Research Article

A Quantum Approximate Optimization Algorithm with Metalearning for MaxCut Problem and Its Simulation via TensorFlow Quantum

Haibin Wang^{1,2}, Jiaojiao Zhao,² Bosi Wang,² and Lian Tong³

¹Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing 210044, China

²School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

³School of Information Engineering, Jiangsu Maritime Institute, Nanjing 211100, China

Correspondence should be addressed to Haibin Wang; whb9741705@163.com

Received 22 October 2020; Revised 20 November 2020; Accepted 4 March 2021; Published 25 March 2021

Academic Editor: Gonglin Yuan

Copyright © 2021 Haibin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A quantum approximate optimization algorithm (QAOA) is a polynomial-time approximate optimization algorithm used to solve combinatorial optimization problems. However, the existing QAOA algorithms have poor generalization performance in finding an optimal solution from a feasible solution set of combinatorial problems. In order to solve this problem, a quantum approximate optimization algorithm with metalearning for the MaxCut problem (MetaQAOA) is proposed. Specifically, a quantum neural network (QNN) is constructed in the form of the parameterized quantum circuit to detect different topological phases of matter, and a classical long short-term memory (LSTM) neural network is used as a black-box optimizer, which can quickly assist QNN to find the approximate optimal QAOA parameters. The experiment simulation via TensorFlow Quantum (TFQ) shows that MetaQAOA requires fewer iterations to reach the threshold of the loss function, and the threshold of the loss value after training is smaller than comparison methods. In addition, our algorithm can learn parameter update heuristics which can generalize to larger system sizes and still outperform other initialization strategies of this scale.

1. Introduction

With the rapid development of quantum computing, people pay more and more attention to using quantum computing to solve combinatorial optimization problems. The maximum cut problem (MaxCut) [1] is a typical NP-hard combinatorial optimization problem in graph theory. A promising approach to solve combinatorial optimization problems on near-term machines is the quantum approximate optimization algorithm (QAOA) [2]. This is a heuristic method, which can be considered as a time discretization of adiabatic quantum computing [3]. Like many quantum computing [4–6] methods, QAOA is a hybrid classical-quantum algorithm that combines quantum circuits and classical optimization of those circuits. The goal of QAOA is

a function of the quantum state, which in turn is a function parameterized by one- and two-qubit gates, and its parameters can be continuously varied. In 2014, Edward Farhi and Jeffrey Goldstone jointly published the QAOA for the MaxCut problem (MaxCut QAOA) algorithm [7]. This algorithm is a polynomial-time approximate optimization algorithm for solving combinatorial optimization problems and has the potential to demonstrate quantum hegemony. However, this classical algorithm could require space doubly exponential in p , where p is the time of unitary transformation. Later, a classical computer is used as the outer-loop optimizer of quantum neural network to find the best parameters, which then were fed to the quantum computer with different sets of parameters. In 2018, Gavin proposed an algorithm [8] that used classical simulation with automatic

differentiation [9] of parameters and stochastic gradient descent to explore the MaxCut QAOA problem. However, it is difficult to implement stochastic gradient descent directly on a quantum computer, since the requisite gradients are expensive to measure [10], requiring many observations for each gradient component. A variety of approaches have been used to optimize MaxCut QAOA circuits, including Nelder–Mead [11], gradient descent [12], limited-memory Broyden–Fletcher–Goldfarb–Shanno Bound (L-BFGS-B) [13], and Bayesian [14] methods. Among them, Nelder–Mead and L-BFGS-B are the local optimization methods, and gradient descent and Bayesian methods are the global optimization methods. However, these approaches are relatively poor in generalization. In other words, a classical neural network trained by these methods to optimize small quantum neural network (QNN) [15] cannot learn parameter update heuristics which can generalize to larger system sizes.

