality, we design the output layer and decompose the UCP task into three subtasks. 1) Feasibility Prediction: Outputs $y_{\text{fea}} \in \{0, 1\}$ to filter infeasible samples. 2) Optimal Value Prediction: Predicts $y_{\text{obj}} \in \mathbb{R}$ for feasible instances. 3) Optimal Solution Prediction: Generates $\mathbf{y}_{\text{sol}} \in \{0, 1\}^n$ (n = number of generators, 1 indicates ON, 0 indicates OFF) using the predicted optimal values.

These subtasks vary in granularity, encompassing two levels: graph-level (feasibility and optimal value) and node-level (optimal solution), requiring a differentiated output architecture. Based on the quantum measurement results, we first use a fully connected layer maps them to node embeddings $\mathbf{H}$. For graph-level tasks, we specially employ an Attention Pooling mechanism to aggregate node embeddings into a global graph

embedding $\mathbf{G}$. This mechanism assigns higher weights to key nodes based on their importance in the graph structure. The attention pooling operation is defined as:

$$\mathbf{G} = \sum_{i=1}^{N} \alpha_i \mathbf{H}_i, \quad \alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{N} \exp(e_j)} \tag{12}$$

where $e_i = \mathbf{w}^\top \tanh(\mathbf{W}_2 \mathbf{H}_i + \mathbf{b}_2)$, and $\mathbf{G} \in \mathbb{R}^d$ is the global graph embedding, $\alpha_i \in (0,1)$ is the normalized attention weight for the i-th node satisfying $\sum_{i=1}^{N} \alpha_i = 1$, $\mathbf{W}_2 \in \mathbb{R}^{k \times d}$, $\mathbf{w} \in \mathbb{R}^k$ are trainable parameters, and $\mathbf{b}_2 \in \mathbb{R}^k$ is the bias term.

1) Feasibility Prediction: $\mathbf{G}$ is normalized via sigmoid and binarized:

$$\hat{y}_{\text{fea}} = \sigma(\mathbf{W}_3 \mathbf{G} + b_3), \quad y_{\text{fea}} = \begin{cases} 0, & \hat{y}_{\text{fea}} \leq 0.5 \\ 1, & \hat{y}_{\text{fea}} > 0.5 \end{cases} \tag{13}$$

where $\hat{y}_{\text{fea}} \in (0,1)$ is the predicted probability for feasibility, y_{fea} is the final binary classification label, $\mathbf{W}_3$ is the weight matrix, and b_3 is the bias term.

2) Optimal Value Prediction: $\mathbf{G}$ is fed into a linear layer:

$$\hat{y}_{\text{obj}} = \mathbf{W}_4 \mathbf{G} + b_4 \tag{14}$$

where $\hat{y}_{\text{obj}} \in \mathbb{R}$ is the predicted optimal objective value, $\mathbf{W}_4$, and b_4 are the weight and bias terms of the linear layer, respectively.

3) Optimal Solution Prediction: Node embeddings with independent weights and biases are processed:

$$\hat{y}_{\text{disc}} = \sigma(\mathbf{W}_d \mathbf{H}_{\text{sol}} + b_d), \quad y_{\text{disc}} = \begin{cases} 0, & \hat{y}_{\text{disc}} \leq 0.5 \\ 1, & \hat{y}_{\text{disc}} > 0.5 \end{cases} \tag{15}$$

$$y_{\text{cont}} = \mathbf{W}_c \mathbf{H}_{\text{sol}} + b_c, \quad \hat{\mathbf{y}}_{\text{sol}} = \left[y_{\text{disc}}^\top, y_{\text{cont}}^\top \right]^\top \tag{16}$$

where $\hat{y}_{\text{disc}}$ is the predicted probability for discrete variables, y_{disc} is the binary decision, y_{cont} is the continuous output, and $\hat{\mathbf{y}}_{\text{sol}}$ is the final predicted optimal solution vector.

