Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/TQE.2020.DOI

Learning Infused Quantum-Classical Distributed Optimization Technique for Power Generation Scheduling

REZA MAHROO[1] (Student Member, IEEE),
AMIN KARGARIAN[1] (Senior Member, IEEE)
[1]Louisiana State University, Baton Rouge, LA 70803 USA
Corresponding author: Amin Kargarian (email: kargarian@lsu.edu).

ABSTRACT The advent of quantum computing can potentially revolutionize how complex problems are solved. This paper proposes a two-loop quantum-classical solution algorithm for generation scheduling by infusing quantum computing, machine learning, and distributed optimization. The aim is to facilitate employing noisy near-term quantum machines with a limited number of qubits to solve practical power system optimization problems such as generation scheduling. The outer loop is a 3-block quantum alternative direction method of multipliers (QADMM) algorithm that decomposes the generation scheduling problem into three subproblems, including one quadratically unconstrained binary optimization (QUBO) and two non-QUBOs. The inner loop is a trainable quantum approximate optimization algorithm (T-QAOA) for solving QUBO on a quantum computer. The proposed T-QAOA translates interactions of quantum-classical machines as sequential information and uses a recurrent neural network to estimate variational parameters of the quantum circuit with a proper sampling technique. T-QAOA determines the QUBO solution in a few quantum-learner iterations instead of hundreds of iterations needed for a quantum-classical solver. The outer 3-block ADMM coordinates QUBO and non-QUBO solutions to obtain the solution to the original problem. The conditions under which the proposed QADMM is guaranteed to converge are discussed. Two mathematical and three generation scheduling cases are studied. Analyses performed on quantum simulators and classical computers show the effectiveness of the proposed algorithm. The advantages of T-QAOA are discussed and numerically compared with QAOA which uses a stochastic gradient descent-based optimizer.

INDEX TERMS Quantum computing, variational quantum algorithm, machine learning, distributed optimization, generation Scheduling.

I. INTRODUCTION

COMPUTIONAL challenges have always played a significant role in the design and operation of complex systems. Power system modernization poses complexity and computational challenges that classical computers and solvers may not meet [1]. Generation scheduling is a fundamental problem in power systems, which determines generating units' status to provide the load while satisfying operational constraints cost-effectively [2]. This problem is mathematically set as a mixed-integer linear programming (MILP) problem [3]. As the number of generating units and the penetration of renewables and distributed resources increase, generation scheduling complexity and computational burden increase exponentially. Computationally efficient and scalable approaches must be developed to deal with this problem.

Quantum computing, a rapidly emerging technology, potentially opens new opportunities in solving computationally challenging problems that classical resources may not be able to address [4]. Quantum computing uses quantum mechanical laws to perform computation operations. Despite significant progress in recent years, universal error-corrected quantum computers have yet to be achieved. Currently, noisy intermediate-scale quantum technology is available to implement quantum algorithms and recognize problems for which quantum computing outperforms classical counterparts.

Solving combinatorial optimization is a research scope that quantum speedup is expected. This is achieved us-

ing methods such as quantum approximate optimization algorithm (QAOA) [5], Grover algorithm [6], and variational quantum eigensolver [7]. The application of quantum computing extends beyond solving combinatorial problems [8], [9]. While quadratic unconstrained binary optimization (QUBO) is a well-known class of combinatorial optimization problems that can benefit from quantum computing, it also offers potential advantages for solving a wide range of other optimization problems such as the quantum interior point method for linear programming and semi-definite programming [10]. A QUBO problem can be transformed into an Ising model using a Hamiltonian that includes the weighted tensor product of Pauli-Z operators [11]. Limitations of quantum computing algorithms and noisy intermediate-scale quantum devices have initiated the development of hybrid quantum-classical techniques to cope with large-scale problems. The aforementioned quantum algorithms, e.g., QAOA, are designed based on gate-based quantum computers, which are still in the early stage of development and unable to solve large problems [12]. Also, a special-purpose design of quantum computers to solve QUBOs has been developed recently based on quantum annealing [13]. Such hybrid algorithms aim to decompose a problem into one subproblem that can be efficiently handled in a classical computer and another subproblem that can be assigned to a quantum processing unit (QPU). Hybrid quantum-classical algorithms can be found in the literature for both general-purpose and special-purpose problems.

To solve MILP problems, [14] and [15] have presented hybrid techniques based on the Benders technique to decompose the original problem into a MILP master problem and a convex linear programming subproblem. In [14], continuous variables are discretized to convert the master problem into a QUBO. This discretization needs ancillary qubits. In [15], a Benders cut selection scheme manages the size of the master problem. The cut selection strategy is a QUBO problem assigned to a QPU. Although Benders-based algorithms guarantee convergence, they take many iterations. A hybrid technique based on the multi-block alternating direction method of multipliers (ADMM) [16] is presented in [17] for MILP problems. The complicating binary variables are relaxed to vary continuously. The problem is split into a QUBO solvable by QPU and continuous blocks solvable by classical solvers. A two-block and three-block version of this method is adopted in [18] and [19], respectively, to solve unit commitment. Although this approach is useful for adjusting the complexity of binary subproblems by encoding them through entangled states encoded within qubits, it might not converge as the problem size increases. Surrogate Lagrangian relaxation is presented in [20] for a hybrid solution of unit commitment. In addition to these algorithms, [21]–[23] present similar ideas for solving specific problems using quantum computing. The authors in [21] and [22] discretize continuous variables into h parts to turn them into binary variables. This method is not practical as for a problem with n continuous variables, $n(h+1)$ ancillary qubits are required. In [23], heuristic methods extend the variational quantum eigensolver for solving MILP problems. The main drawback is the inability to guarantee convergence to the global or local optimum. The proposed approach in [24] integrates a classical heuristic algorithm with a quantum annealer to achieve better-quality solutions in a shorter time frame than traditional methods. Authors in [25] convert the combinatorial optimal power flow problem into a QUBO problem, which is amenable to quantum annealing. In [26], a quantum teaching learning-based algorithm for optimal energy management of microgrids is proposed, which outperforms other optimization algorithms in terms of convergence speed and accuracy. We note that previous studies have mainly proposed approaches for exploiting quantum computers. The computational process of quantum algorithms, such as QAOA, has not been sufficiently investigated.

QAOA is a variational quantum algorithm for solving combinatorial optimization problems. It is a promising candidate for demonstrating quantum advantage in the near future [23]. The main concept of QAOA is to alternately repeat the cost Hamiltonian, in which its ground state encodes the problem solution and mixing Hamiltonian. It relies on preparing a parameterized quantum circuit on a quantum device and a classical optimizer to find the best quantum circuit parameters. QAOA is introduced in [5] to solve combinatorial optimization problems. The model and study are derived from a MaxCut problem. Following that work, [27] has studied QAOA's ability to solve more complex problems. The quality of the solution resulting from QAOA is affected by the quality of variational parameters prepared in a classical optimizer. Therefore, developing effective variational parameters optimization techniques is crucial to achieving quantum advantages. Various approaches are proposed for optimizing variational parameters, including gradient-based [28] and gradient-free methods [29], [30]. Neural networks and learning techniques have also been used to optimize the variational parameters [30], [31]. Optimization-based methods take many more iterations to achieve optimal results as compared to learning-based methods [32]. Scalability is also a key point that learning-based methods cannot address easily.

In this paper, we develop a trainable two-loop quantum-classical optimization algorithm for generation scheduling. Generation scheduling is decomposed into one QUBO and two non-QUBO subproblems. The inner QAOA loop solves QUBO on quantum computers. The iterative interactions between the quantum circuit and classical optimizer are translated as sequential time series-type information. A scalable deep recurrent neural network plays the role of an optimizer mimicking the iterative trace between QPU and a conventional computer to determine the optimal quantum circuit variational parameters. With a proper sampling technique, the chosen learner optimizer provides the proposed trainable QAOA (T-QAOA) with the flexibility to converge within a pre-determined number of iterations instead of hundreds to thousands of iterations that may take by the conventional

QAOA. An outer 3-block quantum-ADMM (QADMM) loop is designed to coordinate generation scheduling QUBO and non-QUBO subproblems. The inner loop learner stays unchanged at every outer QADMM iteration. The scalability and convergence conditions of the proposed algorithm are discussed. Numerical results on a real quantum computer and quantum simulator show the effectiveness of the proposed trainable two-loop algorithm.

The paper is organized as follows. Section II presents generation scheduling and a preliminary discussion on quantum computing. The generation scheduling decomposition and the T-QAOA algorithm are proposed in Section III. Numerical results are discussed in Section IV, and concluding remarks are provided in Section V.

