






Received 21 December 2022; revised 29 March 2023; accepted 29 March 2023; date of publication 12 April 2023; date of current version 21 June 2023.

Digital Object Identifier 10.1109/TQE.2023.3265709

# Automated Quantum Circuit Design With Nested Monte Carlo Tree Search

PEIYONG WANG<sup>1</sup> , MUHAMMAD USMAN<sup>2,3</sup> ,  
UDAYA PARAMPALLI<sup>1</sup>  (Senior Member, IEEE),  
LLOYD C. L. HOLLENBERG<sup>2</sup> , AND CASEY R. MYERS<sup>1,4</sup> 

<sup>1</sup>School of Computing and Information Systems, Faculty of Engineering and Information Technology, The University of Melbourne, Melbourne, VIC 3010, Australia

<sup>2</sup>School of Physics, The University of Melbourne, Parkville, VIC 3010, Australia

<sup>3</sup>Data61, CSIRO, Clayton, VIC 3168, Australia

<sup>4</sup>Silicon Quantum Computing Pty, Ltd., Kensington, NSW 2052, Australia

Corresponding author: Peiyong Wang (e-mail: peiyongw@student.unimelb.edu.au).

This work was supported by the Defence Science Institute, an initiative of the State Government of Victoria.

**ABSTRACT** Quantum algorithms based on variational approaches are one of the most promising methods to construct quantum solutions and have found a myriad of applications in the last few years. Despite the adaptability and simplicity, their scalability and the selection of suitable ansatzes remain key challenges. In this work, we report an algorithmic framework based on nested Monte Carlo tree search coupled with the combinatorial multiarmed bandit model for the automated design of quantum circuits. Through numerical experiments, we demonstrate our algorithm applied to various kinds of problems, including the ground energy problem in quantum chemistry, quantum optimization on a graph, solving systems of linear equations, and finding encoding circuits for quantum error detection codes. Compared to the existing approaches, the results indicate that our circuit design algorithm can explore larger search spaces and optimize quantum circuits for larger systems, showing both versatility and scalability.

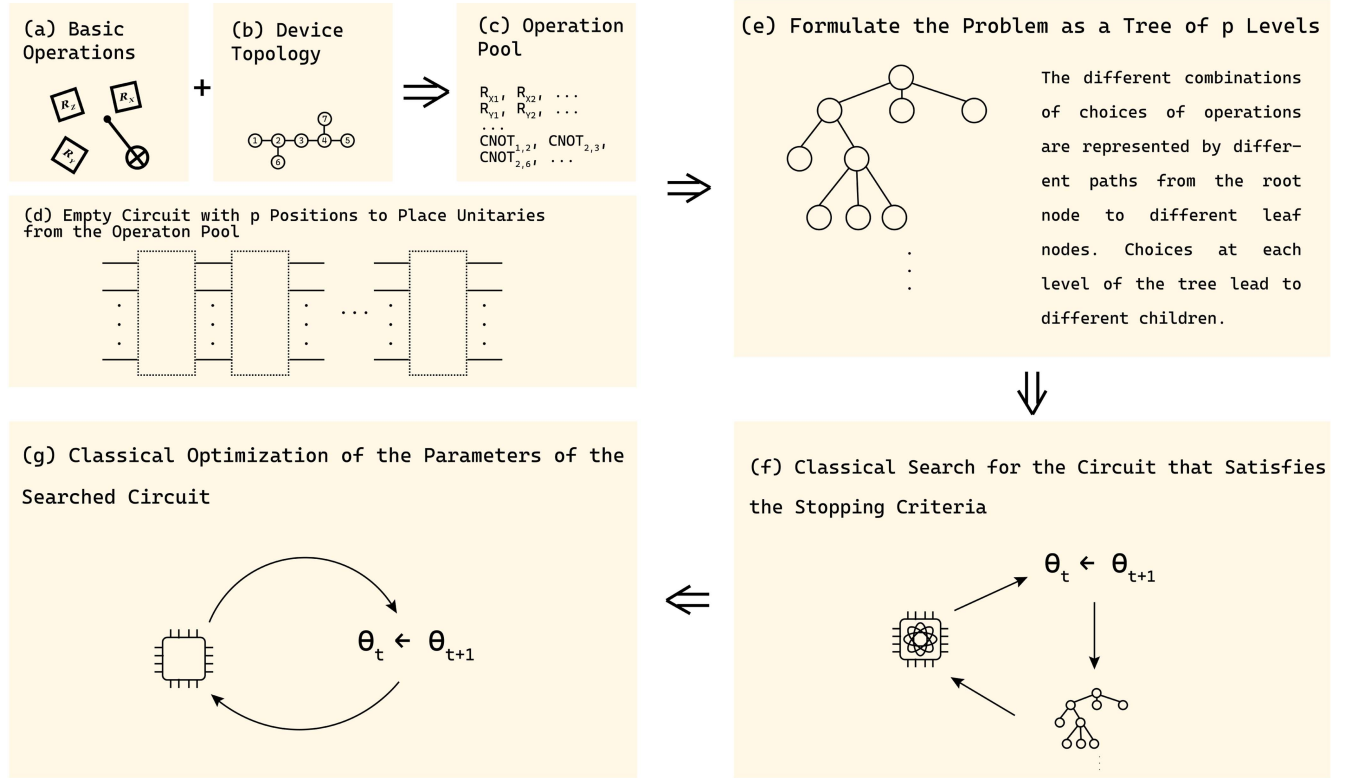
**INDEX TERMS** Artificial intelligence (AI), Monte Carlo tree search (MCTS), quantum machine learning, quantum neural networks, variational quantum circuits (VQCs).

## I. INTRODUCTION

The variational quantum Circuit (VQC), also known as a parameterized quantum circuit (PQC) approach, first proposed for solving the ground state energy of molecules [1], has been extended to many open research problems, including in the field of quantum machine learning [2], quantum chemistry [3], option pricing [4], and quantum error correction [5], [6]. The performance of VQC methods largely depends on the choice of a suitable ansatz, which is not an easy task because, generally, the search space is very large, and it is not well established whether there is a common principle for designing such an ansatz. For problems involving physical systems, such as in quantum chemistry, we can rely on the well-defined properties of molecular systems for ansatz designing, such as the hardware efficient ansatz [7] and physical-inspired ansatz, such as  $k$ -UpCCGSD [8]. However, this cannot be generalized to other areas, such as designing variational error correction circuits or quantum optimization problems. For example, in [6], when developing a variational

circuit that can encode logical states for the five-qubit quantum error correction code, the authors adopted an expensive approach by randomly searching over a large number (on the order of 10 000) of circuits. It is anticipated that with the increasing number of application areas for VQCs and the need for scalability to tackle large problem sizes without relying on fundamental physical properties, such random search methods or methods based purely on human heuristics will struggle to find suitable ansatzes. Therefore, developing efficient methods for the design of VQCs is important. Here, we focus on developing algorithms for the automated design of VQCs by leveraging the power of artificial intelligence (AI), which can be deployed for a wide range of applications.

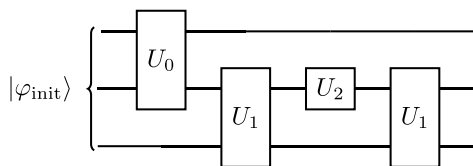
Although modern AI research often focuses on applications of image and natural language processing, the power of AI can also bring new knowledge in many areas, especially scientific discovery. AlphaFold2 managed to discover new mechanisms for the bonding region of the proteins and inhibitors [9] with competitive accuracy in predicting the



**FIGURE 1.** Overview of the algorithmic framework proposed in this article for automated quantum circuit design. The operation pool (c) is obtained by tailoring the basic operations (a) with respect to the device topology (b). After that, we formulate the combinations of different choices of operations at different layer positions in the circuit (d) as a search tree (e). In (f), we evaluate our circuit on a quantum processor or quantum simulator to get the value of the loss or reward function, and according to the value of the loss/reward function, we update the parameters on a classical computer, then, use MCTS to search for the current best circuit. We then send the updated circuit structure together with the updated parameters to the quantum processor/simulator to obtain a new set of loss/reward values. The process depicted in (f) will repeat until a circuit that meets the stopping criteria is found. Then, as shown in (g), we will follow the usual process to optimize the parameters in the searched VQC by classical-quantum hybrid computing.

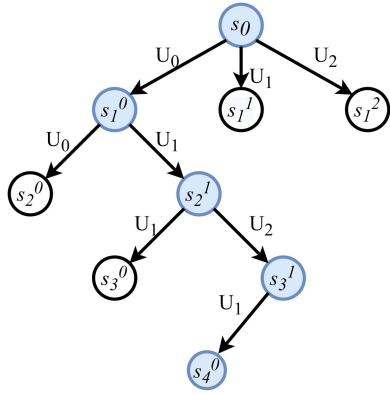
**TABLE 1.** Run-Time-Related Search Hyperparameters and Maximum Circuit Evaluation Numbers.

Experiment	# of layers ( $p$ )	# of parameters per layer ( $l$ )	warm-up batch size ( $b_w$ )	search batch size ( $b_s$ )	search sample round ( $r_s$ )	exploit execute round ( $r_e$ )	# warm-up iterations ( $t_w$ )	# search iterations ( $t_s$ )	max. # circuit eval.
[[4,2,2]]	6	3	1000	200	16	160	0	50	568000
H <sub>2</sub>	30	3	500	100	10	20	5	45	1337500
LiH	20	3	500	25	10	20	2	48	297000
H <sub>2</sub> O	50	3	500	50	10	20	2	48	1095000
VQLS	10	3	50	40	50	100	5	45	263250
QAOA (un-weighted)	15	3	100	100	100	500	2	48	1305200
QAOA (weighted)	10	3	100	50	10	100	10	90	476000

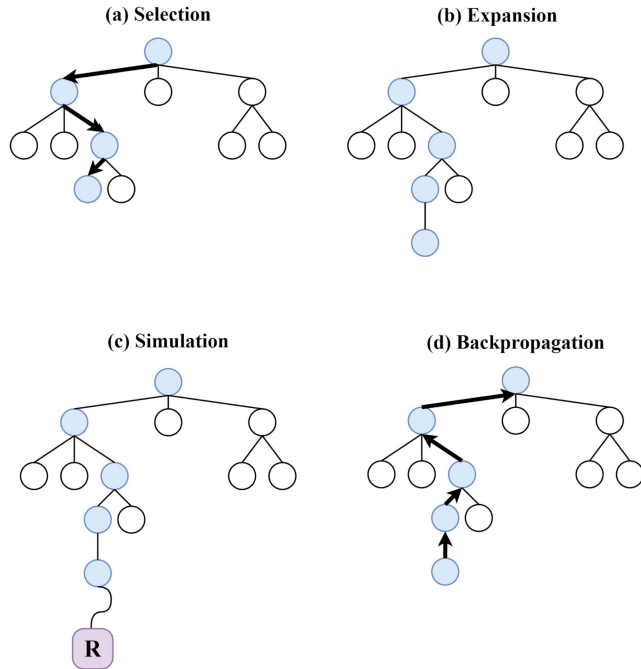


**FIGURE 2.** Example of the circuit corresponding to the series of unitaries applied to  $|\varphi_{\text{init}}\rangle$  in (3). This circuit can also be represented as a path shown in the tree in Fig. 3.