IV. EXPERIMENT

This chapter first introduces the construction method of the bipartite graph dataset for the UCP and the evaluation metrics, clarifying the experimental setup. Subsequently, three types of experiments are carried out on the constructed bipartite graph dataset: an ablation experiment on the quantum feature extraction module and its parameter combinations in Section IV-C, a comparison of the performance of HQGNN and existing methods in Section IV-D, and finally an analysis of the parameter scale of the hybrid model in Section IV-E.

A. Dataset and Evaluation Metrics

Dataset: The IEEE bus system dataset is a classic and widely used standard dataset in power system research. The RTS-GMLC dataset is an extended and modernized version based on the classical IEEE RTS-96 (Reliability Test System). For the UCP addressed in this paper, we select the IEEE-9 (small-scale system) and IEEE-39 datasets (medium-scale system) [20], and construct the RTS-16 (real scenario system) dataset based on the RTS-GMLC dataset [21]. The details of data construction are provided in Appendix A. Meanwhile, in order to obtain high-quality ground truth labels, the constructed UCP MILP instances are input into the commercial solver Gurobi [22] for exact solution which serves as the ground truth for model training and evaluation. All subsequent experiments are carried out based on the constructed bipartite graph dataset, aiming to verify the effectiveness of the proposed bipartite graph representation method for UCP.

Evaluation Metrics: For the three subtasks of UCP, distinct evaluation metrics are employed:

1) **Feasibility Prediction** constitutes a graph classification task. The binary cross-entropy loss function is utilized for model training.
2) **Optimal Objective Value Prediction** corresponds to a graph-level scalar regression task. We use the MSE metric for training and the Optimality Gap metric to measure the gap between the prediction result and the optimal value through Equation (17). The lower value indicates higher prediction accuracy. This metric is a standard evaluation metric for UCP solutions using machine learning or approximation methods [3]:

$$\text{Optimality Gap} = \frac{Z_{\text{method}} - Z_{\text{optimal}}}{Z_{\text{optimal}}} \times 100\% \tag{17}$$

where Z_{method} denotes the objective value predicted by the evaluated method, and Z_{optimal} represents the optimal value obtained via the Gurobi solver.
3) **Optimal Solution Prediction** falls under node attribute prediction tasks. The MSE metric is similarly employed for both training and evaluating to assess solution accuracy.

B. Experimental Setup

The proposed HQGNN framework is implemented in Python, with TensorFlow [23] utilized as the deep learning framework for classical graph embedding and output module implementation, and PennyLane [24] employed for quantum feature extraction module development. A five-fold cross-validation strategy is adopted to partition the dataset, ensuring robust evaluation of model generalization performance. Experiments are conducted on a workstation running Ubuntu 22.04 LTS, equipped with a 13th Gen Intel® Core™ i9-13900H processor (2.60 GHz), 16.0 GB of RAM, and a 64-bit operating system.

C. Ablation Results

To verify the necessity of the quantum module in the proposed HQGNN, we design two sets of ablation experiments based on the three constructed UCP bipartite graph datasets, using five-fold cross-validation.

The first set of ablation experiments focuses on the functional verification of the quantum module in the hybrid model. We compare the performance between the complete HQGNN and the classical baseline model with the quantum module removed to quantify the gain effect of quantum components on the model. To ensure a fair comparison, the core network structures, training hyperparameters, and regularization strategies

TABLE II
ABLATION STUDY RESULTS.