In order to solve these problems, we propose a new local optimization method, which is the quantum approximate optimization algorithm with metalearning for the MaxCut problem (MetaQAOA). This algorithm makes use of the quantum-classical metalearning approach composed of a group of metaoptimization techniques to learn how to modify the parameters of learning algorithms to further customize them for the MaxCut QAOA problems. This could be to ensure that the learning can be well generalized, to better fit the given data with fewer iterations, and to adapt a pretrained neural network model to a new task. Through the simulation experiment on TensorFlow Quantum (TFQ) [16], the convergence of the model shows that the total number of optimization iterations required to achieve a given accuracy has been significantly improved. The experiment simulation via TensorFlow Quantum (TFQ) shows that MetaQAOA requires fewer iterations to reach the threshold of the loss function, and the threshold of the loss value after training is smaller than the other local optimization methods, i.e., Nelder–Mead and L-BFGS-B. In addition, the path chosen by the long short-term memory (LSTM) optimizer on a larger MaxCut problem after being trained on many random smaller MaxCut problems demonstrates that even when there is less data, the neural network learns to generalize its parameter update heuristic to larger system sizes. This provides the possibility of training on small, classical, and simulatable MaxCut QAOA problem instances, in order to initialize larger, classically difficult to simulate MaxCut QAOA problem instances on quantum devices.

The rest of the paper is organized as follows: Section 2 describes the preliminaries about the classical LSTM neural network, metalearning, and the quantum approximate optimization algorithm for the MaxCut QAOA problem; Section 3 presents the design focus of MetaQAOA, including the description of the MaxCut problem, the process of algorithm, and the method of optimizing parameter; in Section 4, experiments are performed in the quantum cloud computing platform to verify the feasibility of the proposed MetaQAOA; Section 5 summarizes what we find in this work and the prospects for future research studies.

2. Preliminaries

2.1. Long Short-Term Memory (LSTM). In general, LSTM refers to long and short-term memory artificial neural network, which is a time loop neural network, and is used to solve the problem of long-term dependence. There are special units called memory blocks (shown in Figure 1) in the hidden layer of LSTM. In addition to special multiplicative units called gates to control the flow of information, the memory blocks contain memory cells with self-connections storing the temporal state of the network. Each memory block contained an input gate, an output gate, and a forget gate [17]. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the other part of the network. Because the forget gate can scale the cell before adding it as input to the cell through the self-connection of the cell, the cell's memory is forgotten or reset adaptively. In addition, the current LSTM architecture (as shown in Figure 2) contains peephole connections from its internal cells to the gates in the same cell to understand the precise timing of the outputs [18].

The LSTM calculates the activation of the network cell iteratively from $t = 1$ to T by making use of the following equations to calculate the mapping from the input sequence $x = (x_1, \dots, x_T)$ to the output sequence $y = (y_1, \dots, y_T)$:

$$\begin{aligned} i_t &= \sigma(w_{im}m_{t-1} + w_{ic}c_{t-1} + w_{ix}x_t + b_i), \\ f_t &= \sigma(w_{fm}m_{t-1} + w_{fc}c_{t-1} + w_{fx}x_t + b_f), \\ c_t &= f_t \odot c_{t-1} + i_t \odot g(w_{cm}m_{t-1} + w_{cx}x_t + b_c), \\ o_t &= \sigma(w_{om}m_{t-1} + w_{ox}x_t + w_{oc}c_t + b_o), \\ m_t &= o_t \odot h(f_t \odot c_{t-1} + i_t \odot g(w_{cm}m_{t-1} + w_{cx}x_t + b_c)), \\ y_t &= G(w_{ym}m_t + b_y), \end{aligned} \quad (1)$$

where w represents weight matrices, diagonal weight matrices for peephole connections are w_{ic} , w_{fc} , and w_{oc} , bias vector is b (the input gate bias vector is b_i), the logistic sigmoid function is σ , and the input gate, output gate, forget gate, and cell activation vectors are i , o , f , and c . They are all the same size as the cell output activation vector m , $\odot$ represents the element-wise product of the vectors, the input and output activation functions are g and h , and the network output activation function is G . In addition, LSTM neural networks are already deep architectures in the sense that they can be considered as a feed-forward neural network unrolled over time, where each layer shares the same parameters, which is shown in Figure 2.

2.2. Metalearning. Metalearning is to study how to design a machine learning model with less training set, so that it can learn well and quickly [19]. One specific example is a neural network model which learns how to optimize other neural network models with parameterized function. Metalearners can not only train machine learning models but also train and optimize general functions. In specific regions where

FIGURE 1: A single memory block.

FIGURE 2: LSTM architecture. The LSTM is unrolled over time.

models are used to optimize other models, early studies explored guided strategy search, which has been replaced by LSTMs [20]. The LSTM is a kind of recurrent neural network, which is used to reduce the vanishing or exploding gradients in other recurrent neural network structures [21, 22]. An LSTM is composed of a cell state, a hidden state, and a gate, all of which are called LSTM cell. In each time step, the cell state is changed based on the hidden state, gate, and data information entered into the LSTM cell. The change of the hidden state depends on the gate and the input, and then, the cell state and hidden state are passed to the LSTM cell in the next time step. A complete processing method for LSTM is given in [23]. An LSTM is instrumental in learning long-term dependencies, such as those in optimization.