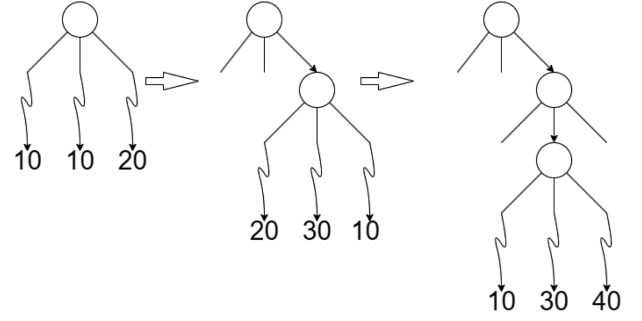
3-D structure of proteins in the 14th Critical Assessment of protein Structure Prediction (CASP) competition. In 2021, machine learning algorithms helped mathematicians discover new mathematical relationships in two different areas of mathematics [10]. Like VQCs, modern deep neural networks (DNN) also face a design problem when composing the network for certain tasks. With the help of AI algorithms, researchers developed techniques to search suitable network architectures in a large search space efficiently. Famous



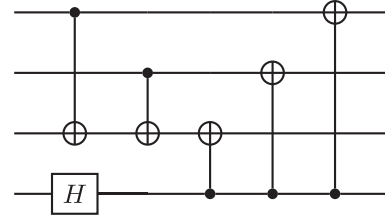
**FIGURE 3.** Tree representation (along the arc with blue-shaded circles) of the unitary described in (3) and (4) as well as Fig. 2. The circle with  $s_0$  is the root of the tree, which represents an empty circuit. Other circles with  $s_i^j$  denote the  $j$ th node at the  $i$ th level of the tree.  $i$  can also indicate the number of layers currently in the circuit at state  $s_i^j$ . For example, on the leftmost branch of the tree, there is a node labeled  $s_2^0$ , indicating that it is the 0th node at level 2. At  $s_2^0$ , the circuit would be  $\mathcal{P}_{s_2^0} = [U_0, U_0]$ , which clearly only has two layers. We can also see that some of the possible branches along the blue-node path are pruned, leading to the size of the operation pool at some nodes being smaller than the total number of possible choices  $c = |C|$ .



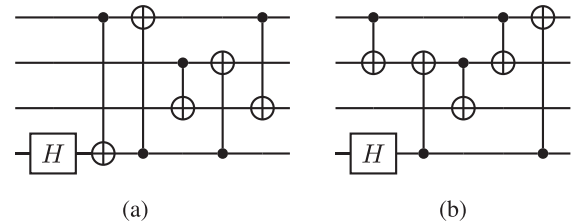
**FIGURE 4.** Four stages of MCTS. (a) Selection: Go down from the root node to a nonfully expanded leaf node. (b) Expansion: Expand the selected node by taking an action. After selecting an action, we will either move to an already discovered child node or, if the action is previously unselected, we will expand a new child node to our current node. (c) Simulation: Simulate the game, which in our case is the quantum circuit, to obtain reward information  $R$ . During simulation, there will be a different policy, called the *rollout* policy, for selecting actions until we reach the termination point of the game or exceed the time limit. We will adopt the nested MCTS method to determine the reward of an action or node, see Fig. 5. (d) Backpropagation: Backpropagation of the reward information along the path (arc) taken.



**FIGURE 5.** Nested MCTS. (Left) The root node has three possible actions, which in this case, are unselected initially. We perform MCTS on all three children nodes (generated by the three possible actions) to update their reward information. After one iteration of MCTS with each child as the root node for the search tree that MCTS performed, the rewards of these three actions leading to the three child nodes are 10, 10, and 20, respectively. In this case, the child node on the right has the highest reward. (Middle) After selecting the right-side child node, we perform the same MCTS on all three possible children nodes as before, which gives updated reward information. In this case, the middle child node has the highest reward, meaning that at this level we expand the middle child node. (Right) Similar operations as before. If we only perform nested MCTS at the root node level, then it will be a level-1 nested MCTS.

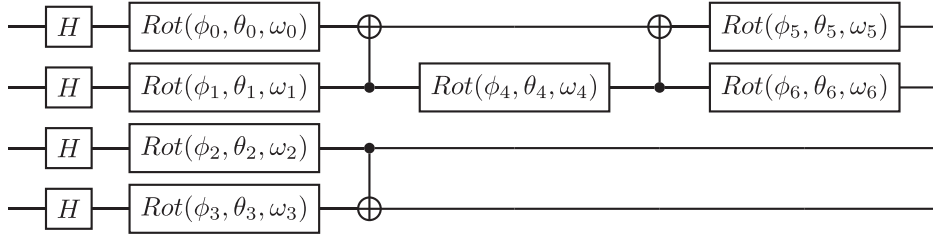


**FIGURE 6.** Encoding circuit of the  $[[4,2,2]]$  code [33] to detect  $X$ - and  $Z$ -errors. It needs 4 physical qubits for 2 logical qubits and has a code distance 2. By our settings, the number of layers equals the number of operations in the circuit. In this figure, the number of layers is 6.

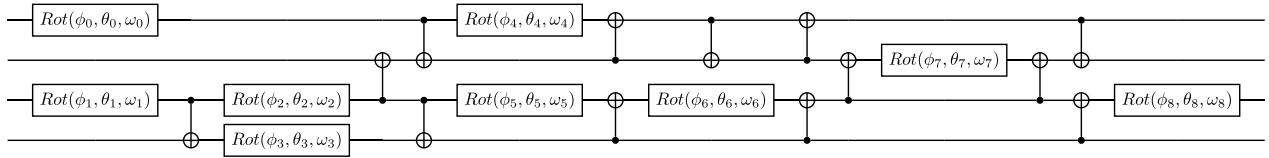


**FIGURE 7.** Two different encoding circuits of the  $[[4,2,2]]$  code produced by the search algorithm. With simple symbolic or numerical calculation, we can easily verify that both of these circuits produce the required codewords for the  $[[4,2,2]]$  error detection code, which indicates our algorithm can produce device-tailored circuits when more restrictions are added.

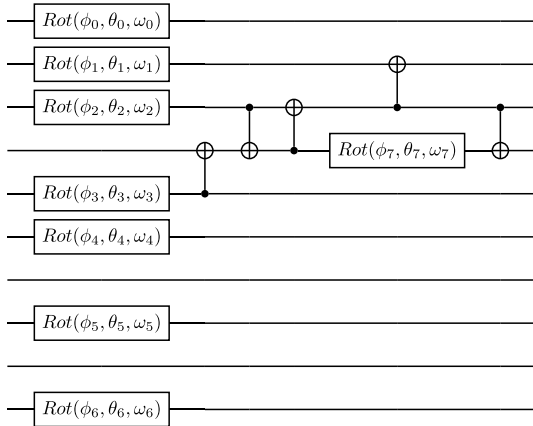
algorithms for neural architecture search (NAS) include the differentiable architecture search (DARTS) algorithm [11], which models the choice of operations placed in different layers as an independent categorical probabilistic model that can be optimized via gradient descent methods, and the PNAS algorithm [12], which models the search process with the sequential model-based optimization strategy. Tree-based algorithms were also proposed for NAS, such as AlphaX [13], which models the search process similarly to the search stage of AlphaGo [14], and an algorithm proposed



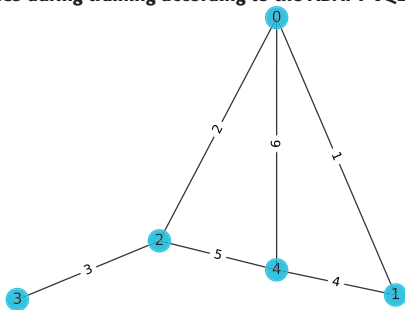
**FIGURE 8.** Circuit searched for the VQLS problem.  $\text{Rot}(\phi, \theta, \omega) = R_Z(\omega)R_Y(\theta)R_Z(\phi)$ . The four Hadamard gates at the beginning of the circuit give an equal superposition and are not included when constructing the search tree, i.e., the composed circuits will always start with four Hadamard gates placed on the four qubits. When drawing the circuit, the Placeholder gates, which are just identity gates, are removed from searched  $\mathcal{P}$ , although they were considered when constructing the search tree.



**FIGURE 9.** Circuit for finding the ground energy of the  $\text{H}_2$  molecule produced by the search algorithm. We can see that there are already familiar structures emerging, such as the SWAP gate between the first two qubits and the Ising coupling gatelike structure underneath the decomposed SWAP gate.



**FIGURE 10.** Circuit structure produced by the search algorithm for LiH. We can see that the structure of the circuit is quite simple, compared to the circuit for  $\text{H}_2$  in Fig. 9, indicating that the initial state  $|\psi_0\rangle = |0\rangle^{\otimes 10}$  is already very close to the ground energy state. We also noticed that the circuit produced by the search algorithm is very simple compared to the ansatz composed with `qml.SingleExcitation` and `qml.DoubleExcitation` from PennyLane [47] according to the physical properties of the molecule, which can be decomposed into 180 gates, including 96 two-qubit control gates after removing gates with small gradient values during training according to the ADAPT-VQE method [48].

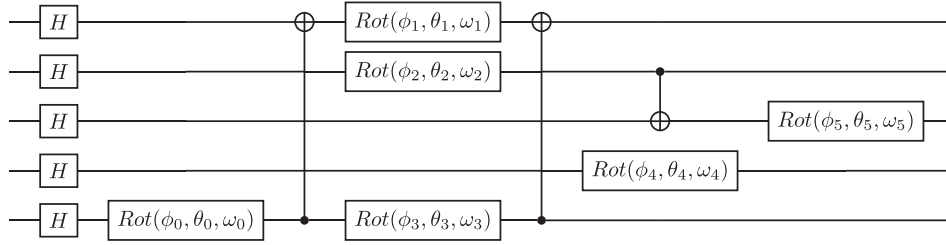


**FIGURE 11.** Problem graph for the weighted MaxCut experiment. The weights on edges (0,2), (0,4), (0,1), (2,4), (4,1), and (2,3) are 2, 6, 1, 5, 4, and 3, respectively.

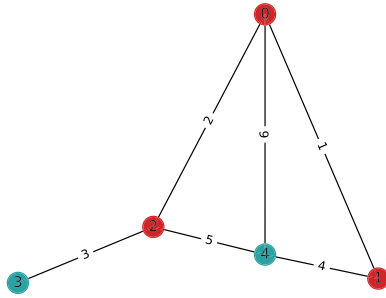
in [15], in which the authors combined tree search and combinatorial multiarmed bandits (CMABs) to achieve better performance.

Based on progress in NAS algorithms, efforts have been made to develop similar approaches for Quantum Ansatz (Architecture) Search (QAS) problems. Zhang et al. [16] adapted the DARTS algorithm [11] from NAS for QAS, which models the distribution of different operations within a single layer with the independent category probabilistic model. The search algorithm updates the parameters in the VQC as well as the probabilistic model. However, it has been shown in NAS literature that DARTS tends to assign fast-converge architectures with high probability during sampling [17], [18]. Also, the off-the-shelf probabilistic distributions for modeling the architecture space tend to have difficulties when the search space is large. Later, the same group of authors developed a neural network to evaluate the performance of PQCs without actually training the circuits and incorporated this neural network into quantum architecture search [19]. While NAS algorithms often focus on image-related tasks and it has been proven through many experiments that one neural network architecture can act as a backbone feature extractor for many downstream tasks, the structures of VQCs for different problems often vary a great deal with different problems, casting some doubts on the generalization abilities of such neural-predictor-based QAS algorithms. Kuo et al. [20] proposed a deep reinforcement learning-based method for tackling QAS. The reinforcement learning agent is optimized by the advantage actor-critic and proximal policy optimization algorithms.

However, NAS algorithms based on policy gradient reinforcement learning have been shown to easily get stuck in local minimal, producing less optimal solutions [21], [22]. Also, the data size for training a reinforcement learning agent



**FIGURE 12.** Searched circuit for the five-node MaxCut problem. We can see that we have a rather simple circuit, which gives us the optimal solution 00011, shown in Fig. 13. Similar to Fig. 23 in the Appendix, the circuit produced by our search algorithm requires a smaller number of CNOT gates than even a single layer of standard QAOA circuit for the problem shown in Fig. 11.



**FIGURE 13.** Sampled solution for the five-node MaxCut problem, with nodes 0, 1, and 2 colored red and nodes 3 and 4 colored blue for the two partitions of the graph.

will overwhelm the system when the number of actions the agent can choose from is large. He et al. [23] applied meta-learning techniques to learn good heuristics of both the architecture and the parameters. Du et al. [24] proposed a QAS algorithm based on the one-shot NAS, where all possible quantum circuits are represented by a supernet with a weight-sharing strategy and the circuits are sampled uniformly during the training stage. After finishing the training stage, all circuits in the supernet are ranked and the best-performed circuit will be chosen for further optimization. Later Linghu et al. [25] applied similar techniques in search of a classification circuit on a physical quantum processor. Meng et al. [26] applied Monte Carlo tree search (MCTS) to ansatz optimization for problems in quantum chemistry and condensed matter physics. These studies often restrict themselves to one or two types of problems and small-sized systems. Then, how to develop a framework for larger and different kinds of problems? In this article, we aim to address this question.