II. PRELIMINARIES

A. COMPACT GENERATION SCHEDULING FORMULATION

Assume that $p_{i,t}$ denotes continuous variables (e.g., power output), $y_{i,t}$ denotes discrete variables (e.g., on/off status) of unit i at time t. Let $\mathcal{I} = \{1, 2, ..., N\}$ and $\mathcal{T} = \{1, 2, ...T\}$ be the set of generating units and time periods. For brevity of notation, we use (p, y) and $(\mathrm{p}_i, \mathrm{y}_i)$ in the following equations where $(\mathrm{p}_i, \mathrm{y}_i)$ refers to vectors of unit i containing variables $(p_{i,t}, y_{i,t}), \forall t \in \mathcal{T}$, and (p, y) refers to vectors containing variables $(p_{i,t}, y_{i,t}), \forall i \in \mathcal{I}$ and $\forall t \in \mathcal{T}$. The generation scheduling problem is as follows:

$$\min_{p_{i,t}, y_{i,t}} \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} [f(p_{i,t}) + g(y_{i,t})], \tag{1a}$$

s.t.

$$Q(p_{i,t}) = 0; \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \tag{1b}$$

$$D(y_{i,t}) = 0; \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \tag{1c}$$

$$W(p_{i,t}, y_{i,t}) \leq 0; \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \tag{1d}$$

where $p_{i,t} \in \mathbb{R}$ and $y_{i,t} \in \{0, 1\}$. The objective function (1a) represents the generation cost f and commitment cost g. Equality (1b) is responsible for constraints associated with continuous variables such as power balance. Generators on/off logic equations are considered in (1c). Constraint (1d) represents the power output limitations, transmission line bounds, reserve requirements, and ramping rate, including binary and continuous variables. To simplify notation, we introduce a representation for the generation cost term $f(p_{i,t})$ and the commitment cost term $g(y_{i,t})$ within the objective function (1a) as follows:

$$\mathcal{F}(\mathrm{p}) = \sum_{i\in\mathcal{I}} f_i(\mathrm{p}_i) = \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} b_i \cdot p_{i,t}, \tag{2a}$$

$$\mathcal{G}(\mathrm{y}) = \sum_{i\in\mathcal{I}} g_i(\mathrm{y}_i) = \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} c_i \cdot y_{i,t}, \tag{2b}$$

where objective function (1a) is equal to $\mathcal{F}(\mathrm{p}) + \mathcal{G}(\mathrm{y})$. b_i is the fixed cost coefficient of unit i and c_i represents the standby cost of unit i, including no-load cost, startup cost, and shutdown cost.

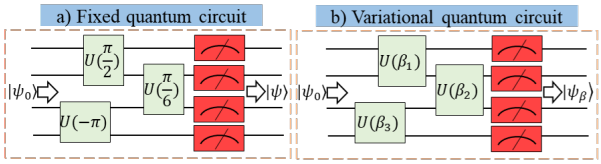

FIGURE 1. Schematic of the fixed and variational quantum circuits.

B. QUANTUM COMPUTING

Classical computers encode information in binary bits, which are either 0s or 1s, and use integrated circuits that contain millions of transistors. Quantum computers also operate data as a series of qubits. Unlike regular bits, qubits can simultaneously be at both $|0\rangle$ and $|1\rangle$ states with a certain probability. Therefore, one qubit can store 2 bits of information. This quality results in a system's exponential scaling advantage regarding the number of required qubits [33]. In a processor with n qubits, 2^n bits of information can be stored. Qubits contain two main properties called superposition and entanglement. Superposition refers to the quantum system's ability to be in multiple states simultaneously, and the correlation between quantum particles is referred to as entanglement [34], [35].

Quantum computers use quantum gates to execute calculations within quantum circuits, as logical circuits and logic gates do in classical computers. Qubits, after initialization, travel through quantum gates, experiencing a rotation, which basically refreshes the probability of each state as $|\psi\rangle = U|\psi_0\rangle$. A typical fixed quantum circuit is shown in Fig. 1(a). Where $|\psi_0\rangle$ is the initial state, U represents the equal unitary operator of the circuit at the given angles, and $|\psi\rangle$ is the output state. The measurement occurs at the end of the circuit, as states with a higher probability appear more frequently in the measurement. Quantum circuits designed based on free parameters are known as variational quantum circuits (VQC) [36], as in Fig. 1(b). The output of a VQC, $|\psi_\beta\rangle = U(\beta)|\psi_0\rangle$, is controlled by its variational parameters β. By optimizing this parameter, VQC conducts different tasks, e.g., solving QUBO problems [5] and system of linear equations [37].

To perform optimization using quantum computers, we should prepare the Hamiltonian of the Ising model or QUBO corresponding to the objective function and constraints of the considered optimization problem. The ground state of a corresponding Ising Hamiltonian is the optimal solution to a QUBO problem [38]. The Ising model or QUBO can be represented by a graph $G(V, E)$ where V and E refer to the set of vertices and edges. Fig. 2 shows an example of an undirected graph based on which the Hamiltonian Ising model can be constructed. For a random graph G, the Hamiltonian of the Ising model is as follows [11]:

$$\mathcal{H}(\mathrm{s}) = \sum_{k\in V} h_k s_k + \sum_{(k,j)\in E} J_{kj} s_k s_j, \quad s_k = \pm 1, \tag{3}$$

where s_k is the spin at vertex $k \in V$, J_{kj} pertains to the interaction between qubits k and j, $(k, j) \in E$, and h_k

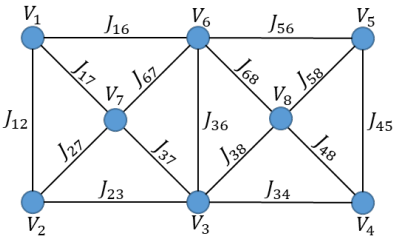

FIGURE 2. A sample graph of a Hamiltonian Ising model.

is the external magnetic field at vertex $k \in V$. From the physical point of view, vertices are physical qubits, and edges represent the potential locations of two-qubit gates. The left-hand-side argument of the Hamiltonian typically represents the state of the system, i.e., the spins' configuration. In the simplest Ising model, each spin can take one of two values, -1 (spin down) or +1 (spin up). For a system of N spins, the state of the system can be represented as a vector or array $\mathrm{s} = (s_1, s_2, ..., s_N)$, where each s_i is the state of the i^{th} spin and is either -1 or +1. This vector s would be the left-hand-side argument of the Ising model's Hamiltonian.

A quadratic unconstrained binary optimization is an energy function that can be transformed into an Ising Hamiltonian ($\{\pm 1\}^n$) with a simple conversion of variables [39]. Consider the following QUBO with a binary variable x_k, linear cost term t_k, and quadratic cost term q_{kj}.

$$C(x) = \sum_{k \in V} t_k x_k + \sum_{(k,j) \in E} q_{kj} x_k x_j, \quad x_k \in \{0, 1\}. \tag{4}$$

To convert QUBO (4) into an Ising model, the relation $x_k = (1 + s_k)/2$ is used [40]. This transformation is applied to all binary variables that appear in the QUBO function. In the case of existing soft constraints $Q(x) = 0$, a QUBO problem can be retrieved by adding a quadratic penalty term $\rho||D(x)||^2$ to the objective function based on penalty method [41], [42]. Inequality constraints such as $W(x) \leq 0$ are converted into equality constraints with the help of non-negative integer slack variables as $W(x) + \zeta = 0$. ζ takes a value as large as $-\min_x W(x)$. This slack variable is expressed as $\zeta = \sum_{l=0}^{l} 2^l I_l$, where $l = \log_2(-\min_x W(x))$ represents the number of required bits. Here, I refers to a set of ancillary bits with a length of l. Eventually, the inequality constraints can be treated like equality constraints. In a nutshell, the total Ising Hamiltonian for an objective function $C(x)$, with equality and inequality constraints $D(x) = 0$ and $W(x) \leq 0$, is as follows:

$$\begin{aligned} \mathcal{H}_c = \mathcal{H}_{obj} &+ \varrho_1 ||D\left(\frac{1+s}{2}\right)||^2 \\ &+ \varrho_2 ||(W\left(\frac{1+s}{2}\right) + \sum_{l=0}^{l} 2^l \frac{1+s}{2}||^2, \end{aligned} \tag{5}$$

where ϱ_1 and ϱ_2 are large positive coefficients. $\mathcal{H}_{obj}$ represents the Hamiltonian regarding the objective function $C(x)$. The set of binary bits I, which represents the slack variables, also needs to be converted into the Ising model using the relation $I_l = (1 + s_l)/2$. Once the QUBO problem has been converted into the Ising Hamiltonians' representation, it becomes amenable to the solution by QAOA. This transformation process allows us to leverage the quantum computation capabilities of QAOA, making it possible to find solutions to the problem, initially defined in the QUBO form, through the corresponding Ising model representation.

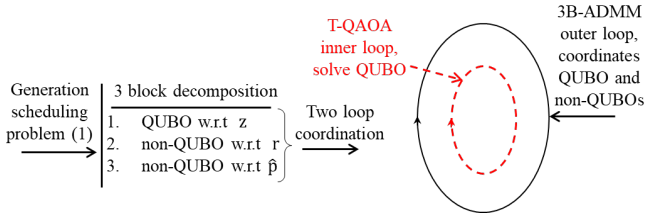

FIGURE 3. Overview of the proposed algorithm.

III. 3-BLOCK QUANTUM ALTERNATING DIRECTION METHOD OF MULTIPLIERS (QADMM)

A decomposition technique is developed to convert the generation scheduling problem (1) into three blocks. The first block formulation is QUBO, which will be solved by T-QAOA on a quantum computer. The other two blocks are convex optimization problems that can be efficiently solved using classical solvers. The three subproblems are coordinated using 3B-ADMM. Fig. 3 shows a schematic of the proposed solution algorithm.