Dataset	Quantum	Feasibility (Error Rate↓)	Optimal Value (Optimality Gap↓)	Optimal Solution (MSE↓)
IEEE 9	FALSE	0.131 ± 0.017	5.824 ± 0.238	0.419 ± 0.251
	TRUE	0.044 ± 0.015	2.629 ± 0.294	0.110 ± 0.020
	improve	8.70%	3.19%	0.309
IEEE 39	FALSE	0.145 ± 0.015	3.183 ± 0.605	0.366 ± 0.030
	TRUE	0.135 ± 0.161	6.737 ± 7.420	0.210 ± 0.002
	improve	0.96%	−3.55%	0.156
RTS-16	FALSE	0.216 ± 0.021	1.283 ± 0.290	1.030 ± 0.614
	TRUE	0.283 ± 0.020	2.774 ± 0.878	0.718 ± 0.004
	improve	−6.7%	−1.01%	0.312

improve : Negative values indicate performance degradation.

are completely the same, detailed configurations are provided in Appendix B.

The experimental results shown in Tab. II indicate that the proposed HQGNN shows certain advantages in key evaluation indicators. Specifically, in the IEEE-9 dataset, our method achieved comprehensive improvement in three tasks, verifying that the quantum module can capture the complex features of small-scale data. At the same time, we can also see that the addition of quantum modules in the IEEE-39 and RTS-16 datasets did not fully achieve the expected performance improvement, which may be related to the increased complexity of the datasets and the problem of poor plateau encountered by quantum modules during gradient propagation.

The second set of experiments focus on the core parameters of the quantum module. We evaluate the influence of different numbers of qubits (n) and quantum circuit depths (m) on the performance of optimal objective value prediction task. Based on the IEEE-9 dataset, we use the mean value of Optimality Gap as the key evaluation indicator.

Fig. 2 shows that a moderate increase in qubit count and circuit depth significantly reduces the Optimality Gap, enhancing the model's ability to capture high-dimensional features and thus improving UCP solution accuracy. The optimal performance is achieved at $m = 2$ and $n = 8$.

However, the complexity of the quantum module is not monotonically positively correlated with the model performance. Excessive increases in either parameter can degrade results, as seen when $m = 3$ and $n = 10$ underperforms the $m = 3$, $n = 6$ configuration. This is likely due to increased noise accumulation from more qubits and overfitting from excessive circuit depth [25].

In addition, we provide the training loss curve in Fig. 3, depicting the training process of five models in the feasibility prediction task of the IEEE-9 dataset. The results indicate that HQGNN converges at a faster rate. It only takes approximately 100 training epochs for HQGNN to reach stability, while the ablated model requires 180 epochs, demonstrating an improvement in training efficiency.

D. Comparison Results

To verify the performance advantages of the proposed HQGNN, we compare its performance with the classical Graph Convolutional Network (GCN) [26], Graph attention networks

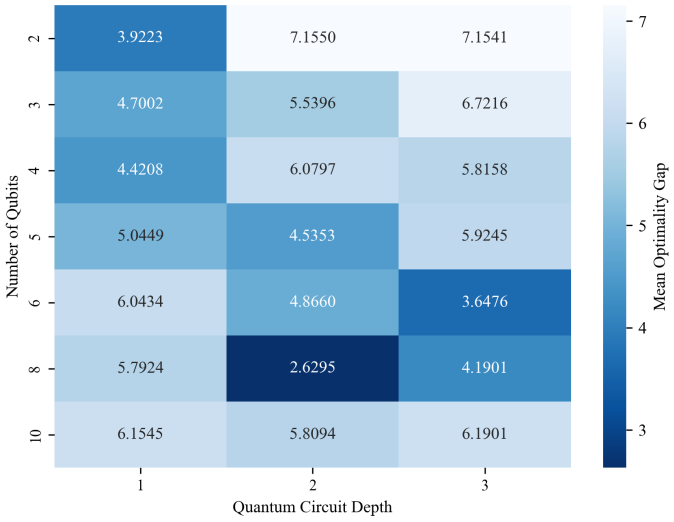

Fig. 2. Heatmap of mean Optimality Gap via five-fold cross-validation (x-axis: quantum circuit depth; y-axis: number of qubits). Each cell value denotes the average Optimality Gap of the parameter combination, used to quantify how quantum parameters affect model performance.