Metalearners have been used in the rapid general optimization of the model, and there are few training examples

[24]: given the random initial parameters, we seek to quickly converge to the “optimal” general parameters defined by some measures. This same problem characteristic also appears in QAOA, and some common distribution across problems [25] may be followed by good parameters. A metalearner could be used to find general good parameters and fine-tuning left to some other optimizer [26].

2.3. The Quantum Approximate Optimization Algorithm for the MaxCut Problem. The MaxCut problem is a typical combinatorial optimization problem and an NP-hard problem. A promising approach to solve combinatorial optimization problems on near-term quantum machines is QAOA. The steps of the quantum approximate optimization algorithm for the MaxCut problem are as shown in Algorithm 1.

Input: initial state $|\psi_0\rangle, P, T, \gamma, \eta$, Hamiltonian H_M, H_C .
Output: final state $|\psi'\rangle$.

- (1) $|\psi_0\rangle = |+\rangle_1 |+\rangle_2 \cdots |+\rangle_n$;
- (2) random γ, η ;
- (3) $\hat{H}_M = \sum_{j \in \mathcal{V}} \hat{X}_j$;
- (4) $\hat{H}_C = \sum_{\{j,k\} \in E} (1/2) (\hat{I} - \hat{Z}_j \hat{Z}_k)$;
- (5) **for** $t = 1$ **to** T **do**
- (6) $\hat{U}(\eta, \gamma) = \prod_{q=1}^P e^{-i\eta_q \hat{H}_M} e^{-i\gamma_q \hat{H}_C}$;
- (7) $|\Psi_t\rangle = \hat{U}(\eta, \gamma) |\psi_0\rangle$
- (8) **end**
- (9) **return** $|\psi'\rangle = |\Psi_t\rangle$

ALGORITHM 1: The quantum approximate optimization algorithm for the MaxCut problem.

- (1) An initial state $|\psi_0\rangle$ is prepared, which is a uniformly distributed quantum superposition state.
- (2) The mixer Hamiltonian $\hat{H}_M$ and the cost Hamiltonian $\hat{H}_C$ are selected appropriately. The selected cost and mixer Hamiltonian need meet this condition that the mixer Hamiltonian is noncommuting with the cost Hamiltonian, i.e., $[\hat{H}_C, \hat{H}_M] \neq 0$.
- (3) The unitary $\hat{U}(\eta, \gamma)$ applied to this initial state is about the exponential form of mixer Hamiltonian $\hat{H}_M$ and cost Hamiltonian $\hat{H}_C$ and can be expressed in the form of a parameterized quantum circuit (PQC).
- (4) We can get the final state after T unitary transformation, as shown in Algorithm 1.

A variety of approaches have been used to optimize PQC, including the local optimization methods (Nelder–Mead [11] and L-BFGS-B [13]) and the global optimization methods (gradient descent [12] and Bayesian [14]). However, these local optimization approaches are relatively poor in generation. In order to enable the classical neural networks trained by these methods to optimize small QNNs, it is possible to learn parameter update heuristics that can be extended to larger system sizes. In this case, we propose a new local optimization method, which is a quantum approximate optimization algorithm with metalearning for the MaxCut problem.

3. The Quantum Approximate Optimization Algorithm with Metalearning for the MaxCut Problem

The main purpose of the quantum optimization algorithm with metalearning for the MaxCut problem proposed in this paper is to train a classical long short-term memory (LSTM) network as an outer-loop optimizer to directly optimize the parameters in the QNN. In this section, we mainly introduce the process and training strategy of the MetaQAOA.

3.1. MaxCut Problem. The MaxCut problem is to find a split $(S, V - S)$ of the vertex set V so that the maximum number of edge weights is divided, i.e., one vertex of the edge is in S

and the other vertex is in $V - S$. This is a typical combination problem. Given a graph $G = (n, \varepsilon)$ with nodes n and edges $\{j, k\} \in \varepsilon$, find a subset $S \subseteq n$ to maximize the number of edges between S and $S \setminus n$. This problem can be attributed to finding the ground state of an Ising model.