In order to develop a search technique that can be applied to larger search spaces and different variational quantum problems, we introduce an algorithm for QAS problems based on the CMAB model as well as MCTS. With a view of exploring extremely large search spaces compared to previous work in the literature, the focus of our strategy is underpinned by a reward scheme, which dictates the choices of the quantum operations at each step of the algorithm with the naive assumption [27]. This enables our strategy to work on larger systems, more than 7 qubits, whereas the existing examples [16], [19], [20], [23], [24] are restricted to typically

3 or 4 qubits, with the largest being 6 qubits. To demonstrate our method, we show its application to a variety of problems, including encoding the logic states for the  $[[4,2,2]]$  quantum error detection code, solving the ground energy problem for different molecules as well as linear systems of equations, and searching the ansatz for solving optimizations problems. Our work confirms that the automated quantum architecture search based on the MCTS+CMAB approach exhibits great versatility and scalability and, therefore, could provide an efficient solution and new insights into the problems of designing VQCs.

This article is organized as follows: Section II introduces the basic notion of MCTS, as well as other techniques required for our algorithm, including nested MCTS and naive assumptions from the CMAB model. Section III reports the results based on the application of our search algorithm to various problems, including searching for encoding circuits for the  $[[4,2,2]]$  quantum error detection code, the ansatz circuit for finding the ground state energy of different molecules, as well as circuits for solving a linear system of equations and optimization. In Section IV, we discuss the results and present conclusions.

## II. METHODS

### A. PROBLEM FORMULATION

In this article, we formulate the quantum ansatz search problem, which is aimed to automatically design VQCs to perform various tasks as a tree structure. We slice a quantum circuit into layers, and for each layer, there is a pool of candidate operations. Starting with an empty circuit, we fill the layers with operations chosen by the search algorithm, from the first to the final layer. The overview of this process is shown in Fig. 1, where we show the overarching algorithmic framework developed in this work.

A quantum circuit is represented as an (ordered) list  $\mathcal{P}$  of operations of length  $p$  chosen from the operation list. The length of this list is fixed within the problem. The operation pool is a set

$$\mathcal{C} = \{U_0, U_1, \dots, U_{c-1}\} \quad (1)$$

with  $|\mathcal{C}| = c$  the number of elements. Each element  $U_i$  is a possible choice for a certain layer of the quantum circuit. Such operations can be parameterized (e.g., the  $R_Z(\theta)$  gate)



or nonparameterized (e.g., the Pauli gates). A quantum circuit with four layers could, for instance, be represented as

$$\mathcal{P} = [U_0, U_1, U_2, U_1] \quad (2)$$

where, according to the search algorithm, the operations chosen for the first, second, third, and fourth layers are  $U_0$ ,  $U_1$ ,  $U_2$ , and  $U_1$ . In this case,  $p = 4$  and the size of the operation pool  $|\mathcal{C}| = c$  (although only three different operations appeared in the circuit, this does not mean we only have three different operations in the operation pool). The search tree for this circuit (see Fig. 2) is shown in Fig. 3. In this article, we will only deal with unitary operations or unitary channels. The output state of such a quantum circuit can then be written as

$$|\varphi_{\text{out}}\rangle = U_1 U_2 U_1 U_0 |\varphi_{\text{init}}\rangle \quad (3)$$

where  $|\varphi_{\text{init}}\rangle$  is the initial state of the quantum circuit. For simplicity, we will use integers to denote the chosen operations (such operations can be whole-layer unitaries, such as the mixing Hamiltonians often seen in typical QAOA circuits or just single- and two-qubit gates).

For example, the quantum circuit from (3) can be written as

$$\mathcal{P} = [0, 1, 2, 1] \quad (4)$$

and the operation at the  $i$ th layer can be referred as  $k_i$ . For example, in the quantum circuit shown in Fig. 2, we have  $k_2 = 1$ .

The performance of the quantum circuit can be evaluated from the loss  $\mathcal{L}$  or reward  $\mathcal{R}$ , where the reward is just the negative of the loss. Both are functions of  $\mathcal{P}$ , and the parameters of the chosen operations  $\theta$

$$\mathcal{L}(\mathcal{P}, \theta) = L(\mathcal{P}, \theta) + \lambda \quad (5)$$

$$\mathcal{R}(\mathcal{P}, \theta) = R(\mathcal{P}, \theta) - \lambda \quad (6)$$

where  $\lambda$  is some (positive-definite) penalty function that may only appear when certain circuit structures appear, as well as other kinds of penalty terms, such as a penalty on the sum of the absolute value of weights or the number of a certain type of gates in the circuit;  $L$  and  $R$  are the loss/reward before applying the penalty. The penalty term  $\lambda$  aims to “sway” the search algorithm from structures we do not desire. Instead of storing all the operation parameters for each different quantum circuit, we share the parameters for a single operation at a certain location. That is, we have a multidimensional array of shape  $(p, c, l)$ , where  $l$  is the maximum number of parameters for the operations in the operation pool. If all the operations in the pool are just the single qubit rotation gate Rot [28]:

$$\begin{aligned} \text{Rot}(\phi, \theta, \omega) &= R_Z(\omega) R_Y(\theta) R_Z(\phi) \\ &= \begin{bmatrix} e^{-i(\phi+\omega)/2} \cos(\theta/2) & -e^{i(\phi-\omega)/2} \sin(\theta/2) \\ e^{-i(\phi-\omega)/2} \sin(\theta/2) & e^{i(\phi+\omega)/2} \cos(\theta/2) \end{bmatrix} \end{aligned} \quad (7)$$

as well as its controlled version CRot gate on different (pairs of) qubits; then, in this case,  $l = 3$ .

To reduce the space required to store the parameters of all possible quantum circuits, for a quantum circuit with operation  $k$  at layer  $i$ , the parameter is the same at that layer for that specific operation for all other circuits with the same operation at the same location, which means we are sharing the parameters of the unitaries in the operation pool with other circuits. For example, in Fig. 3, besides the blue-node arc  $\mathcal{P} = [U_0, U_1, U_2, U_1]$ , there are also other paths, such as  $\mathcal{P}' = [U_0, U_1, U_1, \dots]$ , and since the first two operations in  $\mathcal{P}$  and  $\mathcal{P}'$  are the same, then we will share the parameters of  $U_0$  and  $U_1$  between these two circuits by setting the parameters to be the same for  $U_0$  and  $U_1$  in both circuits. Such a strategy is often called “parameter-sharing” or “weight-sharing” in the NAS literature.

As shown in Fig. 3 and mentioned earlier, the process of composing or searching a circuit can be formulated in the form of a tree structure. For example, if we start from an empty list  $P = []$  with maximal length four and an operation pool with three elements  $C = \{U_0, U_1, U_2\}$ , then the state of the root node of our search tree will be the empty list  $s_0^0 = []$ . The root node will have three possible actions (if there are no restrictions on what kind of operations can be chosen), which will lead us to three children nodes with states  $s_1^0 = [U_0] = [0]$ ,  $s_1^1 = [U_1] = [1]$ ,  $s_1^2 = [U_2] = [2]$ . For each of these nodes, there will be a certain number of different operations that can be chosen to append to the end of the list, depending on the specific restrictions. There will always be a “placeholder” operation that can be chosen if all other operations fail to meet the restrictions. The penalty resulting from the number of “placeholder” operations will only be reflected in the loss (or reward) of the circuit. The nodes can always be expanded with different actions, leading to different children, until the maximum length of the quantum circuit has been reached, which will give us the leaf node of the search tree.

Following the (generalized) definition of the CMAB approach proposed in [27], the process of choosing operations at each layer can be viewed as both a *local* and *global* multiarmed bandit (MAB). A MAB, just as its name indicates, is similar to a slot machine (in the casino) but has multiple levers, or arms, that can be pulled. Or equivalently, it can be viewed as someone who has multiple arms that can pull the levers on different slot machines. In both cases, the rewards obtained from pulling different arms follow different (often unknown) distributions. The person pulling these arms needs to develop a strategy that can maximize his rewards from the machine(s). If we consider the whole circuit search problem as a MAB (the global MAB,  $\text{MAB}_g$ ), then the “arms” are different circuit configurations. Although the rewards of these circuits are relatively easy to obtain based on the value of their cost functions after training is finished (which still requires a fair amount of time for training), the exploding number of possible circuit configurations when the size of operation pool and the number of layers increase makes it impossible to perform an informed search for suitable solutions while training every circuit encountered during the search

process. Since our circuit is basically a combination of different choices of layer unitaries, we can decompose the whole problem into the choices of unitaries at each layer, which is the local MAB,  $MAB_i$ ,  $i$  denoting the MAB problem of choosing the suitable unitary at layer  $i$ . In the local MAB for a single layer, the “arms” of the MAB are no longer the circuit configuration but instead the (permitted) unitary operations from the operation pool  $\mathcal{C}$ . Although the number of choices for the local MABs is considerably smaller than the global MAB, the reward for each arm is not directly observable. In the next section, we will introduce the naive assumption [27] to approximate the rewards of the local MABs from the global MAB, which will help us determine the rewards of the actions on each node (state) on the search tree for MCTS.

## B. MCTS, NESTED MCTS, AND THE NAIVE ASSUMPTION

MCTS is a heuristic search algorithm for a sequence decision process. It has achieved great success in other areas, including defeating the 18-time world champion Lee Sedol in the game of Go [14], [29]. Generally, there are the following four stages in a single iteration of MCTS (see Fig. 4) [30].

- 1) *Selection* [see Fig. 4(a)]. In the selection stage, the algorithm will start from the root of the tree, find a node at the end of an arc (a path from the root of the tree to the leaf node, the path marked by bold arrows and blue circles in Fig. 4). The nodes along the arc are selected according to some policy, often referred to as the “selection policy,” until a nonfully expanded node or a leaf node is reached. If the node is a leaf node, i.e., after selecting the operation for the last layer of the quantum circuit, we can directly jump to the simulation stage to get the reward of the corresponding arc. If the node is not a leaf node, i.e., the node is not fully expanded, and then, we can progress to the next stage.
- 2) *Expansion* [see Fig. 4(b)]. In the expansion stage, we choose a previously unvisited child at the node selected in the previous stage by choosing a previously unperformed action. We can see from the upper right tree in Fig. 4 that a new node has been expanded at the end of the arc.
- 3) *Simulation* [see Fig. 4(c)]. In the simulation stage, if the node obtained from the previous stages is not a leaf node, we continue down the tree, select actions at each node along the arc according to the rollout policy until we have reached a leaf node, i.e., finish choosing the operation for the last layer. After we have the leaf node, we simulate the circuit and obtain the reward  $\mathcal{R}$  based on the loss  $\mathcal{L}$ . Usually, the loss  $\mathcal{L}$  is required to update the parameters in the circuit.
- 4) *Backpropagation* [see Fig. 4(d)]. In this stage, the reward information obtained from the simulation stage is back-propagated through the arc back from the leaf node up to the root of the tree, and the number of visits and the (average) reward for each node along the arc are updated.

The nested MCTS algorithm [31] is based on the vanilla MCTS algorithm. However, before selecting the best child according to the selection policy, a nested MCTS will be performed on the subtrees with each child as the root node. Then, the best child will be selected according to the selection policy with updated reward information; see Fig. 5.

We denote a quantum circuit with  $p$  layers  $\mathcal{P} = [k_1, \dots, k_p]$ , with each layer  $k_i$  having a search space no greater than  $|\mathcal{C}| = c$  (where  $c$  is the number of possible unitary operations, as defined earlier). Each choice for layer  $k_i$  is then a *local arm* for the *local MAB*,  $MAB_i$ . The set of these choices is also denoted as  $k_i$ . The combination of all  $p$  layers in  $\mathcal{P}$  forms a valid quantum circuit, which is called a *global arm* of the *global MAB*,  $MAB_g$ .