A. DECOMPOSITION TECHNIQUE

Generation scheduling (1) includes continuous and binary variables. We apply a reformulation to decompose this problem. Binary variables y are relaxed to vary continuously as $0 \leq \mathrm{y} \leq 1$. Solving problem (1) with these relaxed binary variables, however, does not yield accurate results as the variables y may not be resolved to 0 or 1. Therefore, a set of auxiliary binary variables z and continuous variables r are defined. Three sets of constraints (6e), (6f), and (6g) guarantee the binary nature of y. Optimization (1) is now reformulated as (6).

$$\min_{\mathrm{p,y,r,z}} \mathcal{F}(\mathrm{p}) + \mathcal{G}(\mathrm{y}), \tag{6a}$$

s.t.

$$Q(\mathrm{p}) = 0, \tag{6b}$$

$$D(\mathrm{y}) = 0, \tag{6c}$$

$$W(\mathrm{p}, \mathrm{y}) \leq 0, \tag{6d}$$

$$A_1 \cdot \mathrm{y} - A_2 \cdot \mathrm{z} + A_3 \cdot \mathrm{r} = 0 : \lambda, \tag{6e}$$

$$\mathrm{r} = 0, \tag{6f}$$

$$D(\mathrm{z}) = 0, \tag{6g}$$

$$0 \leq \mathrm{y} \leq 1, \tag{6h}$$

where $z_{i,t} \in \{0, 1\}$, $r_{i,t} \in \mathbb{R}$, and (z, r) refers to $(z_{i,t}, r_{i,t}), \forall i \in \mathcal{I}$ and $\forall t \in \mathcal{T}$. $A_1 = A_2 = A_3$ are identity matrices. The dual variable associated with constraint (6e) is

denoted by λ, provided following a colon. Equation (6e) indicates that relaxed variables y adopt binary values when the associated auxiliary variables r are zero. Let $\hat{\mathrm{p}} = [\mathrm{p}^T, \mathrm{y}^T]^T$ and $\mathcal{F}(\hat{\mathrm{p}}) = \mathcal{F}(\mathrm{p}) + \mathcal{G}(\mathrm{y}) + \iota_\chi(\mathrm{p}, \mathrm{y})$, where $\iota_\chi(\mathrm{p}, \mathrm{y})$ is an indicator function, and $\chi = \{(\mathrm{p}, \mathrm{y}) \subseteq \mathbb{R} | (6b) - (6d), (6h)\}$. Variables of (6) are split into three sets as $\hat{\mathrm{p}} = \{\mathrm{p}, \mathrm{y}\}$, z, and r. With (6e) relaxed, the problem has a block decomposable structure with respect to each set of variables $\hat{\mathrm{p}}$, z, and r. This means that for a given set of other variables, a smaller subproblem of (6) arises for each variable set. For instance, if z and r are known, we can formulate an optimization subproblem consisting only of variables $\hat{\mathrm{p}}$. We dualize (6e) using augmented Lagrangian and relax the soft constraints (6f)-(6g) by adding penalty terms to the objective function as follows:

$$\begin{aligned}\mathcal{L}_\rho(\hat{\mathrm{p}}, \mathrm{z}, \mathrm{r}, \lambda) &= \mathcal{F}(\hat{\mathrm{p}}) \\ &+ \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\left[\frac{\sigma}{2}||r_{i,t}||_2^2 + \frac{\omega}{2}||D(\mathrm{z}_{i,t})||_2^2\right] \\ &+ \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\lambda^T\Big[A_1\cdot\mathrm{y} - A_2\cdot\mathrm{z} + A_3\cdot\mathrm{r}\Big] \\ &+ \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\left[\frac{\rho}{2}||A_1\cdot\mathrm{y} - A_2\cdot\mathrm{z} + A_3\cdot\mathrm{r}||_2^2\right].\end{aligned} \tag{7}$$

where λ denotes dual variables, and σ, ω, and ρ are penalty parameters. We then decompose (7) and solve subproblems according to the procedure outlined in Algorithm 1.

Algorithm 1 QADMM algorithm

1: Initialize: $m = 1$, $\lambda^{(0)}$, $\rho > (\sigma, \omega) > 0$, $\hat{\mathrm{p}}^{(0)}, \mathrm{r}^{(0)}, \epsilon > 0$.
2: **for** $m = 1, 2, \ldots,$ **do**
3: QUBO block update:
4: $\mathrm{z}^{(m)} \leftarrow \underset{\mathrm{z}}{\mathrm{argmin}}\mathcal{L}_\rho(\hat{\mathrm{p}}^{(m-1)}, \mathrm{z}, \mathrm{r}^{(m-1)}, \lambda^{(m-1)})$.
5: Second block update:
6: $\mathrm{r}^{(m)} \leftarrow \underset{\mathrm{r}}{\mathrm{argmin}}\mathcal{L}_\rho(\hat{\mathrm{p}}^{(m-1)}, \mathrm{z}^{(m)}, \mathrm{r}, \lambda^{(m-1)})$.
7: Third block update:
8: $\hat{\mathrm{p}}^{(m)} \leftarrow \underset{\hat{\mathrm{p}}}{\mathrm{argmin}}\mathcal{L}_\rho(\hat{\mathrm{p}}, \mathrm{z}^{(m)}, \mathrm{r}^{(m)}, \lambda^{(m-1)})$.
9: Dual variable update:
10: $\lambda^{(m)} \leftarrow \rho(A_1\cdot\mathrm{y}^{(m)} - A_2\cdot\mathrm{z}^{(m)} + A_3\cdot\mathrm{r}^{(m)}) + \lambda^{(m-1)}$.
11: **if** $||A_1\cdot\mathrm{y}^{(m)} - A_2\cdot\mathrm{z}^{(m)} + A_3\cdot\mathrm{r}^{(m)}|| \leq \epsilon$ **then** Stop.
12: **else**
13: $m \leftarrow m + 1$.
14: **end if**
15: **end for**
16: Return $(\hat{\mathrm{p}}, \mathrm{z}, \mathrm{r})$.

The iteration index m, penalty factors, decision variables, and stopping criteria are set in the initialization step. The first block, called QUBO block, is an optimization problem over the auxiliary variables z given $\hat{\mathrm{p}}$ and r. This block can be assigned to a quantum computer. The second block is a quadratic unconstrained optimization problem over auxiliary variables r given $\hat{\mathrm{p}}$ and z. This problem is not computationally expensive for classical computers as it is convex and unconstrained. The third block, which is a relaxed version of (1) with relaxed binary variables, is a quadratic problem over variables $\hat{\mathrm{p}}$ given z and r. The third block represents a problem significantly simpler to solve than the original problem (1), primarily because it neither contains binary variables nor non-convex constraints. If the third block problem is infeasible, it also implies that the original problem (1) is infeasible, and Algorithm 1 stops. Since the QUBO subproblem is a non-convex problem, ADMM is generally heuristic. However, Algorithm 1 is guaranteed to converge to a stationary point under some conditions for a large enough $\rho > \max\{\sigma, \omega\}$.

The selection of parameters ρ, σ, and ω is generally problem-dependent, and they must satisfy the condition $\rho > \max\{\sigma, \omega\}$. It is crucial to choose these parameters carefully to strike a balance between fulfilling constraints and optimizing the objective function. Each parameter affects the weighting of the connected regularization terms within the Lagrangian function. By assigning larger values to these parameters, the solution will incur greater penalties for larger constraint values, thus assisting in the enforcement of soft constraints. However, using larger parameter values could also make the iterative optimization process more challenging. To select these parameters, we aim to align the weights of regularization terms in the Lagrangian function within a similar range. This balance helps in ensuring that no single term overly dominates the optimization process, which might lead to an undesired solution.

1) Convergence of Mixed-Integer ADMM

The conditions under which Algorithm 1 is guaranteed to converge to a stationary point of the augmented Lagrangian $\mathcal{L}_\rho$ (7) are [17], [43], [44]:

1) (Coercivity). The objective function $\mathcal{F}(\hat{\mathrm{p}}) + \frac{\sigma}{2}||\mathrm{r}||_2^2 + \frac{\omega}{2}||B\cdot\mathrm{z} - H||_2^2$ is coercive over the constraint $A_1\cdot\mathrm{y} + -A_2\cdot\mathrm{z} + A_3\cdot\mathrm{r} = 0$: This condition holds since the term $\frac{\sigma}{2}||\mathrm{r}||_2^2$ is quadratic and all other generation scheduling variables are bounded.
2) (Feasibility). $\mathrm{Im}(A_T) \subseteq \mathrm{Im}(A_3)$, where $A_T = [A_1, A_2]$: Since $\mathrm{Im}(A_3)$ is the entire space, this condition holds for any $\mathrm{Im}(A_T)$.
3) (Lipschitz subminimization paths). It is possible to have a constant $M > 0$ at iteration m such that:
$$||\mathrm{y}^{m-1} - \mathrm{y}^m|| \leq M||A_1\mathrm{y}^{m-1} - A_1\mathrm{y}^m||. \tag{8}$$
This condition holds by setting M equal to 1. Note that the same condition is true for variables z and r with $M = 1$.
4) (Objective regularity). The objective $\mathcal{F}(\hat{\mathrm{p}}) + \frac{\sigma}{2}||\mathrm{r}||_2^2$ is a lower semi-continuous function: Since $\mathcal{F}(\hat{\mathrm{p}})$ is a sum of convex functions and an indicator function of a convex set, it is restricted prox-regular [17]. Also with a constant σ, $\frac{\sigma}{2}||\mathrm{r}||_2^2$ is Lipschitz differentiable. Therefore, this condition holds.