The simplest method is the enumeration method, which is the only method that can find the global optimal solution (no approximation). However, this method is more complex, especially for the case of numerous vertices. Because when MaxCut only divides the vertices into two groups, there are a total of 2^n methods. Therefore, the MaxCut problem is an NP-hard problem.

3.2. The Quantum Approximate Optimization Algorithm with Metalearning for the MaxCut Problem. The quantum approximate optimization algorithm with metalearning for the MaxCut problem (MetaQAOA) we proposed will be to train an optimizer network, i.e., metalearner, with metalearning to learn parameter update heuristics for QNN constructed in the form of the parameterized quantum circuit. The MetaQAOA is a local optimizer, which has a notion of location in the solution space. It can search for candidate solutions from this location. Besides, MetaQAOA is usually fast and is susceptible to finding local minima. Training an outer-loop optimizer on many instances of MaxCut problem to enhance the efficiency of solving unseen instances is the idea of metalearning for optimization. This may be to ensure that the learning is well generalized and that the model can better fit the given data with fewer iterations. The algorithm we proposed mainly improves the update method of parameters in parameterized quantum circuits, and the steps of the proposed algorithm are shown in Algorithm 2.

The main steps of our algorithm are as follows. We first prepare a uniformly distributed quantum superposition state as an initial state $|\psi_0\rangle$:

$$|\psi_0\rangle = |+_1\rangle |+_2\rangle \cdots |+_n\rangle, \quad (2)$$

where n represents the number of nodes in the graph G for the MaxCut problem and $|+\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle)$ is a superposition state.

Then, we select the mixer Hamiltonian $\hat{H}_M$ and the cost Hamiltonian $\hat{H}_C$ as follows:

Input: initial state $|\psi_0\rangle, P, j, k, T, \nu, \varepsilon, h, \gamma, \eta, \varphi$, Hamiltonian H_M, H_{C_1}, H_{C_2} .

Output: final state $|\psi'\rangle$.

- (1) $|\psi_0\rangle = |+\rangle_1 |+\rangle_2 \cdots |+\rangle_n$;
- (2) random γ, η, φ ;
- (3) $h = 0$;
- (4) $\hat{H}_M = \sum_{j \in \mathcal{V}} \hat{X}_j$; $\hat{H}_{C_1} = \sum_{\{j,k\} \in E} (1/2) (\hat{I} - \hat{Z}_j \hat{Z}_k) \rightarrow \hat{H}_{C_2} = \sum_{\{j,k\} \in E} \hat{Z}_j \hat{Z}_k$;
- (5) **for** $t = 1$ **to** T **do**
- (6) $\hat{U}(\eta, \gamma) = \prod_{q=1}^P e^{-i\eta_q \hat{H}_M} e^{-i\gamma_q \hat{H}_{C_2}}$
- (7) $|\Psi_t\rangle = \hat{U}(\eta, \gamma) |\psi_0\rangle$
- (8) $y_t = \langle H_{C_2} \rangle = \langle \Psi_t | H_{C_2} | \Psi_t \rangle$
- (9) $h_{t+1}, \eta_{t+1}, \gamma_{t+1} = \text{LSTM}_\varphi(h_t, \eta_t, \gamma_t, y_t)$
- (10) **end**
- (11) **return** $|\psi'\rangle = |\Psi_t\rangle$;

ALGORITHM 2: The quantum approximate optimization algorithm with metalearning for the MaxCut problem.

$$\hat{H}_M = \sum_{j \in n} \hat{X}_j, \quad (3)$$

$$\hat{H}_{C_1} = \sum_{\{j,k\} \in E} \frac{1}{2} (\hat{I} - \hat{Z}_j \hat{Z}_k), \quad (4)$$

where $\hat{H}_M$ is the sum of Pauli $\hat{X}$ on each qubit, $\hat{H}_{C_1}$ is diagonal in the computational basis, I, X, Z represent Pauli I, X, Z operator, respectively, ε is the edge of the graph G , and j, k are the two nodes connected by the edge.