Since the global arm can be formed from the combination of the local arms, if we use the naive assumption [27], the global reward  $R_{\text{global}}$  for  $MAB_g$  can be approximated by the sum of the reward of local MABs, and each local reward only depends on the choice made in each local MAB. This also means that if the global reward is more easily accessed than the local rewards, the local rewards can be approximated from the global reward. To simplify the algorithm itself and reduce complexity, we will adopt the naive assumption for the distribution of local and global rewards, where we have a linear relationship between the global reward and local rewards

$$R_{\text{global}} = \frac{1}{p} \sum_{i=1}^p R_i. \quad (8)$$

When searching for quantum circuits, we have no access to the reward distribution of individual unitary operations; however, we can apply the naive assumption to approximate those rewards (“local reward”) with the global reward

$$R_i \approx R_{\text{global}} \quad (9)$$

where  $R_i$  is the reward for pulling an arm at *local MAB* <sub>$i$</sub>  and  $R_{\text{global}}$  is the reward for the global arm. With the naive assumption, we will not need to directly optimize the large space of global arms as in traditional MABs. Instead, we can apply MCTS on the local MABs to find the best combination of local arms. Also, such a linearized (and simple) assumption enables our algorithm to be suitable for a wide range of problems.

In the original work on nested MCTS [31], a random policy was adopted for sampling. In this article, we will instead adopt the upper confidence bound (UCB) policy [32], to balance exploration and exploitation. It should be noted that UCB is not the only policy that considers balancing exploration and exploitation. Other policies, such as  $\epsilon$ -Greedy, also try to balance these two different kinds of actions but with more randomness. Unlike  $\epsilon$ -Greedy, UCB will take the number of times each arm is pulled into account when trying to choose between a high-reward arm or a previously less pulled arm. Given a local  $MAB_i$ , with the set of all the

possible choices  $k_i$ , the UCB policy can be defined as

$$\text{UCB} : \underset{\text{arm}_j \in k_i}{\text{argmax}} \bar{R}(k_i, \text{arm}_j) + \alpha \sqrt{\frac{2 \ln n_i}{n_j}} \quad (10)$$

where  $\bar{R}(k_i, \text{arm}_j)$  is the average reward for  $\text{arm}_j$  (i.e., the reward for operation choice  $U_j$  for layer  $k_i$ ) in local  $\text{MAB}_i$ ,  $n_i$  is the number of times that  $\text{MAB}_i$  has been used, and  $n_j$  is the number of times  $\text{arm}_j$  has been pulled. The parameter  $\alpha$  provides a balance between exploration ( $\sqrt{\frac{2 \ln n_i}{n_j}}$ ) and exploitation ( $\bar{R}(k_i, \text{arm}_j)$ ). The UCB policy modifies the reward on which the selection of action will be based.

For small  $\alpha$ , the actual reward from the bandit will play an important role in the UCB-modified rewards, which will lead to selecting actions with previously observed high rewards. When  $\alpha$  is large enough, the second term, which will be relatively large if  $\text{MAB}_i$  has been visited many times but  $\text{arm}_j$  of  $\text{MAB}_i$  has only been pulled a small number of times, will have more impact on the modified reward, leading to a selection favoring previously less visited actions.

### C. QAS WITH NESTED NAIVE MCTS

Generally, a single iteration for the search algorithm will include two steps for nonparameterized circuits and two more parameter-related steps for PQCs. The set of parameters, which will be referred to as the parameters of the super circuit, or just parameters in the following algorithms, follow the same parameter-sharing strategy as described in Section II-A. That is, if the same unitary operation (say,  $U_2$ ) appears in the same location (say, layer #5) across different quantum circuits, then the parameters are the same, even for different circuits. Also, with PQCs, it is common practice to “warm up” the parameters by randomly sampling a batch of quantum circuits, calculating the averaged gradient, and updating the parameters according to the averaged gradient to get a better start for the parameters during the search process. During one iteration of the search algorithm (for the algorithm pseudocode, see Section A in the Appendix), we will generally need to sample a batch of circuits according to a sample policy and perform exploitation to find the best arc (circuit). Both of these two processes will need a function to select a child node, which selects actions based on the selection policy, as well as pruning nodes with low average rewards, and a function to execute a single round. The *ExecuteSingleRound* function selects a path leading from the current node to a leaf node of the tree following the selection policy, as well as performing the simulation stage to obtain and backpropagate rewards. The difference between sampling an arc and exploiting an arc, although they both require multiple executions of the node selection function, is that the *SampleArc* function will first call the *ExecuteSingleRound* function at the root node  $N$  times to update the rewards in the tree following a sampling policy, then select an arc based on the updated rewards while the *ExploitArc* function will execute the *ExecuteSingleRound* function at each node along

the arc from the root  $N$  times. The process for each iteration is listed as follows.

- 1) Sample a batch of quantum circuits from the super circuit with the *SampleArc* function described in Algorithm 1.
- 2) (For PQCs) Calculate the averaged gradients of the sampled batch and add noise to the gradient to guide the optimizer to a more “flat” minimum if needed.
- 3) (For PQCs) Update the super circuit parameters according to the averaged gradients.
- 4) Find the best circuit with the *ExploitArc* function described in Algorithm 2.

We could also set up early-stopping criteria for the search. That is, when the reward of the circuit obtained with Algorithm 2 meets a preset standard, we stop the search algorithm and return the circuit that met such standard (and further fine-tune the circuit parameters if there are any).

With the naive assumption, which means the reward is evenly distributed on the local arms pulled for a global MAB, we can impose a prune ratio during the search. That is, given a node that has child nodes, if the average reward of a child node is smaller than a ratio, or percentage, of the average reward of the said node, then this child node will be removed from the set of all children unless the number of children reached the minimum requirement.

## III. NUMERICAL EXPERIMENTS AND RESULTS

In this section, we will demonstrate via numerical experiments that our algorithmic framework is suitable for various problems. Additionally, our algorithmic framework can be adapted to larger-scale problems in the future beyond the limits of simulation and into the context of implementation on physical devices. The parameters of the circuits in this section can be found in the result files contained in the open-sourced code.<sup>1</sup> The reward and loss plots for the search and training stages can be found in the Appendix.

### A. SEARCHING FOR THE ENCODING CIRCUIT OF $[[4,2,2]]$ QUANTUM ERROR DETECTION CODE

The  $[[4,2,2]]$  quantum error detection code is a simple quantum error detection code, which needs 4 physical qubits for 2 logical qubits and has a code distance of 2. It is the smallest stabilizer code that can detect  $X$ - and  $Z$ -errors [33]. One possible set of code words for the  $[[4,2,2]]$  error detection code is

$$\mathcal{E}_{[[4,2,2]]} = \text{span} \left\{ \begin{array}{l} |00\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \\ |01\rangle_L = \frac{1}{\sqrt{2}}(|0110\rangle + |1001\rangle) \\ |10\rangle_L = \frac{1}{\sqrt{2}}(|1010\rangle + |0101\rangle) \\ |11\rangle_L = \frac{1}{\sqrt{2}}(|1100\rangle + |0011\rangle) \end{array} \right\}. \quad (11)$$

The corresponding encoding circuit is shown in Fig. 6.

<sup>1</sup>[https://github.com/peiyong-addwater/QAS/blob/main/results\\_and\\_plots/](https://github.com/peiyong-addwater/QAS/blob/main/results_and_plots/).



Quantum error detection and correction are vital to large-scale fault-tolerant quantum computing. By searching for the encoding circuit of the  $[[4,2,2]]$  error detection code, we demonstrate that our algorithm has the potential to automatically find device-specific encoding circuits of quantum error detection and correction codes for future quantum processors.

### 1) EXPERIMENT SETTINGS

When searching for the encoding circuit of the  $[[4,2,2]]$  quantum error correction code, we adopted an operation pool consisting of only nonparametric operations: the Hadamard gate on each of the four qubits and CNOT gates between any two qubits. The total size of the operation pool is  $4 + \frac{4!}{2! \times 2!} \times 2 = 16$ . When there are six layers in total, the overall size of the search space is  $16^6 \approx 1.67 \times 10^7$ .

The loss function for this task is based on the fidelity between the output state of the searched circuit and the output generated by the encoding circuit from [33, Sec. 4.3] (also shown in Fig. 6) when input states taken from the set of Pauli operator eigenstates and the magic state  $|T\rangle$  are used

$$\mathcal{S} = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+\rangle, |-\rangle, |T\rangle\} \quad (12)$$

where  $|T\rangle = \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}$ .

The input states (initialized on all four qubits) are

$$\mathcal{I}_{[[4,2,2]]} = \{|\varphi_1\rangle \otimes |\varphi_2\rangle \otimes |00\rangle \mid |\varphi_1\rangle, |\varphi_2\rangle \in \mathcal{S}\} \quad (13)$$

We denote the unitary on all four qubits shown in Fig. 6 as  $U_{[[4,2,2]]}$ , and the unitary from the searched circuit as  $U_S[[4,2,2]]$ , which is a function of the structure  $\mathcal{P}_S[[4,2,2]]$ . The loss and reward function can then be expressed as

$$L_{[[4,2,2]]} = 1 - \frac{1}{|\mathcal{I}_{[[4,2,2]]}|} \sum_{|\psi_i\rangle \in \mathcal{I}} \langle \psi_i | U_S^\dagger[[4,2,2]] O_{[[4,2,2]]} U_S[[4,2,2]] | \psi_i \rangle \quad (14)$$

$$R_{[[4,2,2]]} = 1 - L_{[[4,2,2]]} \quad (15)$$

where

$$O_{[[4,2,2]]}(|\psi_i\rangle) = U_{[[4,2,2]]}|\psi_i\rangle \langle \psi_i | U_{[[4,2,2]]}^\dagger, \quad |\psi_i\rangle \in \mathcal{I}_{[[4,2,2]]}. \quad (16)$$

The circuit simulator used in this and the following numerical experiments is PennyLane [34].

### 2) RESULTS

To verify whether the search algorithm will always reach the same solution, we ran the search algorithm twice, and both times the algorithm found an encoding circuit within a small number of iterations (see Fig. 14 in the Appendix), although the actual circuit in each case is different, as shown in Fig. 7. The search process that gave the circuit in Fig. 7(a) met the early-stopping criteria in four iterations, and the search process that gave the circuit in Fig. 7(b) met the early-stopping criteria in eight iterations, as shown in Fig. 14 in the Appendix.

### B. SOLVING LINEAR EQUATIONS

The variational quantum linear solver (VQLS), first proposed in [35], is designed to solve linear systems  $Ax = b$  on near-term quantum devices. Instead of using quantum phase estimation as in the HHL algorithm [36], which is unfeasible on near-term devices due to large circuit depth, VQLS adopts a variational circuit to prepare a state  $|x\rangle$  such that

$$A|x\rangle \propto |b\rangle. \quad (17)$$

In this section, we will task our algorithm to automatically search for a variational circuit to prepare a state  $|x\rangle$  to solve  $Ax = b$  with  $A$  in the form of

$$A = \sum_l c_l A_l \quad (18)$$

where  $A_l$  are unitaries, and  $|b\rangle = H^{\otimes n}|0\rangle$ .

We will also adopt the local cost function  $C_L$  described in [35]

$$C_L = 1 - \frac{\sum_{l,l'} c_l c_{l'}^* \langle 0 | V^\dagger A_l^\dagger U P U^\dagger A_{l'} V | 0 \rangle}{\sum_{l,l'} c_l c_{l'}^* \langle 0 | V^\dagger A_l^\dagger A_{l'} V | 0 \rangle} \quad (19)$$

where  $U = H^{\otimes n}$ ,  $V$  is the (searched) variational circuit that can produce the solution state  $V|0\rangle = |x\rangle$ , and  $P = \frac{1}{2} + \frac{1}{2n} \sum_{j=0}^{n-1} Z_j$  [37].