Also, this algorithm converges to the global optimum if $\mathcal{L}_\rho$ is Kurdyka–Łojasiewicz (KŁ) function [45], [46]. Function (7) satisfies this condition since it is a semi-algebraic

function. Therefore, by using Algorithm 1, $\mathcal{L}_\rho$, which is a soft-constrained version of the problem (1), will converge to a stationary point for a large enough $\rho > (\sigma, \omega)$. The algorithm assumes the non-QUBO and QUBO subproblems to be solved optimally at any iteration, regardless of whether a classical solver or the QAOA method is employed. Also, interested readers in the above conditions under which problem (6) converges to a global optimum are referred to [17], [43], [45], [46] for further details.

2) Discussion on multi-block ADMM

We note that fixing r to zero and skipping the second block update turns out to be a two-block implementation of ADMM. However, including variable r has two advantages. First, to decompose the linear constraint (6c), a three-block implementation is required, and the second block is an identity matrix whose image represents the entire space. It means for any fixed y and z, there always exists an r such that satisfies (6c), and the feasibility of the problem is guaranteed. Second, constraint (6d) can be handled separately from (6c) and return a convex and Lipschitz differentiable term that can be included in the objective function as $\frac{\sigma}{2}||\mathrm{r}||_2^2$. Numerical evidence presented by [17] shows that, in some cases, a two-block implementation of ADMM may converge more quickly than its three-block counterpart. However, the 2-block ADMM is prone to non-convergence and adheres to the local optimality in large problems.

B. DISTRIBUTED COORDINATION

A power system comprises several subsystems with their local computing processors. Power system problems need solution algorithms that can deal with the growing complexity of the system, protect entities' privacy, and provide a solution in a reasonable period of time [47]. Meanwhile, near-term quantum computers cannot centrally handle large problems due to their limited qubits. This section addresses a distributed QADMM strategy adaptive to the practical implementation of generation scheduling.

Consider a system equipped with a two-way communication infrastructure enabling data exchange between a system coordinator and i subsystems. A local CPU and QPU are embedded in each subsystem and can optimize, control, and coordinate the operation cycle of their generation units. The system coordinator also has a CPU to coordinate the subsystems. Each subsystem tries to schedule its generation to the system by solving its QUBO and non-QUBO optimizations, while the system coordinator coordinates the generation statuses to meet the system's physical limitations. The distributed coordination process can be outlined as follows:

Step 1: given $(\mathrm{z}^{(m-1)}, \mathrm{r}^{(m-1)}, \hat{\mathrm{p}}^{(m-1)})$, system coordinator updates the dual variable $\lambda^{(m)}$ and send it to subsystems.

Step 2: Solve the first, second, and third optimization blocks. QUBO subproblems are solved using a QPU to find $\mathrm{z}^{(m)}$ and pass them to CPU for updating $\mathrm{r}^{(m)}$ and $\hat{\mathrm{p}}^{(m)}$ by solving second and third optimization blocks.

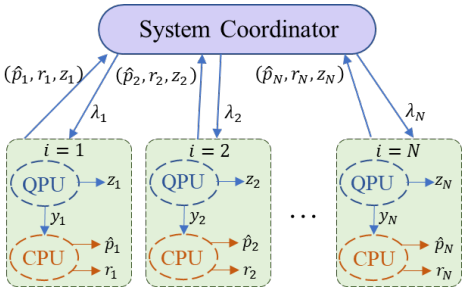

FIGURE 4. CPU and QPU information flow in distributed QADMM.

Step 3: Stop if termination criteria are satisfied, otherwise, go to Step 1.

Generally, generation scheduling has decomposed over units and yields binary and continuous subproblems coordinated through adjusting Lagrangian multipliers iteratively. The continuous subproblems are solved in a classical computer using linear programming methods. The binary subproblems are mapped into Ising models by relaxing constraints, adding their penalty terms to the objective functions, and converting them into QUBO. The derived QUBO subproblems optimize the units' on/off decisions and are solved on QPUs. The continuous subproblems optimize the units' power generation level and are solved on CPUs.

C. QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

QAOA finds the solution to the previously developed QUBO block by minimizing the expected value of its Hamiltonian. This expectation function is derived from quantum states that entangle all possible states in the same probability. The optimal expected value of the Hamiltonian happens by optimizing the rotation angles of quantum gates in a classical solver.

QAOA acts on n qubits, i.e., 2^n dimensional Hilbert space, with each qubit representing the state of a binary variable. The quantum process begins with initializing all qubits at state $|0\rangle$, and then making an equal superposition of all computational basis states by applying a Hadamard gate, $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$, to each qubit as:

$$|+\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle. \tag{9}$$

Given the problem Hamiltonian $\mathcal{H}_c$, QAOA applies a parameterized unitary operator $U(\mathcal{H}_c, \gamma)$, called cost Hamiltonian, depending on angle γ, and then makes a rotation of the resulting state using $U(\mathcal{H}_B, \beta)$, called mixing Hamiltonian, depending on angle β.

$$|\psi(\vec{\gamma}, \vec{\beta})\rangle = U(\mathcal{H}_c, \gamma_p)U(\mathcal{H}_B, \beta_p)...U(\mathcal{H}_c, \gamma_1)U(\mathcal{H}_B, \beta_1)|+\rangle^n, \tag{10}$$

where these operators are formulated as:

$$U(\mathcal{H}_c, \gamma) = e^{-i\gamma\mathcal{H}_c} = \prod_{(k,j)\in edges} e^{-i\gamma\mathcal{H}_c^{(k,j)}}, \tag{11}$$

$$U(\mathcal{H}_B, \beta) = e^{-i\beta B} = \prod_{k=1}^{N} e^{-i\beta X_k}, \tag{12}$$

where variational parameters $\gamma \in [0, 2\pi]$ and $\beta \in [0, \pi]$, or so-called angles [5], denote the evolution times of the quantum circuit and are utilized to construct a parameterized QAOA circuit. X_k refers to the Pauli-X operator, $B = \sum_{k=1}^{N} X_k$, and p represents the depth of the circuit, i.e., the number of layers for which parameterized $U(\mathcal{H}_c, \gamma)$ and $U(\mathcal{H}_B, \beta)$ are repeated. At every layer, the number of cost Hamiltonian depicts edges, and the number of mixing Hamiltonian depicts the number of vertices of graph G. Given the depth of the circuit, $2p$ angles, i.e., $\gamma = [\gamma_1, ..., \gamma_p]$ and $\beta = [\beta_1, ..., \beta_p]$, need to be tuned to make QAOA yield the optimal result. In practice, a trade-off determines the number of layers p between the obtained approximation ratio, the parameter optimization complexity, and accumulated errors. Ideally, increasing p enhances the QAOA solution quality, although the complexity of optimizing QAOA parameters with higher p limits its benefits. QAOA is generally implemented in a quantum circuit as Fig. 5.

After preparing the state $|\psi(\vec{\gamma}, \vec{\beta})\rangle$, another important component of QAOA is to calculate the expectation value of $\mathcal{H}_c$ in the state of $|\psi(\vec{\gamma}, \vec{\beta})\rangle$. The expectation value is defined as:

$$F(\vec{\gamma}, \vec{\beta}) = \langle \psi(\vec{\gamma}, \vec{\beta}) | \mathcal{H}_c | \psi(\vec{\gamma}, \vec{\beta}) \rangle. \tag{13}$$

QAOA minimizes the expectation value by updating the quantum state $|\psi(\vec{\gamma}, \vec{\beta})\rangle$ using a classical computer such that the expected function (13) is minimized to obtain the optimal values of $\vec{\gamma}$ and $\vec{\beta}$ denoted as $\vec{\gamma}^*$ and $\vec{\beta}^*$.

$$(\vec{\gamma}^*, \vec{\beta}^*) = \arg\min_{\vec{\gamma}, \vec{\beta}} F(\vec{\gamma}, \vec{\beta}) \tag{14}$$

As shown in Fig. 5, variational parameters γ and β are prepared in a classical computer and fed to the quantum circuit iteratively until convergence. Thus, the entire QADMM process involves two loops, as shown in Fig. 3. The outer loop performs ADMM and incorporates QAOA, and the inner loop is a hybrid quantum-classical process. A good initialization and optimization process will lead to fewer iterations to find the optimal variational parameters. An overview of the QAOA steps is provided in Algorithm 2. The optimal variational parameter finding strategy introduced in (13)-(14) is the approach presented by the original QAOA paper [5]. The QAOA algorithm 2 terminates when it fulfills a predefined termination criterion. In a typical QAOA workflow, this criterion is based on the results of the classical optimization loop used to tune the quantum circuit parameters. Setting a maximum iteration, convergence threshold, and time limit are the most common termination criteria for Algorithm 2. After termination, the best solution found during the iterations is returned. Due to the probabilistic nature of QAOA and quantum computing in general, the returned solution may not be the absolute optimal solution, but rather a good approximation to it. Typically, different optimizers require hundreds

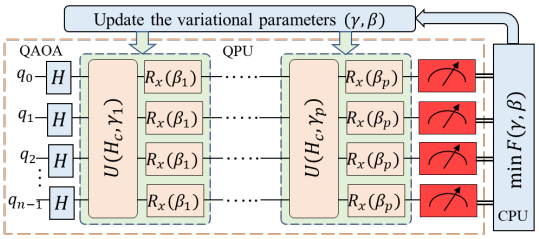

FIGURE 5. Quantum circuit diagram for QAOA.

to possibly thousands of quantum-classical iterations to reach a comparable parameter landscape optimum. Also, the classical optimizer requires more time to obtain the results at every iteration as the size of the expectation function (14) increases. These are the major QAOA bottlenecks, which are resolved in the next section.