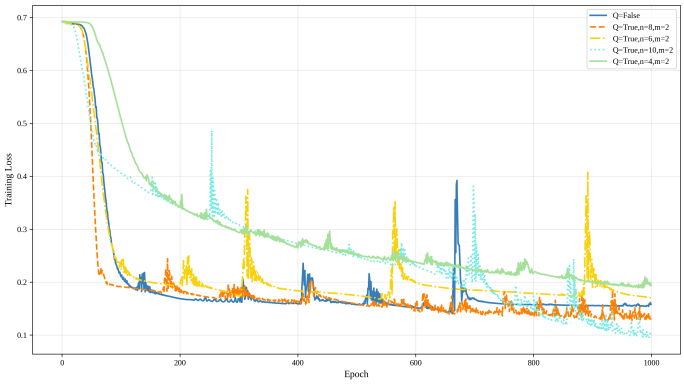

Fig. 3. Training loss curves of five models for feasibility prediction on the IEEE9 dataset (x-axis: epoch; y-axis: training loss). Legend notations: Q = False (ablation model without quantum module), Q = True (with quantum module), n = number of qubits, m = circuit depth.

(GAT) [27], Centralized convolutional networks (CenGCN) [28] and the Quantum Spatial Graph Convolutional Network (QSGCN) [29]. The HQGNN adopts a combination of quantum bits and circuit depth that achieves optimal performance in ablation experiments. To ensure a strict and fair comparison, the key hyperparameters of the three types of models were configured consistently.

The results in Tab. III show that in the optimal solution prediction task, our approach achieves a reduction in MSE by 24.4% on average compared to the best-performing baseline, demonstrating its superior accuracy in recovering near-optimal commitment decisions. Furthermore, on the smallest-scale system (IEEE-9), our method attains the best performance across all three evaluation tasks. These results collectively validate the effectiveness of our framework in leveraging quantum-enhanced representations to improve predictive accuracy in UCP.

TABLE III
COMPARATIVE EXPERIMENTAL RESULTS.

Dataset	Method	Feasibility (Error Rate↓)	Optimal Value (Optimality Gap↓)	Optimal Solution (MSE↓)
IEEE 9	Ours	**0.044 ± 0.015**	**2.629 ± 2.948**	**0.110 ± 0.020**
	GCN	0.086 ± 0.031	5.159 ± 1.015	0.139 ± 0.011
	GAT	0.373 ± 0.141	7.261 ± 0.924	0.289 ± 0.152
	CenGCN	0.371 ± 0.145	7.247 ± 0.928	0.786 ± 0.032
	QSGCN	0.170 ± 0.155	9.645 ± 0.866	0.848 ± 0.603
IEEE 39	Ours	0.135 ± 0.161	6.737 ± 0.742	**0.210 ± 0.002**
	GCN	**0.071 ± 0.006**	**1.006 ± 1.046**	0.258 ± 0.031
	GAT	0.284 ± 0.171	6.202 ± 4.744	0.804 ± 0.084
	CenGCN	0.073 ± 0.006	3.109 ± 0.727	1.550 ± 0.038
	QSGCN	0.430 ± 0.001	18.758 ± 3.197	1.388 ± 0.838
RTS-16	Ours	0.283 ± 0.200	2.774 ± 0.878	**0.718 ± 0.004**
	GCN	0.249 ± 0.015	**1.441 ± 0.889**	1.088 ± 0.025
	GAT	**0.207 ± 0.253**	1.621 ± 0.142	1.104 ± 0.081
	CenGCN	0.300 ± 0.245	1.566 ± 0.196	1.369 ± 0.039
	QSGCN	-	-	-

Entries denoted by "-" indicate that QSGCN is not applicable to the RTS-16 dataset due to the excessively long training time.

TABLE IV
COMPARISON OF PARAMETER SCALES.