### 1) EXPERIMENT SETTINGS

The linear system to be solved in our demonstration is

$$A = \zeta I + JX_1 + JX_2 + \eta Z_3 Z_4 \quad (20)$$

$$|b\rangle = H^{\otimes 4}|0\rangle \quad (21)$$

with  $J = 0.1$ ,  $\zeta = 1$ ,  $\eta = 0.2$ . The loss function we adopted follows the local loss  $C_L$  in (19). However, since the starting point of the loss values often has a magnitude of  $10^{-2}$ – $10^{-3}$ , we will need to scale the reward function

$$\mathcal{R} = e^{-10C_L} - \lambda \quad (22)$$

where  $\lambda$  is a penalty term depending on the number of Placeholder gates (which can be considered as identity operations, just holding a place and doing nothing) in the circuit. The operation pool consists of CNOT gates between neighboring two qubits as well as the first and fourth qubits, the Placeholder, and the single qubit rotation gate Rot [28] as follows:

$$\begin{aligned} Rot(\phi, \theta, \omega) &= R_Z(\omega) R_Y(\theta) R_Z(\phi) \\ &= \begin{bmatrix} e^{-i(\phi+\omega)/2} \cos(\theta/2) & -e^{i(\phi-\omega)/2} \sin(\theta/2) \\ e^{-i(\phi-\omega)/2} \sin(\theta/2) & e^{i(\phi+\omega)/2} \cos(\theta/2) \end{bmatrix}. \end{aligned} \quad (23)$$

Such a circular connection topology of CNOTs is more suitable for today's quantum hardware than a nonrestricted connection topology since CNOTs over nonneighboring qubits often require a lot of SWAP gates on the actual hardware, introducing more noise into the circuit. Such a four-qubit circular topology exists on Google's Sycamore [38], although this device is not publicly available. Also, from the searched

circuit shown in Fig. 8, the algorithm only used CNOT gates between neighboring qubits.

The size of the operation pool  $c = |\mathcal{C}| = 16$ , and the number of layers  $p = 10$ , giving us a search space of size  $|\mathcal{S}| = 10^{16}$ . There is also an additional restriction of the maximum number of CNOT gates in the circuit, which is 8, the number of CNOT gates required to create two layers of circular entanglement.

## 2) RESULTS

For the VQLS experiment, the search quickly terminates when the reward reaches the early-stop threshold (shown in Fig. 15 in the Appendix). During the finetune stage, the loss drops down to close to zero within 100 iterations (see Fig. 15 in the Appendix). Although facing a large search space, our algorithm can still find a circuit (shown in Fig. 8) that minimizes the loss function [see Fig. 15(b) in the Appendix] and leads us to results close to the classical solution. The mean-square error (MSE) between the VQLS results with searched circuit and the classical solution is  $9.8522 \times 10^{-6}$ . A comparison of the results obtained by directly solving the linear equation  $Ax = b$  and by sampling the state  $|x\rangle$  produced by the searched circuit is shown in Fig. 16 in the Appendix.

## C. SEARCH FOR QUANTUM CHEMISTRY ANSATZ

Recently, there has been a lot of progress made in finding the ground state energy of simple molecules on near-term quantum computers with the variational circuit, both on the theoretical [39], [40], [41] and experimental [1], [7], [42], [43], [44], [45] front. Normally, when designing the ansatz for the ground energy problem either a physically plausible or a hardware-efficient ansatz needs to be found. However, our algorithm provides an approach, which can minimize the effort needed to carefully choose an ansatz and automatically design the circuit according to the device gate set and topology.

Generally speaking, solving the ground energy problem with quantum computers is an application of the variational principle [46]

$$E_0 \leq \frac{\langle \tilde{0} | H | \tilde{0} \rangle}{\langle \tilde{0} | \tilde{0} \rangle} \quad (24)$$

where  $H$  is the system Hamiltonian,  $|\tilde{0}\rangle$  is the “trial ket” [46], or ansatz, trying to mimic the real wave function of the ground state with energy  $E_0$  (the smallest eigenvalue of the system Hamiltonian  $H$ ). Starting from  $|0^{\otimes n}\rangle$  for an  $n$ -qubit system, the “trial ket” can be written as a function of a set of (real) parameters  $\theta$

$$|\tilde{0}\rangle = |\varphi(\theta)\rangle = U(\theta)|0^{\otimes n}\rangle. \quad (25)$$

Given an ansatz, the goal of optimization is to find a set of parameters  $\theta$  that minimizes the right-hand side of (24). However, in our experiments, the form of the trial wave function will no longer be fixed—we will not only vary the

parameters but also the circuit structure that represents the ansatz.

## 1) EXPERIMENT SETTINGS

We first define the three chemistry problems before presenting the results.

### a) Search an ansatz for finding the ground energy of $H_2$

In this experiment, we adopted the 4-qubit Hamiltonian  $H_{H_2}$  for the hydrogen molecule  $H_2$  generated by the PennyLane-QChem [34] package, when the coordinates of the two hydrogen atoms are (0, 0, -0.6614) and (0, 0, 0.6614), respectively, in atom units. The goal of this experiment is to find an ansatz that can produce similar states as the four-qubit Givens rotation for single and double excitation. The unitary operator<sup>2</sup> that performs single excitation on a subspace spanned by  $\{|01\rangle, |10\rangle\}$  can be written as

$$U(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi/2) & -\sin(\phi/2) & 0 \\ 0 & \sin(\phi/2) & \cos(\phi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (26)$$

And the transformation of the double excitation on the subspace spanned by  $\{|1100\rangle, |0011\rangle\}$  is<sup>3</sup>

$$\begin{aligned} |0011\rangle &\rightarrow \cos(\phi/2)|0011\rangle + \sin(\phi/2)|1100\rangle \\ |1100\rangle &\rightarrow \cos(\phi/2)|1100\rangle - \sin(\phi/2)|0011\rangle. \end{aligned} \quad (27)$$

Following the work in [47], we initialized the circuit with the 4-qubit vacuum state  $|\psi_0\rangle = |0000\rangle$ . We denote the unitary for the searched ansatz  $U_{SA}$ , which is a function of its structure  $\mathcal{P}_{SA}$  and corresponding parameters. Then, the loss and reward functions can be written as

$$L_{H_2} = \langle \psi_0 | U_{SA}^\dagger H_{H_2} U_{SA} | \psi_0 \rangle \quad (28)$$

$$R_{H_2} = -L_{H_2}. \quad (29)$$

The operation pool consists of Placeholder, Rot, and CNOT gates with a linear entanglement topology (nearest neighbor interactions). A linear topology is more suitable for today’s NISQ hardware since we can find a line of qubits on current devices (and certainly on more advanced devices), such as the 20-qubit line (circle), which can be found on the *ibmq\_montreal* device, requiring fewer SWAP gates for the transpiled circuit. It is worth noting that the linear topology is the hardest scenario, which demonstrates our algorithm can work effectively under such harsh conditions. Also, restricting CNOT gates to only near-neighbor helps reduce the size of the operation pool and, hence, the search space, which will work better under limited sampling rounds. The maximum number of layers is 30, with a maximum number of CNOT

<sup>2</sup> <https://pennylane.readthedocs.io/en/latest/code/api/pennylane.SingleExcitation.html>

<sup>3</sup> <https://pennylane.readthedocs.io/en/latest/code/api/pennylane.DoubleExcitation.html>

gates  $30/2 = 15$ , and no penalty term for the number of Placeholder gates

$$\mathcal{R}_{H_2, \text{Pool 1}} = R_{H_2}. \quad (30)$$

Such settings of the operation pool and the number of layers will give us an overall search space of size  $14^{30} \approx 2.42 \times 10^{34}$ . However, the imposed hard limits and gate limits will drastically reduce the size of the search space.

#### b) Search an ansatz for finding the ground energy of LiH

The loss and reward functions for the LiH task are similar to  $H_2$

$$L_{\text{LiH}} = \langle \psi_0 | U_{\text{SA}}^\dagger H_{\text{LiH}} U_{\text{SA}} | \psi_0 \rangle \quad (31)$$

$$R_{\text{LiH}} = -L_{\text{LiH}} \quad (32)$$

and the initial state is also the vacuum state  $|\psi_0\rangle = |0\rangle^{\otimes 10}$ . The Hamiltonian is obtained at bond length 2.9693 Bohr, or 1.5713 Angstrom, with 2 active electrons and 5 active orbitals. The size of the operation pool  $c = |\mathcal{C}| = 38$ , including Rot gates, Placeholder, and CNOT gates operating on neighboring qubits on a line topology. The maximum number of layers is 20, giving us a search space of size  $|\mathcal{S}| = 38^{20} \approx 3.94 \times 10^{31}$ . The “hard limit” on the number of CNOT gates in the circuit is  $20/2 = 10$ .

#### c) Search an ansatz for finding the ground energy of $H_2O$

The loss and reward functions of the water molecule are shown as follows:

$$L_{H_2O} = \langle \psi_0 | U_{\text{SA}}^\dagger H_{H_2O} U_{\text{SA}} | \psi_0 \rangle \quad (33)$$

$$R_{H_2O} = -L_{H_2O} \quad (34)$$

and the initial state is also the vacuum state  $|\psi_0\rangle = |0\rangle^{\otimes 8}$ . The Hamiltonian is obtained when the three atoms are positioned at the following coordinates:

$$\begin{aligned} \text{H} : (0., 0., 0.) \\ \text{O} : (1.6323, 0.8641, 0) \\ \text{H} : (3.3609, 0., 0.). \end{aligned} \quad (35)$$

Units are in Angstrom. Active electrons are set to 4 and active orbitals are set to 4. The size of the operation pool  $c = |\mathcal{C}| = 30$ , including Rot gates, Placeholder, and CNOT gates operating on neighboring qubits on a line topology. The maximum number of layers is 20, giving us a search space of size  $|\mathcal{S}| = 30^{50} \approx 7.18 \times 10^{73}$ . The “hard limit” on the number of CNOT gates in the circuit is 25.

## 2) RESULTS

To avoid cluttering, the rewards during the search process and losses during fine-tuning can be found in Section B3. Besides restrictions on CNOT topology, we also constrained the depth and number of CNOT gates. Although the current energy estimations are good, they can be further improved toward chemical precision by increasing the searched circuit size.

Also, shorter circuits perform better in the presence of noise. Detailed insights regarding the search, fine-tuning stages, and corresponding plots can be found in the Appendix, Section B3. Generally speaking, although the rewards change is rather stochastic during the search stage, the searched circuit converges to energy values close to the reference values.

#### a) $H_2$ results

The searched ansatz is presented in Fig. 9. We can see from Fig. 9 that the unitaries are not randomly placed on the four qubit lines; instead, they present familiar structures such as the decomposition of the SWAP gate (the three CNOT gates after  $\text{Rot}(\phi_4, \theta_4, \omega_4)$ ) and Ising coupling gates ( $\text{Rot}(\phi_6, \theta_6, \omega_6)$  with the two CNOT gates on its right and left side). An example of the Ising coupling gates (often appearing in quantum optimization problems) is the  $R_{ZZ}$  gate

$$\begin{aligned} R_{ZZ}(\theta) &= e^{-i\frac{\theta}{2}Z \otimes Z} \\ &= \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \\ &= \text{CNOT}_{1,2} R_{Z_2}(\theta) \text{CNOT}_{1,2} \end{aligned} \quad (36)$$

where  $\text{CNOT}_{1,2}$  is the CNOT gate controlled by the first qubit and target on the second qubit, and  $R_{Z_2}(\theta)$  is a Z-rotation gate on the second qubit. However, other parts of the circuit are not familiar, which indicates that the search algorithm can go beyond human intuition. The total number of gates in the circuit is 22, including 13 local CNOT gates.

#### b) LiH results

The searched ansatz is presented in Fig. 10. The circuit produced by the search algorithm is simpler compared to the  $H_2$  ansatz in Fig. 9, indicating that the initial state may be very close to the ground energy state.

#### c) $H_2O$ results

The searched ansatz is presented in Fig. 20 in the Appendix due to its large depth, which has 38 gates in total, including 10 local CNOT gates. Although some familiar structures exist, such as the Ising coupling in the circuit, the heuristics behind most parts of the circuit are already unintuitive for human researchers.