Algorithm 2 QAOA steps

1: Initialize the quantum state by applying the Hadamard operator.
2: Model the cost Hamiltonian using the QUBO function.
3: Model the mixing Hamiltonian.
4: Create the circuits.
5: Run and measure the final state.
6: Update the variational parameters γ and β.
7: Go to step 2.

D. QAOA CIRCUIT DESIGN

Designing a QAOA circuit, as demonstrated in Fig. 5, entails several steps. Central to this process is the construction of the cost Hamiltonian and the mixing Hamiltonian, which significantly dictate the structure of the QAOA circuit. The cost Hamiltonian encodes the problem that we want to solve, and its eigenvalues correspond to the cost function we aim to optimize. For a standard combinatorial optimization problem, after translating the problem into a QUBO or Ising model, the coefficients and constants from this would form the terms in the cost Hamiltonian. The mixing Hamiltonian enables superpositions of states within the Hilbert space, thereby facilitating a comprehensive exploration of the problem space. Here are general steps to design a QAOA circuit:

1) Problem formulation: Identify and formulate the combinatorial optimization problem as a QUBO or Ising model.
2) Cost Hamiltonian $U(\mathcal{H}_c, \gamma)$: The cost Hamiltonian encodes the optimization problem, and it typically comprises a combination of Pauli-Z gates and identity gates. This is because many optimization problems can be expressed as a sum of multi-qubit Z terms when formulated as an Ising model or QUBO problem. Depending on the problem, it might need to use controlled-Z gates or more complex operations like multi-controlled gates. To implement a term in the Hamiltonian like $s_i s_j$ (representing a coupling between spins i and j in

the Ising model), it could use a circuit consisting of a Hadamard gate on qubit j, a controlled-Z gate using qubits i and j, and another Hadamard gate on qubit j.

3) Mixing Hamiltonian $U(\mathcal{H}_B, \beta)$: The mixing Hamiltonian drives transitions between different states in the Hilbert space. This is typically implemented with Pauli-X gates acting on each qubit to perform a bit-flip operation. For more complex QAOA variations, the mixing Hamiltonian can be composed of other gates such as Y, or a combination of X and Y gates.
4) Prepare the initial state: This state is usually a simple, quickly prepared state such as the uniform superposition of all computational basis states. This is achieved by applying a Hadamard gate to each qubit initialized to $|0\rangle$ state.
5) Apply the QAOA circuit: This involves applying $U(\mathcal{H}_c, \gamma)$ and $U(\mathcal{H}_B, \beta)$ alternately for a total of p layers.

Parameters γ and β are optimized using classical computational resources to find the set of parameters that minimizes the expectation value of the cost Hamiltonian. This is an iterative process where the parameters are adjusted, the QAOA circuit is run (either on a quantum computer or simulator), the expectation value of the Hamiltonian is calculated, and the process repeats with new parameters. These steps provide a basic guide to designing a QAOA circuit. The specifics of the problem formulation and Hamiltonian definition depend on the problem to solve.

E. TRAINABLE-QAOA (T-QAOA)

We aim to train a learner to play the role of an optimizer for QAOA to update the variational parameters. The expected value $F(\vec{\gamma}, \vec{\beta})$ of problem Hamiltonian is used as the cost function with respect to parameterized state $|\psi(\vec{\gamma}, \vec{\beta})\rangle$ evolved from the QAOA circuit. To choose an optimizer architecture, the QAOA cost function and parameters evaluations are translated over several quantum-classical iterations as a sequential learning problem. Recurrent neural networks (RNNs) are a type of neural network commonly used to process such sequential information [48]. RNNs are networks that take an input vector, create an output vector, and possibly store some information in memory for later use. A particular type of RNN framework that is used for the problem at hand is long-short-term memory (LSTM) which has outperformed other RNN architectures in many applications [49].

Fig. 6 shows the structure of the proposed T-QAOA. At an iteration ν, variational parameters $(\gamma, \beta)_{\nu-1}$, the estimated cost function $z_{\nu-1}$, and the hidden state of the classical network $h_{\nu-1}$ are fed to LSTM from the prior step. LSTM has its trainable hyperparameters ϕ and employs a generalized mapping as:

$$h_{\nu+1}, (\gamma, \beta)_{\nu+1} = LSTM_{\phi}(h_{\nu}, (\gamma, \beta)_{\nu}, z_{\nu}), \tag{15}$$

which suggests new variational parameters and a new internal hidden state. After training the weights ϕ, the new set of generated variational parameters is sent to QPU for evaluation. This loop continues upon convergence. To generate the first query, we arbitrarily fix the variational parameters to a dummy value $(\gamma, \beta)_0 = (0, 0)$, implement the QAOA circuit, and set the cost function to the obtained z_0.

It is important to select an appropriate loss function during training to measure the LSTM performance on the training dataset. We use another loss function $\mathcal{L}(\phi)$ as:

$$\mathcal{L}(\phi) = \mathbf{w}.\mathbf{z}_{\nu}(\phi) \tag{16}$$

which is called "cumulative regret" and is the summation of loss function history at all iterations uniformly averaged over the horizon. $\mathbf{w}$ are coefficients that weigh the progression of the recurrence loop. We set a higher weight for the last steps as they contain more important information. This way, during the first steps of optimization, LSTM is freer to explore a larger section of parameter space, whereas, towards the end, it is restricted to choosing an optimal solution. The LSTM optimizer is trained to run for a fixed number of iterations. However, it is possible to allow it to optimize for more iterations than it was initially trained, but later iterations may have weak performance.

The merit of the proposed LSTM optimizer over other learning-based approaches is its scalability, which our problem desperately needs. Since QADMM is sequential, it returns the same QUBO problem at every iteration with only some coefficients updated. It means the QPU faces a slightly different QUBO problem at every ADMM iteration, such that the QAOA circuit remains the same, but the cost Hamiltonian changes, and the parameters of the gates in the circuit will need to be adjusted accordingly. As such, the optimizer should be scalable so that a single set of training data will cover the entire process. Moreover, every standard optimizer follows a step-by-step path to update the variational parameters iteratively. Applying a learner only to predict the optimal variational parameters might yield a warm starting point for variational parameters. LSTM-based learners can be trained to find the paths every standard optimizer might take to reach the optimal values in a few iterations. Therefore, even if their results are not optimal, since they interact with the quantum circuit through predicting process, they provide more efficient starting points for other optimizers that execute local searches.

Another challenge is preparing a dataset to train a learning-based optimizer for large-scale systems. As established in [50], for an LSTM-based optimizer, the training dataset can be driven from the same system with fewer ranges of qubits. Therefore, for a power system with thousands of units, we can train an LSTM optimizer using the dataset driven from a subset of the given system. In addition, the power system topology is prone to change over time by adding or removing lines. In this case, a trained LSTM optimizer is expected to work based on the previous dataset, as it is not sensitive to a slight change in the grid topology. To generate the training dataset, given $\mathcal{H}_c$ and the fixed number of required qubits, we sample random values of coefficients J_{kj} and h_k using

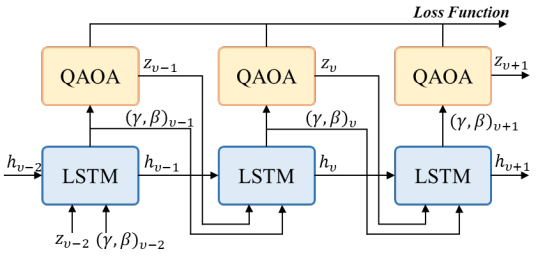

FIGURE 6. Unrolled trainable QAOA diagram.

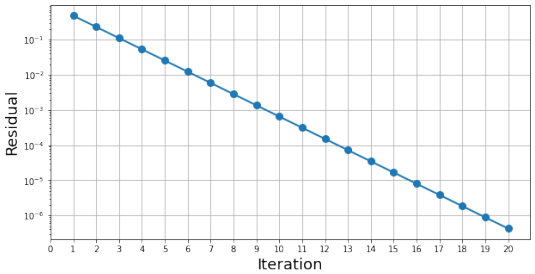

FIGURE 7. QADMM residual for illustrative example 1.

independent Gaussian distributions with zero mean and unit variance. Then, the Ising model circuit is constructed using (10) for the sampled Hamiltonian.

F. DISCUSSION ON T-QAOA SOLUTION QUALITY

Using a learner such as LSTM to update the variational parameters in each iteration of the QAOA algorithm, instead of solving an optimization problem directly, introduces a different approach to optimizing the parameters. In this case, LSTM would learn to predict the optimal parameters based on the given inputs and the desired objective. The quality of the solution obtained using this approach depends on several factors, including the effectiveness of the LSTM model, the amount and quality of training data, and the complexity of the problem being solved. The LSTM model would need to be trained on a suitable dataset that includes inputs (e.g., problem instances) and corresponding outputs (e.g., optimal parameter values) to learn the relationship between the inputs and desired parameter values. The quality of the solution obtained using the LSTM approach would depend on the accuracy and generalization capabilities of the trained model. It's important to note that this approach may have limitations. The performance of LSTM or any other machine learning model is subject to factors such as the availability and representativeness of training data, the complexity of the problem, and the suitability of the model architecture. Additionally, the LSTM-based approach may not guarantee optimal solutions but rather approximate solutions based on the learned predictions. The solution quality achieved using an LSTM-based approach within the QAOA algorithm would require thorough evaluation, including comparisons with other optimization methods, benchmarking against known problem instances, and sensitivity analysis.