Model	d_e	$qubit$	$depth$	P_{module}	P_{total}	$\Delta P(\%)$
Baseline	8	–	–	488	2,701	–
Quantum	8	4	1	32	2,245	-16.80
Baseline	16	–	–	1872	10,233	–
Quantum	16	2	2	120	8,481	-17.12

d_e: embedding size, $qubit$: number of qubits, $depth$: depth of quantum module, P_{module}: parameters of single module, P_{total}: total parameters, $\Delta P(\%)$: percentage reduction; Entries marked with "–" indicate that the metric is not applicable (N/A) because baseline models are purely classical and contain no quantum components.

E. Parameter Efficiency Analysis

To validate the parameter efficiency of the quantum module in HQGNN, we compare its total parameter count against a classical counterpart and analyze how this efficiency scales with feature dimension and quantum-circuit hyperparameters.

As shown in Tab. IV, replacing the quantum module with a classical graph-convolutional layer significantly increases the parameter count. HQGNN reduces total parameters by 16.80% at an embedding size of 8, and by 17.12% at an embedding size of 16, demonstrating its superior scalability.

Besides, we compare the parameter growth patterns of the classical and quantum modules as the feature dimension increases (assuming equivalent representation dimensions). As shown in Fig. 4, the classical graph convolutional module exhibits rapid parameter expansion, highlighting its inherent parameter explosion problem. In contrast, the quantum module demonstrates a significantly slower growth rate, avoiding exponential parameter increases as the number of qubits (equivalent embedding dimension) rises.

V. CONCLUSION

In this paper, we try to address two key limitations of UCP: the lack of universal and efficient representation approaches and the explosion of parameter scale. We propose HQGNN, a hybrid quantum-classical graph neural network framework that models UCP as a unified bipartite graph and integrates a quantum feature extraction module to capture high-order graph dependencies. Comprehensive experiments on IEEE and RTS datasets validate the necessity of the quantum module in HQGNN and demonstrate the superiority of the hybrid model. Current limitations include performance in large-scale real-world scenarios and reliance on simulated environments. Future work will focus on optimizing the noise robustness of the quantum module, extending its application to complex UCP scenarios, and exploring integration with advanced optimization algorithms. This study provides a novel paradigm for UCP solving and inspires research on quantum-classical hybrid intelligence in power system optimization.

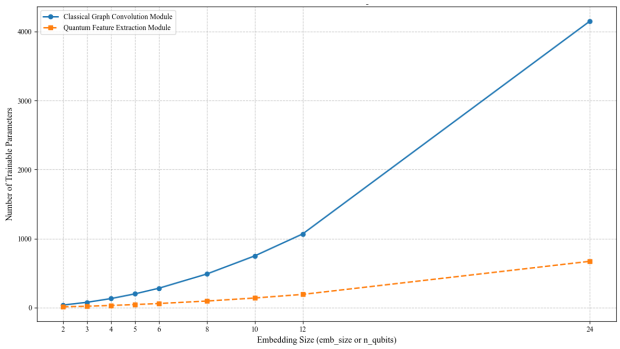

Fig. 4. Trainable parameters of quantum and classical graph convolution modules vs. embedding size (x-axis: embedding size; y-axis: trainable parameters). It shows quantum modules have slower parameter growth than classical graph convolution modules.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (No. 2024YFB4504100).

REFERENCES

[1] L. Montero, A. Bello, and J. Reneses, "A review on the unit commitment problem: approaches, techniques, and resolution methods," *Energies*, vol. 15, no. 4, p. 1296, 2022.

[2] W. Ackooij, I. Lopez, A. Frangioni, C. Gentile, F. Lacalandra, and M. Tahanan, "Large-scale unit commitment under uncertainty: an updated literature survey," *Annals of Operations Research*, vol. 271, pp. 11–85, 2018.

[3] Á. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," arXiv preprint arXiv:1902.01697, 2019.

[4] J. Zou, S. Ahmed, and X. Sun, "Multistage stochastic unit commitment using stochastic dual dynamic integer programming," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 1814–1823, Jul. 2019.