The reason why we choose this mixer Hamiltonian is that each item is a noncommuting cost Hamiltonian, i.e., $[\hat{H}_C, \hat{H}_M] \neq 0$, and it can be exponentiated with minimal gate depth. One traditional family of Hamiltonian used in QAOA is the Ising models, and thus, we convert the cost Hamiltonian into Ising Hamiltonian $\hat{H}_{C_2}$, as shown in Algorithm 2. Therefore, equation (4) can be written as follows:

$$\hat{H}_{C_2} = \sum_{\{j,k\} \in E} \hat{Z}_j \hat{Z}_k. \quad (5)$$

Thirdly, the unitary $\hat{U}(\eta, \gamma)$ applied onto this initial state is given by

$$\hat{U}(\eta, \gamma) = \prod_{q=1}^P e^{-i\eta_q \hat{H}_M} e^{-i\gamma_q \hat{H}_{C_2}}, \quad (6)$$

where P is the number of times the unitary transformation is executed and the difference is that the parameters $\{\eta, \gamma\}$ are different at each time step. These parameters at each time step are as follows:

$$\begin{aligned} \vec{\eta} &= (\eta_1, \eta_2, \dots, \eta_P)^T, \\ \vec{\gamma} &= (\gamma_1, \gamma_2, \dots, \gamma_P)^T. \end{aligned} \quad (7)$$

This unitary transformation with parameters expressed in the form of a general parameterized quantum circuit is shown in Figure 3. The initial state of the circuit is $|0\rangle$ and Hadamard gate is performed on each qubit to prepare entangled state. Although these unitaries do not necessarily

act on all qubits, we arrange them as “blocks” here, where “block” operations may be repeated P times with different parameters in the circuit. Finally, the expected value is obtained by measurement.

A QNN is constructed in the form of a parameterized quantum circuit. A single time step of MetaQAOA is shown in Figure 4. An iteration starts from the LSTM sending the set of candidate parameters $\{\vec{\eta}, \vec{\gamma}\}$ to the QNN. Then, the QNN executes a parameterized circuit $\hat{U}(\eta, \gamma)$, and it is responsible for generating the state $|\Psi(\eta, \gamma)\rangle$. The purpose of measuring this state is to extract relevant information, i.e., the expectation value of the cost Hamiltonian. The classical subroutine suggests parameters $\{\eta, \gamma\}$ based on the values provided by the quantum computer and sends the new parameters back to the quantum device.

This process is repeated T times until the given goal is reached, as shown in Algorithm 2. In other words, the quantum-classical optimization loop can be unrolled into a single temporal quantum-classical computational graph over time, as shown in Figure 5. More precisely, at the t^{th} optimization iteration, the LSTM receives the previous QNN query's estimated cost function expectation $y_t = p(y|\{\eta_t, \gamma_t\})$, where y_t is the estimate of $\langle \hat{H} \rangle_t$ and the parameters $\{\eta_t, \gamma_t\}$ represent input, as shown in Algorithm 2. At this time step, LSTM also receives information h_t stored in its internal hidden state from the previous time step. The LSTM itself has trainable parameters φ , so the parameterized mapping it applies is

$$h_{t+1}, \eta_{t+1}, \gamma_{t+1} = \text{LSTM}_\varphi(h_t, \eta_t, \gamma_t, y_t), \quad (8)$$

where h_t is the information stored in the internal hidden state, η_t, γ_t represent the parameters of QNN, y_t is the expectation value of the cost function at t time step, and φ is the trainable of the LSTM. Besides, h_{t+1} represents a new hidden state and η_{t+1}, γ_{t+1} are the parameters of the QNN generated by LSTM at $t + 1$ time step.

Once a new set of QNN parameters is proposed, the LSTM will send it to the QNN for evaluation, and then the loop will continue.

Finally, the final state is

FIGURE 3: Parameterized quantum circuit, with unitaries $U_j(\eta_j, \gamma_j) = e^{-i\eta_j \hat{H}_M} e^{-i\gamma_j \hat{H}_C}$, input state $|0\rangle$, and classical register c .

FIGURE 4: A single time step of MetaQAOA, where the classical LSTM outputs parameters $\{\vec{\eta}, \vec{\gamma}\}$ dependent on the expectation value $\langle \hat{H} \rangle$ by the QNN.

FIGURE 5: The temporal hybrid computational graph for the metalearning optimization of the LSTM and a QNN is unrolled.

$$|\Psi_\theta\rangle = \prod_{q=1}^P e^{-i\eta_q \hat{H}_M} e^{-i\gamma_q \hat{H}_{C_2}} |\psi_0\rangle, \quad (9)$$

which is our parameterized output, where $\theta = \{\eta, \gamma\}$ is a variational parameter to be optimized by the LSTM and P is the number of times the unitary transformation is executed. In addition, $|\psi_0\rangle$ is an initial state and $\hat{H}_M, \hat{H}_{C_2}$ are the mixer Hamiltonian and Ising Hamiltonian, respectively.