IV. SIMULATION

Two illustrative mathematical examples and three generation scheduling problems are used to validate the performance of QADMM and the LSTM optimizer. A study of the variational parameters will also be conducted. In the first example, we use a typical MILP problem to show the convergence and correctness of QADMM. In the second example, we use a QUBO problem to study the effect of variational parameters and validate the LSTM optimizer performance by comparing it with a standard optimization technique based on stochastic gradient descent (SGD). Also, the performance of QADMM and LSTM optimizer are evaluated on a 3-unit, 3-hour generation scheduling example. The results of QADMM are compared with the classical implementation of 3-block ADMM, and the LSTM optimizer performance is compared with SGD. To test the model's scalability, a 24-hour generation scheduling problem for 10- and 100-unit systems is solved, and the proposed LSTM optimizer is used to optimize the variational parameters. A noise-free quantum simulator (statevector) is used in combination with IBM's Qiskit [51], Terra, and IBMQ providers. Neural network training and inference are conducted in Keras and TensorFlow [52].

A. ILLUSTRATIVE MATHEMATICAL EXAMPLES

1) Example 1

We examine an example of QADMM's performance under simple setups. Consider the following problem [17]:

$$\min_{x\in\chi, y\in\mathbb{R}} x_1 + x_2 + x_3 + 5(y-2)^2, \tag{17a}$$

s.t.

$$x_1 + x_2 = 1, \tag{17b}$$

$$x_1 + x_2 + x_3 \geq 1, \tag{17c}$$

$$x_1 + x_2 + x_3 + y \leq 3, \tag{17d}$$

The steps in Algorithm 1 are followed to solve (17). We initialize penalty factors as $\rho = 1001$, $\sigma = 1000$, and $\omega = 900$ [17]. According to the QADMM direction, we decompose the problem into a QUBO subproblem solved by a quantum computer and a continuous and quadratic unconstrained subproblem solved by a classical computer. Fig. 6 shows the reduction of the constraint's residual, defined as $r' = ||Iy - Iz + Ir||$ in Algorithm 1. The algorithm converges after 19 iterations and yields the optimal solution as $x = [1, 0, 0]$ and $y = 2$ with an optimality gap of zero.

2) Example 2

We examine how QAOA solves a simple QUBO problem. Then we apply the trained LSTM optimizer to facilitate the process and evaluate its performance. This is a MaxCut problem on a triangular graph with two edges weighing five and one weighing 1 (see Fig. 8). The objective function is the sum of weights for edges connecting nods between the two subsets, i.e., $C(x) = \sum_{k,j=1}^{n} J_{kj} x_k (1 - x_j)$ and $x \in \{0, 1\}$. To solve this problem on a quantum computer, we need to translate it into an Ising Hamiltonian form,

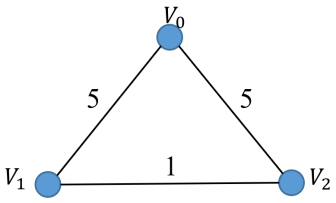

FIGURE 8. Graph of illustrative example 2.

$\mathcal{H}_c = \sum_{(k,j)\in E} \frac{1}{2} J_{kj}(1 - s_k s_j)$ and $s \in \{\pm 1\}$ by taking the relation $x_k \to (1 + s_k)/2$. We apply the steps explained in Algorithm 2. We create $U(\mathcal{H}_c, \gamma)$ and $U(\mathcal{H}_B, \beta)$ from the Ising Hamiltonian and build a one-layer circuit with two variational parameters as in Fig. 9, which includes the following steps:

1) Applying Hadamard gates to initial state $|q_2 q_1 q_0\rangle = |000\rangle$. This step prepares the initial state, i.e., an equal superposition of all possible states.

$$|\psi_0\rangle = \frac{1}{\sqrt{2^3}}(|000\rangle + |001\rangle + ... + |111\rangle). \quad (18a)$$

2) The controlled-phase gate is applied to the state $|\psi_0\rangle$. In this gate, a specific phase is introduced to the target qubit when the control qubit(s) are in the $|1\rangle$ state. To illustrate the effect of the controlled-phase gate in the first 2-qubit that includes two controlled-NOT gates and one controlled-phase gate, we derive the resulting state $|\psi_1^{'}\rangle$ as follows:

$$|\psi_1^{'}\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\gamma} \end{pmatrix} \right) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\gamma} & 0 & 0 \\ 0 & 0 & e^{-i\gamma} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \quad (18b)$$

In (18b), a phase shift of $-\gamma$ is applied to the two-qubit system. As per the first step of the algorithm, the state $|\psi_1^{'}\rangle$ is subjected to a Hadamard gate and the resulting $|\psi_1\rangle$ state can be derived using a two-qubit circuit as outlined below:

$$|\psi_1\rangle = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\gamma} & 0 & 0 \\ 0 & 0 & e^{-i\gamma} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad (18c)$$
$$= \frac{1}{2}(|00\rangle + e^{-i\gamma}|01\rangle + e^{-i\gamma}|10\rangle + |11\rangle).$$

In a similar way, considering a three-qubit quantum circuit depicted in Fig. 9, the state $|\psi_1\rangle$ is formulated as:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^3}}(|000\rangle + e^{-i\gamma}|001\rangle + e^{-3i\gamma}|010\rangle + ... + e^{-i\gamma}|110\rangle + |111\rangle). \quad (19)$$

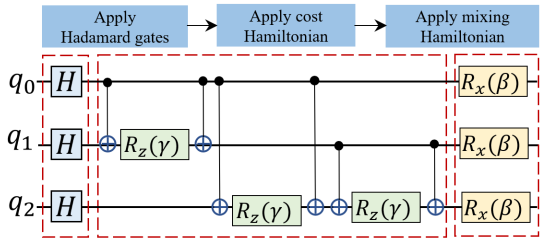

FIGURE 9. QAOA circuit of illustrative example 2.

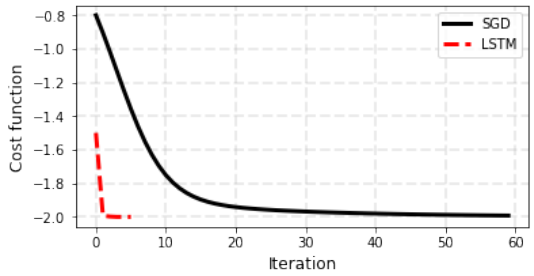

FIGURE 10. LSTM vs. SGD cost function of illustrative example 2.

3) A layer of R_x gates is implemented. Specifically, each qubit undergoes a rotation through $R_x(\beta) = \begin{pmatrix} \cos(\beta) & -i\sin(\beta) \\ -i\sin(\beta) & \cos(\beta) \end{pmatrix}$ before the measurement is taken. This results in a $-\beta$ rotation being applied to the state $|\psi_1\rangle$.

The measurement of the final state regarding the set of (γ, β) happens after applying the above steps. To address data passing between classical and quantum processors, we build a custom model of an LSTM network. The LSTM optimizer is trained for five iterations, i.e., the CPU and QPU exchange information five times. We set $\mathbf{w} = \frac{1}{5}(0.1, 0.2, 0.4, 0.6, 0.8)$, giving higher priority to the latter steps in the loop. One hundred data are randomly generated using Gaussian distribution with zero mean and unit variance given the problem's objective function and the number of required qubits. The trained LSTM optimizer performance is compared with SGD in Figs. 10 and 11. The variational parameters for both the LSTM optimizer and SGD are initialized to $(0, 0)$. LSTM terminates in fewer iterations than SGD. Fig. 10 illustrates the evolution of the cost function over iterative refinements of the variational parameters as proposed by the respective optimizers. Fig. 11 illustrates the path suggested by LSTM and SGD in the space of the parameters. Given the periodic nature of variational parameters, Fig. 11 shows more than one optimal solution. The LSTM optimizer yields the point $(\gamma, \beta) = (-0.31, 0.62)$ in 5 iterations, and the SGD yields $(\gamma, \beta) = (1.26, 0.62)$ in 60 iterations. γ provided by SGD is $\frac{\pi}{2}$ ahead of the one provided by LSTM optimizer. In the dataset used to train the LSTM, there are samples that (γ, β) converge to four different optimal points. LSTM optimizer learns the periodic behavior and offers the closest optimal point.