[5] Y. Yang and L. Wu, "Machine learning approaches to the unit commitment problem: current trends, emerging challenges, and new strategies," *Electr. Power Syst. Res.*, vol. 191, p. 106592, 2021.

[6] C. Zhang, Z. Qin, and Y. Sun, "Learning-based branching acceleration for unit commitment with few training samples," *Appl. Sci.*, vol. 15, no. 6, p. 3366, 2025.

[7] F. Pourahmadi and J. Kazempour, "Unit commitment predictor with a performance guarantee: a support vector machine classifier," *IEEE Trans. Power Syst.*, vol. 40, pp. 715–727, 2023.

[8] Y. Zhang, Y. Yang, X. Li, T. Wang, and G. Chen, "Transformer and RF ensemble learning to solve large-scale security constrained unit commitment," in *Proc. 8th Int. Conf. Energy, Electr. Power Eng. (CEEPE)*, pp. 7–12, 2025.

[9] S. Schmitt, I. Harjunkoski, M. Giuntoli, J. Poland, and X. Feng, "Fast solution of unit commitment using machine learning approaches," in *Proc. IEEE 7th Int. Energy Conf. (ENERGYCON)*, pp. 1–6, 2022.

[10] M. Pan, Y. Gao, C. Huang, Y. Zhao, and Y. Yu, "Heterogeneous ensemble learning to solve sub-hourly unit commitment," in *Proc. 2nd Int. Conf. Power Syst. Electr. Technol. (PSET)*, pp. 207–212, 2023.

[11] L. Yang, P. Li, and J. Jian, "Solving unit commitment problems with graph neural network based initial commitment prediction and large neighborhood search," arXiv preprint arXiv:2505.14408, 2025.

[12] A. Frangioni, C. Gentile, and F. Lacalandra, "Tighter approximated MILP formulations for unit commitment problems," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 105–113, Feb. 2009.

[13] Y. Zhang, J. Wang, B. Zeng, and Z. Hu, "Chance-constrained two-stage unit commitment under uncertain load and wind power output using bilinear benders decomposition," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3637–3647, Sep. 2017.

[14] J. Ostrowski, M. F. Anjos, and A. Vannelli, "Tight mixed integer linear programming formulations for the unit commitment problem," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 39–46, Feb. 2012.

[15] C. Zhang and L. Yang, "A hybrid approach for unit commitment with splitting technique and local search," *Elect. Power Syst. Res.*, vol. 227, p. 110084, 2024.

[16] Z. Chen, J. Liu, X. Wang, and W. Yin, "On representing mixed-integer linear programs by graph neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[17] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 6861–6871.

[18] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2016, pp. 3844–3852.

[19] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Phys. Rev. A*, vol. 103, no. 3, pp. 032430, Mar. 2021.

[20] R. Zhao, Z. Song, Y. Xu, J. Lu, W. Guo, and H. Li, "Low-carbon demand response program for power systems considering uncertainty based on the data-driven distributionally robust chance constrained optimization," *IET Renew. Power Gener.*, vol. 19, no. 1, p. e13021, Jun. 2024.

[21] Illinois Institute of Technology (IIT)-ICSEG, "ICSEG power system test cases dataset," 2025. [Online]. Available: https://icseg.iti.illinois.edu/power-cases/

[22] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2024. [Online]. Available: https://www.gurobi.com

[23] M. Abadi, A. Agarwal, P. Barham, et al., "TensorFlow: large-scale machine learning on heterogeneous systems," 2015, Software available from: https://www.tensorflow.org.

[24] V. Bergholm, J. Izaac, M. Schuld, et al., "PennyLane: Automatic differentiation of hybrid quantum-classical computations," arXiv preprint arXiv:1811.04968, 2018.

[25] R. Harper, S. T. Flammia, and J. J. Wallman, "Efficient learning of quantum noise," *Nat. Phys.*, vol. 16, pp. 1184–1188, 2020.