3.3. Training Strategy. In this section, we will provide more detailed information on how the MetaQAOA trains an LSTM to optimize the parameters of QNN. The goal of quantum-classical metalearning is to train LSTM to learn an effective parameter update heuristics for a set of cost functions of interest, i.e., to find an optimizer that efficiently optimizes the special distribution of optimization area on

average. The better optimizer is an optimizer that finds the best approximate local minimum of the cost function in as few function queries as possible. The whole optimal qualification is decided by the current problem category and the application area of interest.

In order to learn this general optimization strategy, the LSTM is trained by using a random instance, which has a fairly general distribution of functions, that is, functions are sampled from Gaussian processes. Since we focus on the known QNN optimization landscapes which are different from the classical Gaussian process optimization landscapes, we aim to train neural optimizers for MaxCut QAOA problems and QNN ansatz. In order to explore the effectiveness of our method, we will train the LSTM for random QNN instances in the target problem category, i.e., MaxCut QAOA, and test the trained network on (larger) previously unseen instances in the target problem category. The

number of variational parameters in these ansatzes does not depend on the size of the system, but only on the number of alternating operators. This means we can use this training strategy to train our LSTM on smaller QNN instance sizes for MaxCut QAOA problems and then test for generalization on larger QNN instance sizes.

4. Experiment Simulation

4.1. Experimental Environment and Dataset. In order to train and test the LSTM optimizer on MaxCut QAOA problems, our experiments are conducted on the CPU and GPU. The TensorFlow Quantum (TFQ) [16], an open-source quantum cloud computing platform for the rapid prototyping of hybrid quantum-classical models for classical or quantum data, is introduced to realize the simulation experiments. Their parameters are shown in Table 1. Besides, the random problem instances are generated in the following form. Firstly, we fix an integer n , and then an integer uniformly from the range $k \in [3, n-1]$ is sampled randomly. Finally, we can generate a random graph $\mathcal{G}_{n,p}$ and the probability of connection between nodes is $p = k/n$, where k is the number of neighbors in each node and n represents the number of nodes. In order to illustrate it more vividly, we generate a random 3-regular graph with 10 nodes and the probability of being neighbors between nodes is $p = 0.3$, as shown in Figure 6. To generate training data, we uniformly sample $n_{\text{train}} \in [6, 12]$, yielding QNN system sizes of at most 12 qubits. To generate our testing data, we also uniformly sample $n_{\text{test}} \in [8, 14]$. Because our experiment is to verify that the trained network can be generalized to larger system and problem instances, the condition, i.e., $n_{\text{train}} < n_{\text{test}}$, has to be met.

In our experiment, 500 problem instances sampled from this training dataset and 250 problem instances as validation dataset are selected to train the LSTM. In addition, we select 100 problem instances sampled from this testing set as testing data. The batch size of each training is 64, and there are a total of 1500 epochs for iterative training. In addition, the Adam optimizer provided by the open-source library is introduced for the network model, and the learning rate is set as 0.001. We choose a short time horizon to minimize the complexity of the training. For reference, 5 iterations is a significantly less number of quantum-classical optimization iterations than what is typically seen in previous works on QNN optimization.

4.2. Circuit Design. We only show the circuits of MaxCut QAOA unitary with $n = 6$ qubits and $n = 8$ qubits at the first time step, i.e., $T = 1$. In our experiment, we have carried out 5 time steps. Expect that the parameters of the different time steps, i.e., γ_t, η_t , are distinct, the unitary circuits with $T = 2, 3, 4, 5$ are the same as those with $T = 1$.

The unitary circuits are shown in Figure 7, where Figure 7(a) shows the circuit of MaxCut QAOA unitary with $T = 1$ and $n = 6$ qubits and Figure 7(b) shows the circuit of MaxCut QAOA unitary with $T = 1$ and $n = 8$ qubits. These circuits are all parameterized quantum circuits, in which

TABLE 1: The experimental environment.