In gradient descent, the algorithm starts at an initial point in the parameter space (in our case, variational parameters

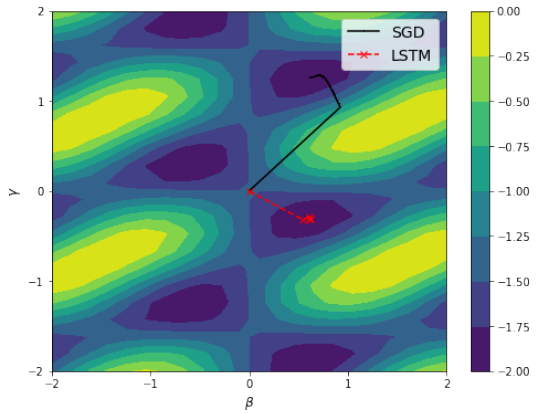

FIGURE 11. Example 2 expectation function contour plot.

are initialized to $(0, 0)$) and iteratively updates parameters to move in the direction of the steepest descent of the cost function. However, due to the stochastic nature of quantum computing, it is possible to observe non-monotonic behavior in the optimization process. This implies that the cost function might initially increase before starting to decrease toward the optimal value as we observe in Fig. 11. There are several reasons why this can happen:

- Noisy measurements: Quantum computations are susceptible to various kinds of errors and noise which might impact the measurement outcomes, leading to fluctuations in the cost function.
- Choice of optimizer: Different optimizers have different behaviors. For instance, some might allow for larger steps in the parameter space which could initially lead away from the minimum.
- Non-convex cost function: The cost functions in quantum computing problems, including QAOA, are usually non-convex, meaning they can have many local minima. The optimizer might temporarily get stuck in a less optimal minimum before eventually finding the global minimum.
- Initial parameters: The choice of initial parameters can influence the path that the optimization takes. If we start at $(0, 0)$, we might initially move in a direction that increases the cost function before eventually finding the correct path toward the minimum.

B. QADMM GENERATION SCHEDULING

This section verifies the correctness and effectiveness of the proposed QADMM and LSTM optimizer as applied to generation scheduling problems. QADMM is compared with the classical 3-block ADMM and the LSTM optimizer with the SGD optimization approach. We consider three scenarios for each case study:

- S1: Algorithm 1 is applied to problem (1), and the QUBO block is solved using a classical solver (Gurobi).
- S2: Algorithm 1 is applied to problem (1), the QUBO block is solved using Algorithm 2, and QAOA classical optimizer is SGD.

TABLE 1. Data of 3-Unit Generations

Unit	P_{min}	P_{max}	R_d	R_{up}	c_f	c_{su}	c_{sd}	b
1	50MW	350MW	300	200	5	20	0.5	0.1
2	80MW	300MW	150	100	7	18	0.3	0.125
3	40MW	140MW	100	100	6	5	1	0.15

- S3: Algorithm 1 is applied to problem (1), the QUBO block is solved using Algorithm 2, and QAOA classical optimizer is LSTM (the proposed algorithm).

Note that Algorithm 1's initializations are the same for all three scenarios.

1) 3-unit 3-hour generation scheduling

The system includes three units supplying a demand of 160 MW, 500 MW, and 400 MW at three consecutive hours. There are three subproblems for the problem decomposed over generating units. Each subproblem contains nine binary variables: units on/off, start-up, and shut-down status at time t ($3 \times 3 = 9$). The units' characteristics are given in Table I. An LSTM optimizer is trained for eight iterations with 800 observations and tested with 200 observations. The optimal status of units and the cost for each scenario are provided in Table II. The optimal on/off status and the operation cost obtained are the same for all scenarios. Fig. 12 portrays the reduction of the constraint residual for all scenarios at every ADMM iteration. The ADMM convergence performance is similar in all three cases. However, there are instances where both S2 and S3 diverged from S1. This could be related to the suboptimal solution of QUBOs after certain outer iterations, and another could be a variation in the configurations of the classical solvers that may induce such discrepancy. We note that perfect optimization or prediction of variational parameters is not a mandatory condition for the QAOA circuit to reach the optimal QUBO solution. Also, as problem size increases, sensitivity to optimal variational parameters also grows. In the context of this example, given its relatively small scale with 9 binary variables and efficient training of the LSTM, both the LSTM and SGD methods are capable of effectively approximating the optimal variational parameters. The residual approaches to the stopping criterion after 25 iterations. In $S2$ and $S3$, at every outer ADMM iteration, a hybrid quantum-classical interaction is conducted to find the optimal QAOA variational parameters. Though starting from the same initial point, the LSTM optimizer is trained to terminate after 8 interactions, while the SGD optimizer needs 86 iterations on average to converge. Fig. 13 shows the step-by-step moving toward an optimal point in $S2$ and $S3$ in the last ADMM iteration, which takes 65 iterations for SGD to converge.

2) Large-scale generation scheduling

A study of the performance of QADMM and the LSTM optimizer in large-scale generation scheduling problems is conducted. The number of considered units is 10 and 100, and the time horizon has been extended to 24 hours. In every

TABLE 2. Generation Scheduling Results and Operation Cost of *S1-S3* ($)

Unit	t_1	t_2	t_3	Cost *S1*	Cost *S2*	Cost *S3*
1	on	on	on	121	121	121
2		on		37.9	37.8	37.8
3		on	on	54	54	54
Total	-	-	-	212.9	212.8	212.8

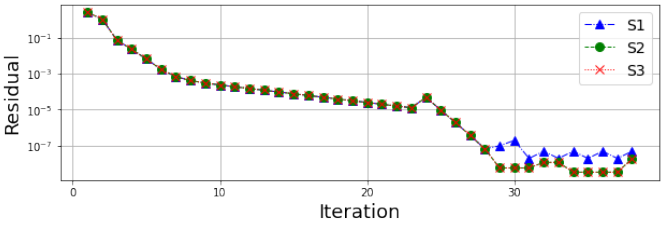

FIGURE 12. QADMM residual for 3-unit generation scheduling problem.

case, we have the same number of subproblems equal to the number of generating units. Each subproblem contains 72 binary variables: units on/off, start-up, and shut-down status at time t ($3 \times 24 = 72$). Two LSTM optimizers are trained for ten iterations with 800 observations and tested with 200 observations. The first LSTM is trained using a 72-qubit quantum circuit and a Gaussian distribution of coefficients for the Ising model. The second learner is trained using quantum circuits with a number of qubits in the range of [40, 50] and the same Gaussian distribution of coefficients. Table III shows the optimal on/off status of 5 selected generating units of the 100-unit case. The on/off status obtained by QADMM is the same as those of the classical central solution determined by Gurobi. Fig. 14 represents the reduction of the constraints residual for both 10-unit and 100-unit cases. As the system scales, QADMM maintains its convergence capabilities. Scaling up the system results in more subproblems and slower convergence. The 10-unit system converged after 77 iterations, and the 100-unit system converged after 126 iterations. A comparison between the SGD and LSTM optimizers is given in Fig. 15 for subproblem 1 in the last ADMM iteration. Both approaches started from the same initial point. LSTM optimizers terminate after ten iterations,

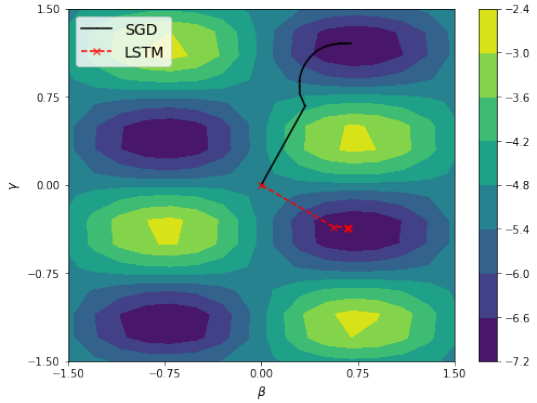

FIGURE 13. Contour plot of expectation function for 3-unit system.

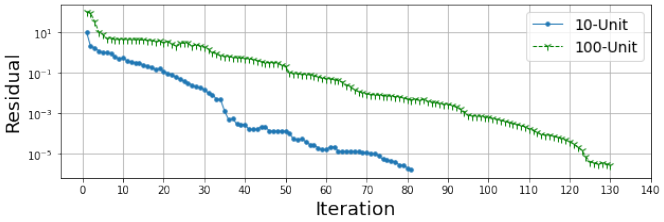

FIGURE 14. QADMM residual for 10- and 100-unit generation scheduling problems.

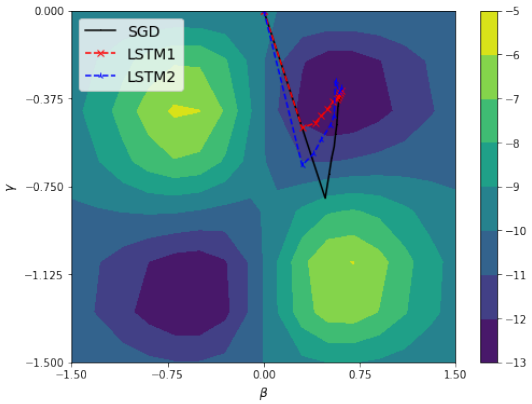

FIGURE 15. Contour plot of expectation function for 100-unit system.

while the SGD optimizer continues 84 iterations on average to converge. Furthermore, classical optimization takes more time to solve the problem as the dimension of expectation value (13) increases. A comparison between LSTM1, a well-trained optimizer, and LSTM2, a randomly-trained optimizer, is shown in Fig. 15. Though, from different paths, both optimizers obtain the optimal results after ten iterations.

V. CONCLUSION

This paper presents a scalable two-loop quantum-classical algorithm to solve the generation scheduling problem within the quantum computing framework. A three-block decomposition breaks generation scheduling into a QUBO and two non-QUBOs. A trainable QAOA solves the QUBO, and 3B-ADMM coordinates computation operations of quantum computer to solve the QUBO and conventional computer to solve non-QUBOs.