[26] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 1263–1272, 2017.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.(ICLR)*, 2018.

[28] F. Xia, L. Wang, T. Tang, X. Chen, X. Kong, G. Oatley, and I. King, "CenGCN: Centralized convolutional networks with vertex imbalance for scale-free graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4555–4569, May. 2023.

[29] J. Zheng, Q. Gao, M. Ogorzałek, J. Lü, and Y. Deng, "A quantum spatial graph convolutional neural network model on quantum circuits," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 3, pp. 5706–5720, Mar. 2025.

APPENDIX

A. Construction Details of Dataset

We now present the construction details of the dataset. Taking RTS-16 for example, we select 16 coal-fired steam turbine units (`steam+coal` category) from original RTS-GMLC dataset. Next, we calculate the parameter Fuel Cost(C_i^{fuel}) in the objective function, while other parameters can be directly obtained from the original dataset file: the startup cost and shutdown cost is extracted from the `start_heat_cold` field and `Non_Fuel_Shutdown_Cost` field. Based on the the heat rate (`HR_incr_1`)and fuel price(`Fuel_Price`) , we can calculate the Fuel Cost:

$$\text{Fuel Cost } = Heat\_Rate \times 0.001 \times Fuel\_Price$$

Taking the unit `101_STEAM_3` as an example:

$$6713 \text{ BTU/kWh} \times 0.001 \times 2.11 \text{ \$/MMBTU} \approx 14.17 \text{ \$/MWh}$$

Based on the above construction, we construct three complete UCP datasets (IEEE9, IEEE39, RTS-16). Due to space constraints, only partial parameters of the RTS-16 dataset are presented in Table V.

TABLE V
DETAILED PARAMETERS OF RTS-16

Unit	Startup Cost ($)	Shutdown Cost ($)	Min Power (MW)	Max Power (MW)	Fuel Price (/$MWh)	Base Load (MW)
1	11172	0	30	76	14.19	2317
2	11172	0	30	76	14.19	
3	11172	0	30	76	18.46	
4	11172	0	30	76	18.46	
5	22784	0	62	155	20.4	
6	22784	0	62	155	19.68	
7	22784	0	62	155	19.43	
8	36749	0	140	350	19.98	
9	11172	0	30	76	22.19	
10	11172	0	30	76	21.12	
11	11172	0	30	76	21.12	
12	22784	0	62	155	18.57	
13	22784	0	62	155	19.03	
14	22784	0	62	155	19.03	
15	36749	0	140	350	18.86	
16	22784	0	62	155	21.12	

B. Implementation Details of Methods

We adopt the Adam optimizer with initial learning rate 1×10^{-3}, weight decay of 1×10^{-4}, early stopping patience of 10, batch size of 16, dataset split ratio of 0.8, training epochs of 1000. Gradients are computed using the parameter-shift rules for quantum module. We set the loss criterion to `nn.MSELoss()` for regression tasks and `nn.CrossEntropyLoss()` for classification tasks. Data standardization is performed. Aditional hyperparameter details are provided in Table VI.

TABLE VI
IMPLEMENTATION DETAILS OF THE COMPARATIVE METHODS.

Method	**Parameter Configuration**
Our Method	embedding size = 8, hidden dim = 32, Feasibility Prediction:(qubits = 6, depth = 1), Optimal Objective Value Prediction:(qubits = 8, depth = 3), Optimal Solution Prediction:(qubits = 6, depth = 2).
GCN [26]	weight decay = 1×10^{-4}, hidden dim = 32, layers = 4, dropout rate = 0.2.
GAT [27]	attention heads = 4, hidden dim = 32, layers = 4, dropout rate = 0.2.
CenGCN [28]	hidden dim = 32, layers = 2, dropout rate = 0.2, normalize = True.
QSGCN [29]	$K = 8$, $M = 4$, $D = 8$, $T = 100$.