## D. SOLVING THE MAXCUT PROBLEM

As a classic and well-known optimization problem, the Max-Cut problem plays an important role in network science, circuit design, as well as physics [49]. The objective of the MaxCut problem is to find a partition  $z$  of vertices in a graph  $G = (V, E)$ , which maximizes the number of edges connecting the vertices in two disjoint sets  $A$  and  $B$

$$C(z) = \sum_{a=1}^m C_a(z) \quad (37)$$

where  $C_a(z) = 1$  if the  $a$ th edge connects one vortex in set  $A$  and one vortex in set  $B$ , and  $C_a(z) = 0$  otherwise. To perform the optimization on a quantum computer, we need to transform the cost function into an Ising formulation

$$H_C = - \sum_{(i,j) \in E} \frac{1}{2} (I - Z_i Z_j) w_{ij} \quad (38)$$

where  $Z_i$  is the Pauli  $Z$  operator on the  $i$ th qubit and  $w_{ij}$  is the weight of edge  $(i, j) \in E$  for weighted MaxCut problem. For unweighted problems,  $w_{ij} = 1$ . In this formulation, vertices are represented by qubits in the computational bases. By finding the wave function that minimizes the cost Hamiltonian  $H_C$ , we can find the solution that maximizes  $C(z)$ . Previously, the major components of the quantum approximate optimization algorithm (QAOA) ansatz are the cost Hamiltonian encoded by the cost unitary and the mixing Hamiltonians encoded by the mixing unitaries [50]. Although this ansatz can find all the solutions in an equal superposition form, it is not always effective when the number of layers is small. Also, when the number of qubits (vertices) grows, the required number of layers and the number of shots during measurement to extract all of the solutions will also grow.

Since we already have a Hamiltonian as our cost function in Section III-C, we follow a similar approach as quantum chemistry to find one solution when the number of vertices is large.

In this section, we only demonstrate results for a weighted MaxCut problem. An example of an unweighted MaxCut problem is shown in the Appendix.

## 1) EXPERIMENT SETTINGS

For weighted MaxCut, we have a five-node graph, which is shown in Fig. 11. The solution for this problem 00011 (11100) is simpler than the unweighted version. The reward and loss functions follow the same principle of the unweighted problem. The size of the operation pool  $c = \mathcal{C} = 20$ , and the number of layers  $p = 10$ , leading to a search space of size  $|\mathcal{S}| = 20^{10} \approx 1.02 \times 10^{13}$ . The “hard” restriction on the maximum number of CNOTs in the circuit is 5.

## 2) RESULTS

The search rewards and fine-tune losses for the weighted MaxCut problem are shown in Fig. 21. We can see that the search converged quickly and the fine-tune loss is very close to -18, indicating that the circuit (see Fig. 12) produced by our search algorithm can, indeed, find an optimal solution (see Fig. 13).

## IV. DISCUSSION

In this article, we formulated the circuit search problem using a tree structure. The sampled circuit can be represented as an arc (path from the root to a leaf) on the tree. We also introduced the CMAB and the naive assumption to model the selection of unitary operators for each layer in the circuit and linearly approximated the rewards of different

unitaries with the reward of a fully constructed circuit. The search process is solved with the MCTS algorithm. We demonstrated the effectiveness of our algorithmic framework with various examples, including finding the encoding circuit of the  $[[4,2,2]]$  quantum error detection code, developing the ansatz for variationally solving system of linear equations, searching the circuit for solving the ground state energy problem of different molecules, as well as circuits for solving optimization problems on a graph. To our understanding, this is the first work to propose such a versatile framework for the automated discovery of quantum circuits with MCTS and CMABs.

From the numerical experiments and results shown in the previous sections, we can see that by formulating the quantum ansatz search problem as a tree-based structure, one can impose various kinds of restrictions on the circuit structure as well as pruning low-reward actions, therefore effectively reducing the size of the search space. The approach and implementation presented here can be adapted to user-specific search problems following the given examples in the code,<sup>4</sup> as long as the new search problem has a well-defined reward (and loss) function. Although in most of our experiments, we adopted linear topologies for CNOT connections, it is worth noting that even if, in the ideal simulations, an all-to-all CNOT connected graph will result in more CNOTs being selected in the circuit. When evaluating the quantum circuit on a noisy simulator or an actual NISQ device with restricted topology, nonneighboring CNOT gates will bring more SWAP gates in the transpiled circuit, hence more noise in the circuit, leading to an output state farther from the desired state, which will be reflected in the reward signal for MCTS.

## 1) RUNTIME ANALYSIS

We analyze the required runtime base on the number of circuit evaluations needed. When applying our algorithm to searching a circuit for a real quantum device, single-circuit evaluation time could be very small for large circuits compared with running on a simulator, especially for circuits with large amounts of entanglement, which are very hard to simulate classically but can be easily evaluated on a quantum device.

Based on the hyperparameters given in Table 1, we can calculate the maximum number of circuit evaluations<sup>5</sup> as follows:

$$n = [(b_w + p \times l \times 2 \times b_w) + p \times r_e] \times t_w \\ + [b_s \times r_s + p \times l \times 2 \times b_s + p \times r_e] \times t_s \quad (39)$$

where

<sup>4</sup> <https://github.com/peiyong-addwater/QAS>

<sup>5</sup> The actual number of circuit evaluations should be smaller than the calculated since the search process may stop once the reward meets the early-stop criteria. Also, the actual number of circuit evaluations required in calculating gradients with the parameter-shift method should be much smaller than the calculated ones since not every layer has parameterized operations. We only calculate the upper bound.



- 1)  $n$ : number of circuit evaluations;
- 2)  $p$ : number of layers in the circuit;
- 3)  $l$ : number of parameters per layer;
- 4)  $b_w$ : random sample batch size during warm-up iterations;
- 5)  $b_s$ : sampling size during search iterations;
- 6)  $r_s$ : number of execution rounds during search sampling;
- 7)  $r_e$ : number of execution rounds during exploitation;
- 8)  $t_w$ : number of iterations for warm-up;
- 9)  $t_s$ : number of iterations for searching.

The results and parameters are also shown in Table 1.

## 2) COMPARISON WITH OTHER RESEARCH

Other works in this area, including the differentiable quantum ansatz algorithm proposed in [16], and other QAS algorithms based on metalearning [23], or reinforcement learning [20], generally investigate small-scale problems, such as 3- or 4-qubit quantum Fourier transforms in [16], and 3-qubit classification tasks and the 4-qubit  $H_2$  ground-state energy problem in [24]. A larger example can be seen in [19], which is a 6-qubit transversal Ising field model. In comparison, we investigate some larger-scale problems with a larger search space and a wider range of applications, from solving linear equations to finding the ground state energy of small molecules.

## 3) FUTURE WORK

Several hyperparameters, such as the maximum number of gates and CNOTs in the circuit, still need to be tuned before the search algorithm can produce satisfying results, which leaves space for improvement in the automation level of the algorithm. We can see from Table 1 that the choices of the hyperparameters listed are still largely arbitrary. A future research project will run large quantities of experiments to empirically find the scaling properties of the hyperparameters with respect to the problem scale, i.e., the number of qubits, size of the operation pool, and the number of layers. In the future, possible investigations include the performance of our algorithm under noise and the improvement of the algorithm's scalability by introducing parallelization to the tree search algorithm when using a quantum simulator. Introducing more flexible value and/or policy functions into the algorithm would be a fruitful research direction. Although we did not aim for large-scale simulation in this article, it would be interesting to see how the algorithm would perform when the number of qubits and number of layers increase in the future. Also, interpretability is still a huge problem for AI/deep learning-based algorithms. Although we can observe some structure from the search results, and since the operation pools often consist of simple single qubit rotation gates and CNOTs, which will make the searched circuit look more random than circuits composed of rotation and entanglement layers, one requires more insights regarding the search process to ensure that the algorithm does not just

produce a random circuit. We also would like to investigate different quantum complexity measures and information scrambling properties for the circuits produced during the search process.

In summary, our research has shown that MCTS enhanced with CMAB is a versatile and efficient approach to search for quantum circuits for a variety of problems, even when the search space is large—addressing an important requirement in the application of variational quantum algorithms to a range of problems of interest.

## 4) DATA AVAILABILITY

The code and data for this article are hosted on <https://github.com/peiyong-addwater/QAS>.

## APPENDIX SUPPLEMENTARY MATERIALS

### A. PSEUDOCODE FOR THE TREE SEARCH PROCESS

---

#### Algorithm 1 SampleArc

---

**Input:** sample policy *Policy*, parameters of the super circuit *param*, number of rounds in sampling  $N$   
**Output:** list representation  $\mathcal{P}$  of quantum circuit  
 $curr \leftarrow \text{GetRoot}(Tr)$   $\triangleright$  Starting from the root node of the tree  $Tr$   
 $i \leftarrow 0$   $\triangleright$  Counter  
**while**  $i < N$  **do**  
   $\text{ExecuteSingleRound}(curr, \text{Policy}, param)$   
   $i \leftarrow i + 1$   
**end while**  
**while**  $curr$  is not leaf node **do**  
   $curr \leftarrow \text{SelectNode}(curr, \text{Policy})$   
**end while**  
 $\mathcal{P} \leftarrow \text{GetListRepresentation}(curr)$

---



---

#### Algorithm 2 ExploitArc

---

**Input:** exploit policy *Policy*, parameters of the super circuit *param*, number of rounds in exploitation  $N$   
**Output:** list representation  $\mathcal{P}$  of quantum circuit  
 $curr \leftarrow \text{GetRoot}(Tr)$   $\triangleright$  Starting from the root node of the tree  $Tr$   
**while**  $curr$  is not leaf node **do**  
   $i \leftarrow 0$   $\triangleright$  Counter  
  **while**  $i < N$  **do**  
     $\text{ExecuteSingleRound}(curr, \text{Policy}, param)$   
     $i \leftarrow i + 1$   
  **end while**  
   $curr \leftarrow \text{SelectNode}(curr, \text{Policy})$   
**end while**  
 $\mathcal{P} \leftarrow \text{GetListRepresentation}(curr)$

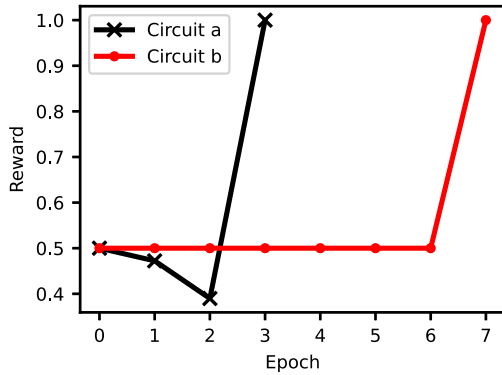
---

### Algorithm 3 SelectNode

**Input:** current node  $n$ , selection policy  $Policy$   
**Output:** selected node  $n'$   
**if**  $n$  is fully expanded **then**  
    $PruneChild(n) \triangleright$  Prune children nodes  
   according to certain threshold  
 $n' \leftarrow GetBestChild(n, Policy) \triangleright$  Select  
   the best child  
**else**  
 $n' \leftarrow ExpandChild(n) \triangleright$  Expand the node  
**end if**

### Algorithm 4 ExecuteSingleRound

**Input:** current node  $n$ , selection policy  $Policy$ ,  
 parameters of the super circuit  $param$   
**Output:** leaf node  $n'$   
 $n' \leftarrow n$   
**while**  $n'$  is not leaf node **do**  
 $n' \leftarrow SelectNode(n', Policy)$   
**end while**  
 $R \leftarrow Simulation(n', param) \triangleright$  Obtain  
 reward from simulation  
 $Backpropagate(n', R) \triangleright$  Back-propagate the  
 reward information along the arc



**FIG. 14.** Rewards when searching for encoding circuits of the  $[[4,2,2]]$  code. We can see that in both cases the algorithm was able to find the encoding circuit that generated the required code words in just a few iterations. “Circuit a” refers to the search rewards for the circuit in Fig. 7(a) and “Circuit b” refers to the search rewards for the circuit in Fig. 7(b).