Environments	Parameters
CPU	Intel (R) Xeon (R) Gold 5218
Hard disk	Samsung PM881 512 GB
GPU	NVIDIA GeForce RTX 2080 Ti
Simulation platform	TensorFlow Quantum 2.3.0
Operating system	Windows 10

qubits connected by ZZ gates are neighbours in the generated 3-regular graph and γ_0 and η_0 represent the MaxCut QAOA parameters.

4.3. Results Analysis

4.3.1. Comparison of Training Performance. In order to evaluate the training performance of the algorithm proposed in this paper, some local optimizers, i.e., Nelder-Mead [11], L-BFGS-B [13], and our MetaQAOA, are compared. Nelder-Mead and L-BFGS-B are gradient-free and gradient-based approaches, respectively, which are standard local optimizers. L-BFGS-B is a local optimizer and has access to the gradients. Out of all optimizers chosen, it is the closest to the MetaQAOA in terms of information available to the optimizer and computational burden (i.e., the cost of computing the gradients). Nelder-Mead was chosen as it provides a widely recognized benchmark. Our MetaQAOA requires fewer iterations to reach the threshold of the loss function.

In this paper, we use the observed improvement at each time step as the loss function to measure the quality of the neural network model. This loss function is as follows:

$$L(\boldsymbol{\varphi}) = \mathbb{E}_{f,y} \left[\sum_{t=1}^T \min\{f(\theta_t) - \min_{i < t} [f(\theta_i)], 0\} \right], \quad (10)$$

which is summed over the history of the optimization and the observed improvement at t time step is derived from the difference between the recommended value, $f(\theta_t)$, and the best value obtained over the past history of the optimization until the $t-1$ time step, $\min_{i < t} [f(\theta_i)]$. If there is no improvement at a given time step, then the contribution to the loss is zero. However, temporarily increasing the cost function and then significantly improving the historical best level will be rewarded rather than penalized.

Our experiment aims at the QAOA for MaxCut problem of random 3-regular graphs generated by $n=10$ qubits, yielding QNN system sizes of at 10 qubits. The three methods are compared from the relationship between the training epoch and the value of the loss function as shown in Figure 8. From this figure, we can get that, compared with the other two methods, the metalearner requires fewer iterations to reach the threshold of the loss function, the threshold of loss value after iteration is smaller, and the closest comparable competitor to the metalearner among these methods is L-BFGS-B. This shows that our proposed algorithm has better training performance.

FIGURE 6: A random 3-regular graph with 10 nodes.

FIGURE 7: The circuits of MetaQAOA unitary (a) with $T = 1$ and $n = 6$ qubits and (b) with $T = 1$ and $n = 8$ qubits.

4.3.2. Generalization Performance. Different from the other two methods, our proposed algorithm is generalizable. To illustrate the generalization performance, four kinds of experiments are simulated via TFQ. In this paper, we choose $n_{\text{train}} = 6$ and $n_{\text{test}} = 8$ for the first kind of experiment, $n_{\text{train}} = 8$ and $n_{\text{test}} = 10$ for the second kind of experiment, $n_{\text{train}} = 10$ and $n_{\text{test}} = 12$ for the third kind of experiment, and $n_{\text{train}} = 12$ and $n_{\text{test}} = 14$ for the last kind of experiment.

By inputting 1000 randomly selected parameter combinations $\{\eta, \gamma\}$ into the optimized neural network, the distribution of QNN cost function results can be obtained which

form four circular regions as shown in Figure 9. In addition, the darker the colour of regions, the better the result of QNN's cost function. We make the updated value of the parameter $\gamma^* = (\gamma + 0.5) \times 100$, $\eta^* = (\eta + 0.5) \times 100$. In Figure 9, the horizontal axis is used to represent the value of γ^* , and the vertical axis is used to represent the value of η^* . In order to clearly show the change of parameters, we set epoch = 50. The red dotted line indicates the change trend of the two parameters through 5 time steps, and the red star represents the value of the parameter after the last update. As we can see from Figure 9, the LSTM immediately begins guessing near the basin of attraction and continues to explore

FIGURE 8: The comparison of the three methods in terms of the relationship between the training epoch and the value of the loss function.

FIGURE 9: Continued.

FIGURE 9: The path chosen by the LSTM optimizer on a n_{test} -qubit MaxCut problem after being trained on a set of random n_{train} -qubit MaxCut problems. (a) The path with $n_{\text{train}} = 6$ and $n_{\text{test}} = 8$. (b) The path with $n_{\text{train}} = 8$ and $n_{\text{test}} = 10$. (c) The path with $n_{\text{train}} = 8$ and $n_{\text{test}} = 10$. (d) The path with $n_{\text{train}} = 8$ and $n_{\text{test}} = 10$.

around the region looking to improve the estimate, so we can draw a conclusion that the neural network learns to generalize its heuristic to larger system sizes.