Simulation results on two mathematical and three generation scheduling problems show that T-QAOA terminates within a predetermined number of iterations, much fewer than that of the traditional QAOA. For instance, for a MaxCut problem, T-QAOA converges after five iterations, while the traditional QAOA with stochastic gradient descent converges after 60 iterations. For generation scheduling with 100 generators, T-QAOA takes ten iterations to find the optimal results obtained after 84 iterations of the traditional QAOA. Also, 3B-ADMM coordinates conventional and quantum computers computation operations to achieve optimal results of the original generation scheduling problem. The overall QADMM convergence residual is in the range of 10^{-6}, which is promising.

TABLE 3. On/Off Schedule of 5 Selected Units of the 100-Unit Problem

Unit	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}	t_{23}	t_{24}
1	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on
10							on	on	on				on	on	on	on	on	on	on					
45						on	on	on	on	on	on	on	on	on	on	on	on	on	on	on				
81		on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on	on
90							on	on	on	on			on	on	on	on	on	on	on					

REFERENCES

[1] J. Tovar-Facio, M. Martín, and J. M. Ponce-Ortega, "Sustainable energy transition: Modeling and optimization," Current Opinion in Chemical Engineering, vol. 31, p. 100661, 2021.

[2] A. J. Conejo and L. Baringo, Power system operations. Springer, 2018.

[3] S. Y. Abujarad, M. W. Mustafa, and J. J. Jamian, "Recent approaches of unit commitment in the presence of intermittent renewable energy resources: A review," Renewable and Sustainable Energy Reviews, vol. 70, pp. 215–223, 2017.

[4] A. A. Abd El-Latif, Y. Maleh, M. Petrocchi, and V. Casola, "Guest editorial: Advanced computing and blockchain applications for critical industrial iot," IEEE Transactions on Industrial Informatics, vol. 18, no. 11, pp. 8282–8286, 2022.

[5] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv preprint arXiv:1411.4028, 2014.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212–219, 1996.

[7] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," Nature communications, vol. 5, no. 1, pp. 1–7, 2014.

[8] E. Zahedinejad and A. Zaribafiyan, "Combinatorial optimization on gate model quantum computers: A survey," arXiv preprint arXiv:1708.05294, 2017.

[9] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," Journal of combinatorial optimization, vol. 28, no. 1, pp. 58–81, 2014.

[10] I. Kerenidis and A. Prakash, "A quantum interior point method for lps and sdps," ACM Transactions on Quantum Computing, vol. 1, no. 1, pp. 1–32, 2020.

[11] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of ising machines and a software development for ising machines," Journal of the Physical Society of Japan, vol. 88, no. 6, p. 061010, 2019.

[12] G. Nannicini, "Performance of hybrid quantum-classical variational heuristics for combinatorial optimization," Physical Review E, vol. 99, no. 1, p. 013304, 2019.

[13] C. C. McGeoch, "Theory versus practice in annealing-based quantum computing," Theoretical Computer Science, vol. 816, pp. 169–183, 2020.

[14] C.-Y. Chang, E. Jones, and P. Graf, "On quantum computing for mixed-integer programming," arXiv preprint arXiv:2010.07852, 2020.

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

[16] S. Boyd, N. Parikh, and E. Chu, Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, 2011.

[17] C. Gambella and A. Simonetto, "Multiblock admm heuristics for mixed-binary optimization on classical and quantum computers," IEEE Transactions on Quantum Engineering, vol. 1, pp. 1–22, 2020.

[18] N. Nikmehr, P. Zhang, and A. M. Bragin, "Quantum distributed unit commitment: An application in microgrids," IEEE Transactions on Power Systems, 2022.

[19] R. Mahroo and A. Kargarian, "Hybrid quantum-classical unit commitment," in 2022 IEEE Texas Power and Energy Conference (TPEC), pp. 1–5, IEEE, 2022.

[20] F. Feng, P. Zhang, M. A. Bragin, and Y. Zhou, "Novel resolution of unit commitment problems through quantum surrogate lagrangian relaxation," IEEE Transactions on Power Systems, 2022.

[21] A. Ajagekar and F. You, "Quantum computing for energy systems optimization: Challenges and opportunities," Energy, vol. 179, pp. 76–89, 2019.

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

[23] L. Braine, D. J. Egger, J. Glick, and S. Woerner, "Quantum algorithms for mixed binary optimization applied to transaction settlement," IEEE Transactions on Quantum Engineering, vol. 2, pp. 1–8, 2021.

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

[25] T. Morstyn, "Annealing-based quantum computing for combinatorial optimal power flow," IEEE Transactions on Smart Grid, 2022.

[26] L. P. Raghav, R. S. Kumar, D. K. Raju, and A. R. Singh, "Optimal energy management of microgrids using quantum teaching learning based algorithm," IEEE Transactions on Smart Grid, vol. 12, no. 6, pp. 4834–4842, 2021.

[27] C. Y.-Y. Lin and Y. Zhu, "Performance of qaoa on typical instances of constraint satisfaction problems with bounded degree," arXiv preprint arXiv:1601.01744, 2016.

[28] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, "Quantum approximate optimization algorithm for maxcut: A fermionic view," Physical Review A, vol. 97, no. 2, p. 022304, 2018.

[29] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer," Physical Review A, vol. 94, no. 2, p. 022309, 2016.

[30] M. Streif and M. Leib, "Training the quantum approximate optimization algorithm without access to a quantum processing unit," Quantum Science and Technology, vol. 5, no. 3, p. 034008, 2020.

[31] R. Shaydulin and Y. Alexeev, "Evaluating quantum approximate optimization algorithm: A case study," in 2019 tenth international green and sustainable computing conference (IGSC), pp. 1–6, IEEE, 2019.

[32] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," in Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 2367–2375, 2020.

[33] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[34] A. Steane, "Quantum computing," Reports on Progress in Physics, vol. 61, no. 2, p. 117, 1998.

[35] F. Amani, R. Mahroo, and A. Kargarian, "Quantum-enhanced dc optimal power flow," in 2023 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6, IEEE, 2023.

[36] Y. Zhou and P. Zhang, "Noise-resilient quantum machine learning for stability assessment of power systems," IEEE Transactions on Power Systems, 2022.

[37] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, "Variational quantum linear solver," arXiv preprint arXiv:1909.05820, 2019.

[38] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, et al., "Quantum optimization using variational algorithms on near-term quantum devices," Quantum Science and Technology, vol. 3, no. 3, p. 030503, 2018.

[39] S. Kumar, H. Zhang, and Y.-P. Huang, "Large-scale ising emulation with four body interaction and all-to-all connections," Communications Physics, vol. 3, no. 1, pp. 1–9, 2020.

[40] S. Boettcher, "Analysis of the relation between quadratic unconstrained binary optimization and the spin-glass ground-state problem," Physical Review Research, vol. 1, no. 3, p. 033142, 2019.

[41] C.-Y. Wang and D. Li, "Unified theory of augmented lagrangian methods for constrained global optimization," Journal of Global Optimization, vol. 44, no. 3, pp. 433–458, 2009.

[42] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," Mathematical and computational Applications, vol. 10, no. 1, pp. 45–56, 2005.

[43] K. Sun and X. A. Sun, "A two-level distributed algorithm for general constrained non-convex optimization with global convergence," arXiv preprint arXiv:1902.07654, 2019.
[44] R. Mahroo, A. Kargarian, M. Mehrtash, and A. J. Conejo, "Robust dynamic tep with an security criterion: A computationally efficient model," IEEE Transactions on Power Systems, vol. 38, no. 1, pp. 912–920, 2022.
[45] J. Bolte, A. Daniilidis, and A. Lewis, "The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems," SIAM Journal on Optimization, vol. 17, no. 4, pp. 1205–1223, 2007.
[46] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods," Mathematical Programming, vol. 137, no. 1, pp. 91–129, 2013.
[47] R. Mahroo-Bakhtiari, M. Izadi, A. Safdarian, and M. Lehtonen, "Distributed load management scheme to increase pv hosting capacity in lv feeders," IET Renewable Power Generation, vol. 14, no. 1, pp. 125–133, 2020.
[48] L. Sun, J. Du, L.-R. Dai, and C.-H. Lee, "Multiple-target deep learning for lstm-rnn based speech enhancement," in 2017 Hands-free Speech Communications and Microphone Arrays (HSCMA), pp. 136–140, IEEE, 2017.
[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[50] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas, "Learning to learn without gradient descent by gradient descent," in International Conference on Machine Learning, pp. 748–756, PMLR, 2017.
[51] IBM-Qiskit, "Available: https://qiskit.org/," IBM ILOG CPLEX Optimizer, 2021.
[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: a system for large-scale machine learning," in 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283, 2016.

REZA MAHROO (Student Member, IEEE) received the B.Sc. degree from the Iran University of Science and Technology, Tehran, Iran, in 2016, and the M.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 2019. He is currently working toward a Ph.D. degree with the Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA, USA. His research interests include optimization theory, quantum computing applications, and improving power systems analysis using artificial intelligence.

AMIN KARGARIAN (Senior Member, IEEE) is currently an Associate Professor with the Electrical and Computer Engineering Department, Louisiana State University, Baton Rouge, LA, USA. His research interests include optimization, machine learning, quantum computing, and their applications to power systems.