## B. REWARD AND LOSS PLOTS FOR THE EXPERIMENTS

In this section, we put the plots for the search and fine-tune process of the numerical experiments.

- 1) THE  $[[4,2,2]]$  CODE EXPERIMENT
- 2) VQLS EXPERIMENT
- 3) QUANTUM CHEMISTRY EXPERIMENTS
- 4) WEIGHTED MAXCUT

### C. ADDITIONAL UNWEIGHTED MAXCUT EXPERIMENTS

#### 1) EXPERIMENT SETTINGS

The problem graph for the unweighted MAXCUT experiment is shown in Fig. 22. This problem has six equally optimal solutions: 1001100, 0110010, 0111010, 1000101, 1001101, and 0110011, all with  $C(z) = 7$ . The loss function is based on the expectation of the cost Hamiltonian  $H_C$

$$L_{\text{MAXCUT}} = (\langle + |)^{\otimes 7} U_{\text{SA}}^\dagger H_C U_{\text{SA}} (| + \rangle)^{\otimes 7}. \quad (\text{A1})$$

The reward function is simply the negative of the loss function

$$\mathcal{R}_{\text{MAXCUT}} = -L_{\text{MAXCUT}}. \quad (\text{A2})$$

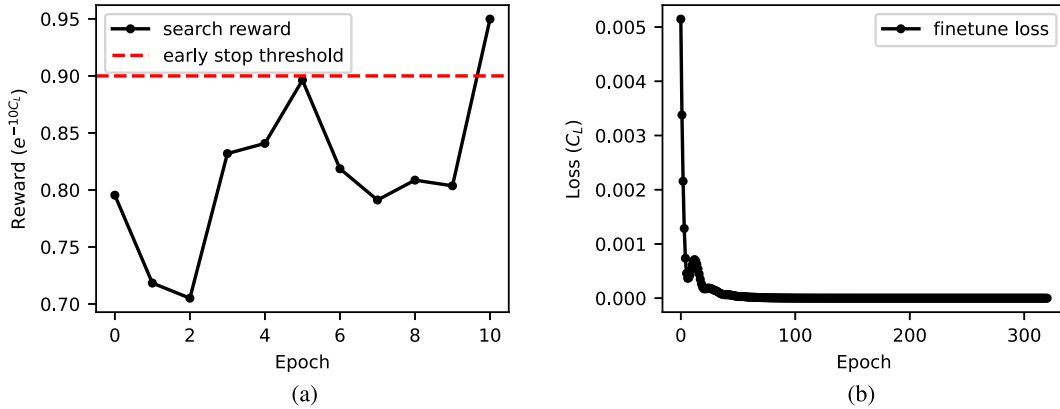
We ran the search algorithm twice with the same basic settings, including the operation pool and the maximum number of layers. Since there is a random sampling process during the warm-up stage, the final solutions found by the algorithm are expected to be different. The operation pool consists of CNOT gates between every two qubits, the Placeholder and the single qubit rotation gate [28]

$$\begin{aligned} Rot(\phi, \theta, \omega) &= R_Z(\omega) R_Y(\theta) R_Z(\phi) \\ &= \begin{bmatrix} e^{-i(\phi+\omega)/2} \cos(\theta/2) & -e^{i(\phi-\omega)/2} \sin(\theta/2) \\ e^{-i(\phi-\omega)/2} \sin(\theta/2) & e^{i(\phi+\omega)/2} \cos(\theta/2) \end{bmatrix}. \end{aligned} \quad (\text{A3})$$

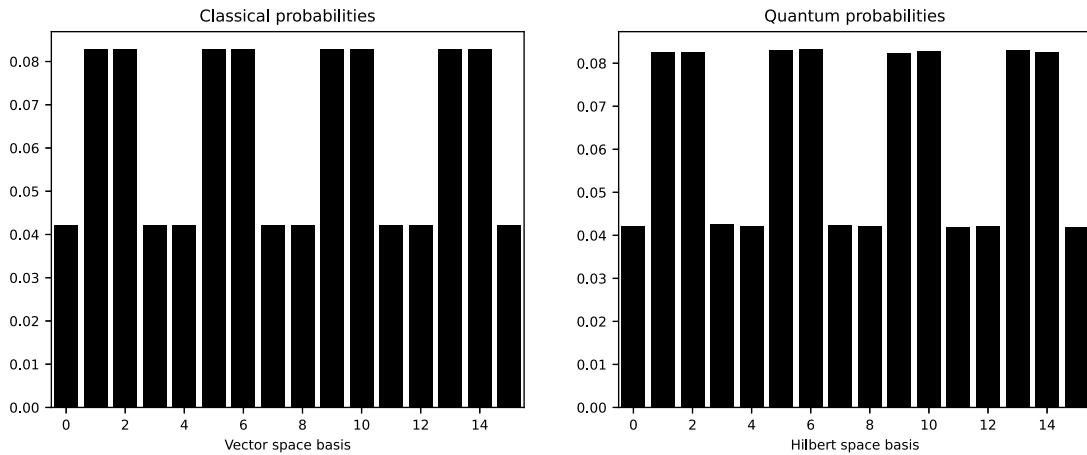
The size of the operation pool  $c = |\mathcal{C}| = 28$ , and the number of layers  $p = 15$ , leading to a search space of size  $|\mathcal{S}| = 28^{15} \approx 5 \times 10^{21}$ . The “hard” restrictions on the maximum number of CNOT gates in a circuit, which is 7, can help reduce the size of the search space.

#### 2) RESULTS

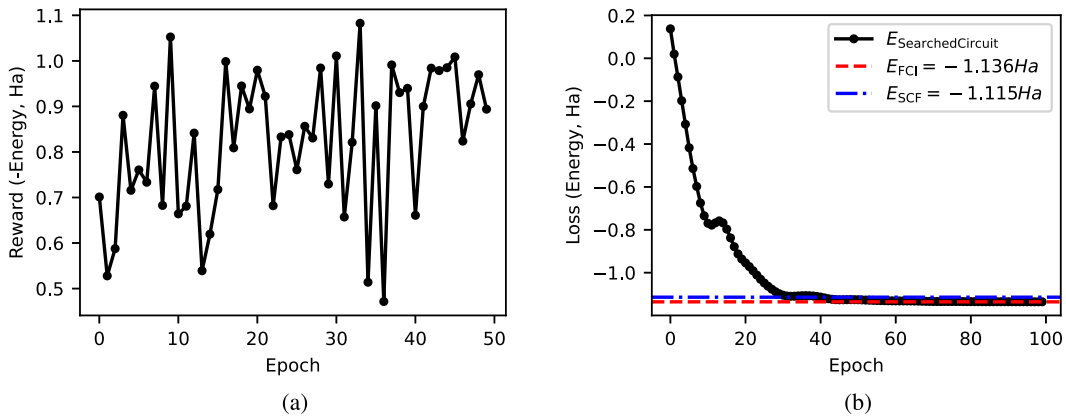
The two runs of the search algorithms gave us two circuits see Fig. 23, leading to two of the six optimal solutions (see Fig. 24). The search rewards and fine-tune losses for both circuits are shown in Fig. 25. During the search stage, since we already know the maximum reward it could reach is 7, and the reward can only be integers, we set the early-stopping limit to 6.5 to reduce the amount of time spent on searching, which means the algorithm will stop searching and proceed to fine-tuning the parameters in the circuit after the reward exceeds 6.5. In a real-world application, we could let the search algorithm run through all of the preset numbers of iterations and record the best circuit structure as well as the corresponding rewards at each iteration at the same time. Then, after the search stage finishes, we can choose the best circuit (or top- $k$  circuits) in the search history to fine-tune, increasing our chance to find the optimal solution.



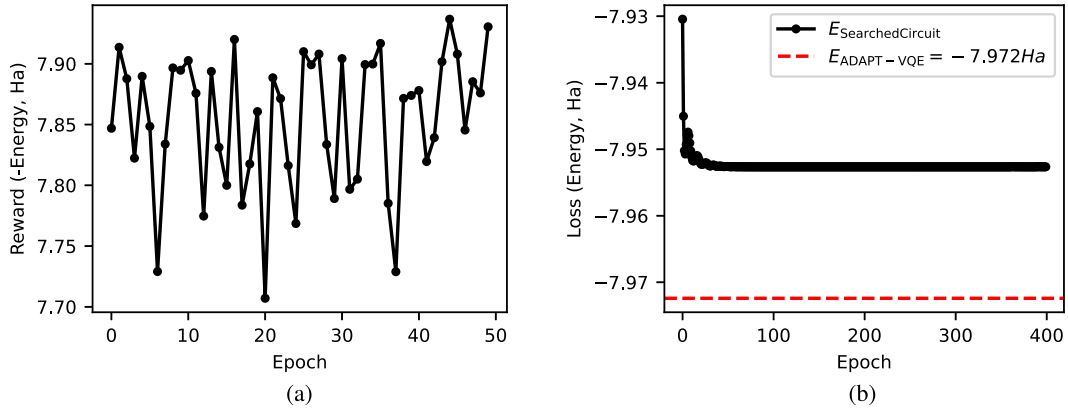
**FIG. 15.** Search rewards and fine-tune loss for VQLS experiment. The change of rewards with respect to the iterations is shown in (a). We can see that the reward quickly reached the early stopping threshold at iteration 10. In the VQLS case, the reward is scaled since the initial reward with random sampled circuit structure and parameters is already at the magnitude of  $10^{-2}$ . Fine-tune loss for the VQLS circuit is shown in (b). After the search stopped at iteration 10 shown in (a). The structure of the circuit is left unchanged and its parameters are optimized to achieve smaller losses. The final loss of the parameters is very close to 0.



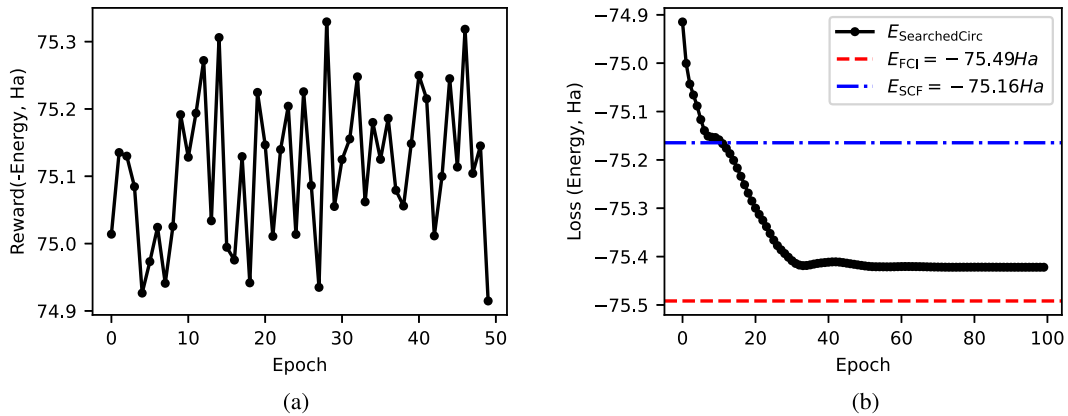
**FIG. 16.** Comparison between classical probabilities, obtained from solving the matrix equation with the classical method, i.e.,  $x = A^{-1}b$ , of the normalized solution vector  $\frac{x}{\|x\|}$  for  $Ax = b$  (left), and the probabilities obtained by sampling the state  $|x\rangle$  produced by the trained circuit in Fig. 8(right). The number of shots for measurement is  $10^6$ . We can see that the quantum results are very close to the classically obtained ones, with MSE between the VQLS results with the searched circuit and the classical solution is  $9.8522 \times 10^{-6}$ , showing that our algorithm can be applied to finding variational ansatz for VQLS problems.