5. Conclusions

According to the shortcomings of weak generalization of the current methods of optimizing MaxCut QAOA circuits, we propose a quantum approximate optimization algorithm with metalearning for MaxCut problem (MetaQAOA) via TFQ platform. Our algorithm trains a classical LSTM network as an outer-loop optimizer to directly optimize the parameters in the QNN constructed by the parameterized quantum circuit. The simulation experiment results via TFQ show that MetaQAOA requires fewer iterations to reach the threshold of the loss function, and the threshold of loss value after training is smaller than the other local optimization methods, i.e., Nelder–Mead and L-BFGS-B. In addition, our method can learn parameter update heuristics that generalize to larger system sizes and still outperform other initialization strategies of this scale. But the metalearning optimizer neural networks in our MetaQAOA cannot scale to arbitrary problems and number of parameters, and research that builds upon it might be addressed in future studies.

Data Availability

The data used to support this study are included within this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The support of all the members of the quantum research group of NUIST is especially acknowledged, and their professional discussions and suggestions have provided us with a lot of help. This work was supported by the Natural Science Foundation of China under Grant nos. 62071240 and 61802002 and in part by the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant no. 19KJB520028, and the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

References

- [1] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *Siam Journal on Computing*, vol. 4, no. 3, pp. 221–225, 1975.
- [2] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, <https://arxiv.org/abs/1411.4028>.
- [3] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, Article ID 015002, 2018.
- [4] A. Peruzzo, J. McClean, P. Shadbolt et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, p. 4213, 2014.
- [5] J. R. McClean, J. Romero, R. Babbush et al., "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, Article ID 023023, 2016.
- [6] A. Kandala, A. Mezzacapo, K. Temme et al., "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.
- [7] G. E. Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," 2014, <https://arxiv.org/abs/1811.08419>.

- [8] T. Tamayo-Mendoza, C. Kreisbeck, R. Lindh et al., “Automatic differentiation in quantum chemistry with an application to fully variational Hartree-Fock,” *ACS Central Science*, vol. 4, no. 5, pp. 559–566, 2017.
- [9] G. G. Guerreschi and M. Smelyanskiy, “Practical optimization for hybrid quantum-classical algorithms,” 2017, <https://arxiv.org/abs/1701.01450>.
- [10] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.
- [11] Z. Wang, S. Hadfield, Z. Jiang et al., “Quantum approximate optimization algorithm for MaxCut: a fermionic view,” *Physical Review A*, vol. 97, Article ID 022304, 2018.
- [12] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.
- [13] H. J. Kushner, “A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise,” *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.
- [14] M. V. Altaisky, “Quantum neural network,” *International Journal of Theoretical Physics*, vol. 36, no. 12, pp. 2855–2875, 2001.
- [15] M. Broughton, G. Verdon, T. McCourt et al., “TensorFlow quantum: a software framework for quantum machine learning,” 2020, <https://arxiv.org/abs/2003.02989>.
- [16] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: continual prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [17] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with LSTM recurrent networks,” *Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.
- [18] Y. Bengio, S. Bengio, and J. Cloutier, *Learning a Synaptic Learning Rule*, Université de Montréal, Département D’Informatique Et De Recherche Opérationnelle, Montreal, Canada, 1990.
- [19] K. Li and J. Malik, “Learning to optimize,” 2016, <https://arxiv.org/abs/1606.01885>.
- [20] M. Andrychowicz, M. Denil, S. Gomez et al., “Learning to learn by gradient descent by gradient descent,” in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3981–3989, Barcelona, Spain, December 2016.
- [21] Y. Bengio, P. Simard, P. Frasconi et al., “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [22] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.
- [23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [24] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico, May 2016.
- [25] D. Wecker, M. B. Hastings, and M. Troyer, “Training a quantum optimizer,” *Physical Review A*, vol. 94, no. 2, Article ID 022309, 2016.
- [26] G. Verdon, M. Broughton, and J. Biamonte, “A quantum algorithm to train neural networks using low-depth circuits,” 2017, <https://arxiv.org/abs/1712.05304>.