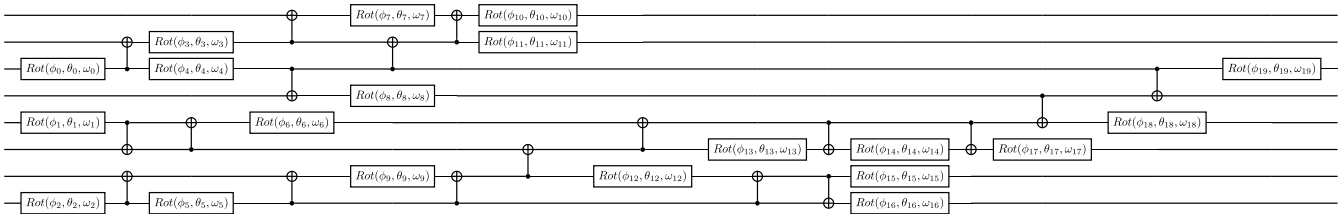
**FIG. 17.** Search rewards and fine-tune loss for  $H_2$  circuit experiment. In (a), we have the search rewards for the  $H_2$  ansatz. We can see that for most of the 50 iterations, the reward for the best circuit sampled from the search tree stays over 0.7. In (b), we have the fine-tuned loss for the searched  $H_2$  circuit. At the last iteration of optimisation, the energy is around -1.1359 Ha. The classically computed full configuration interaction result with PySCF [52]; [53]), which is around -1.132 Ha and marked by the red horizontal dashed line. We can see that our result is closer to the ground truth (FCI energy, the red dashed line) than the ground state energy computed by the self-consistent field (SCF) methods (the blue dot-dashed line).



**FIG. 18.** Search rewards and fine-tune loss for the LiH circuit experiment. In (a), we have the search rewards for the LiH ansatz. We can see that for most of the 50 iterations, the reward for the best circuit sampled from the search tree stays over 7.7 Ha. In (b), we have the fine-tune loss for the searched LiH circuit. At the last iteration of optimization, the energy is around -7.9526 Ha, closer to the energy obtained by the ADAPT-VQE method [49], but with a much shorter circuit.

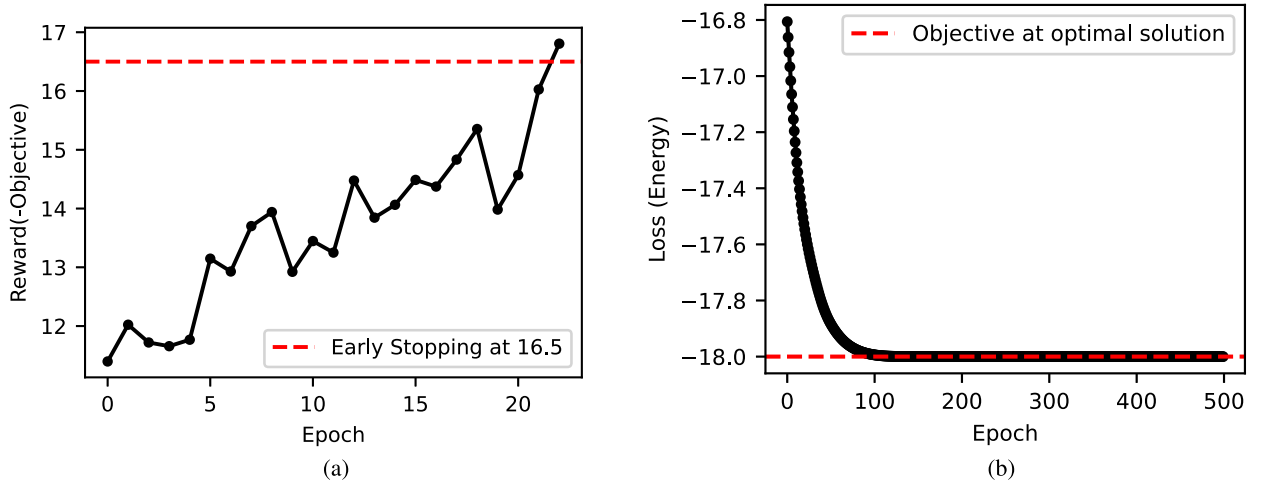


**FIG. 19.** Search rewards and fine-tune loss for the H<sub>2</sub>O circuit. In Fig. 17(a), we have the search rewards for the H<sub>2</sub>O ansatz. For most of the 50 iterations, the reward for the best circuit sampled from the search tree stays over 74.9 Ha. In (b), we have the fine-tune loss for the searched H<sub>2</sub>O circuit. At the last iteration of optimization, the energy is around -75.4220 Ha, closer to the classically computed full configuration interaction energy with PySCF [52], [53], which can be viewed as the true ground state rather than the energy obtained with the SCF method.

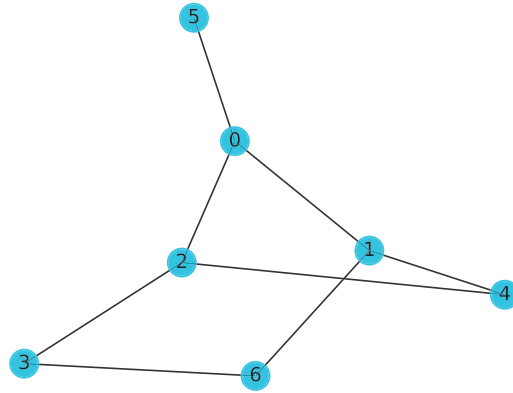


**FIG. 20.** Circuit for H<sub>2</sub>O produced by the search algorithm. We can see that our circuit respects the connectivity of the qubits, which in this case, is linear. Ising rotationlike structures also emerge as in the H<sub>2</sub> circuit in Fig. 9. We also noticed that the search algorithm's circuit is much shorter than the ansatz composed with `qml.SingleExcitation` and `qml.DoubleExcitation` from PennyLane [48] according to the molecule's physical properties, which, after removing gates with small gradient values during training, can be decomposed into 528 gates, including 276 two-qubit control gates.

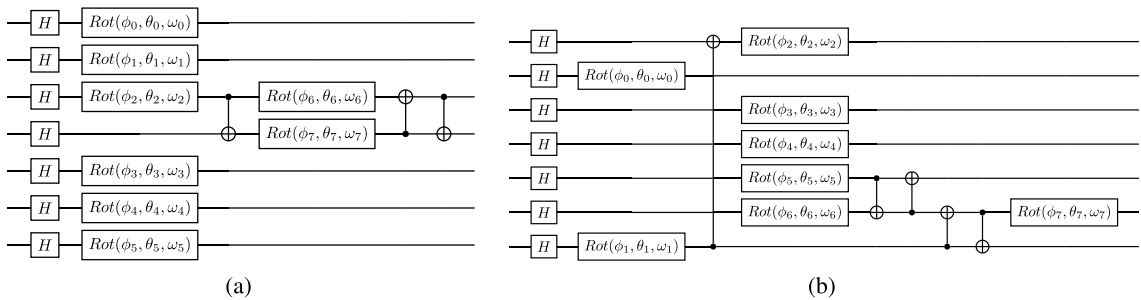




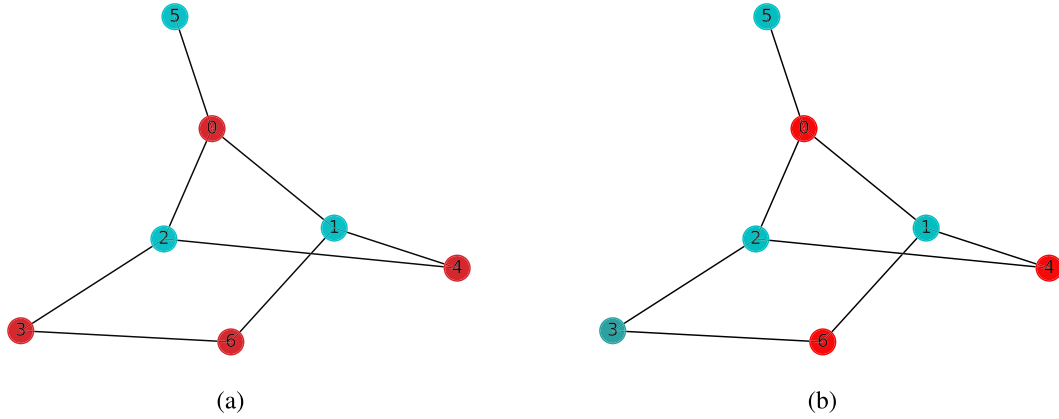
**FIG. 21.** Search rewards and fine-tune losses of for the five-node MAXCUT problem. (a) Search rewards for the five-node weighted MAXCUT problem. (b) Fine-tune loss for the five-node weighted MAXCUT problem.



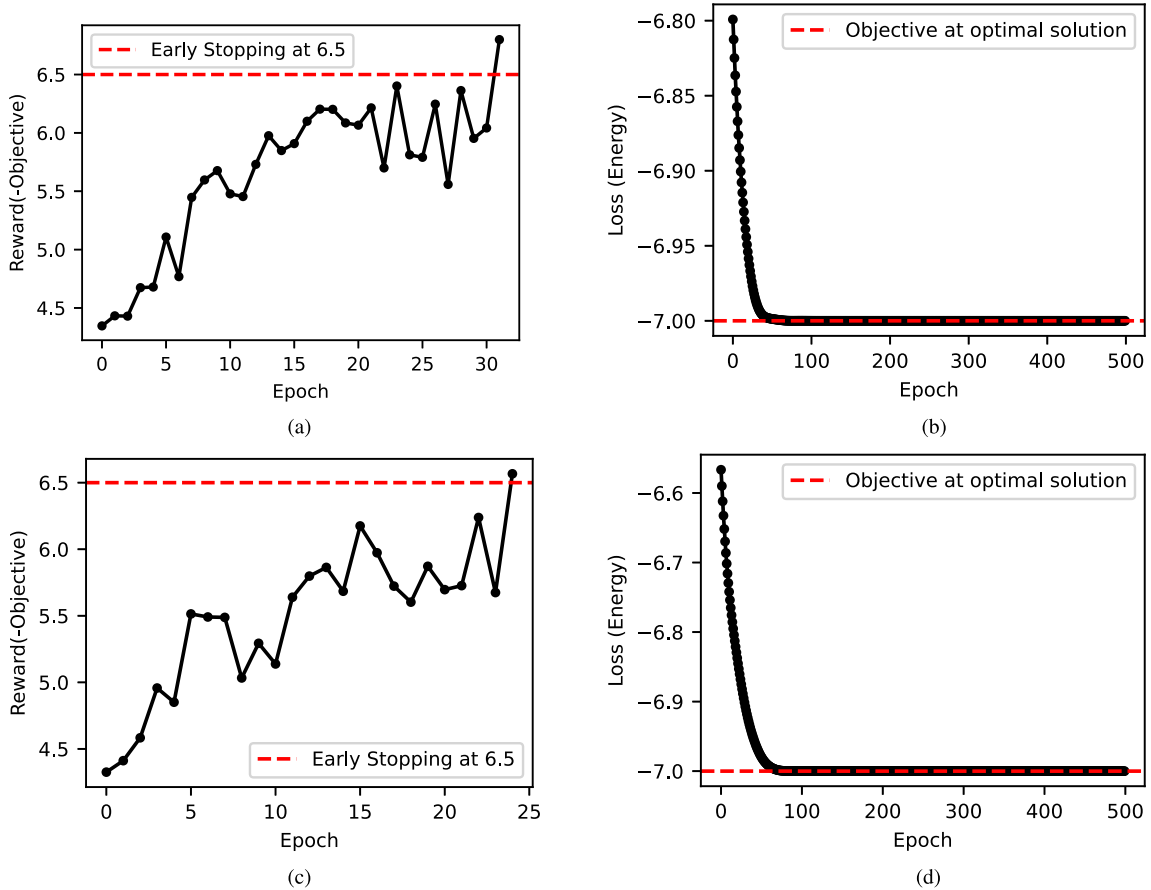
**FIG. 22.** Problem graph for the unweighted MAXCUT experiment



**FIG. 23.** Two different circuits finding two different solutions of the MAXCUT problem shown in Fig. 22. Figure (a) gives the solution 0110010 [see Fig. 24(a)] and (b) gives the solution 0111010 [see Fig. 24(b)]. If we only want one of the optimal solutions, our algorithmic framework can produce much shorter circuits than the standard QAOA ansatz, making it more favorable on current NISQ devices. From the problem graph shown in Fig. 22, the standard QAOA circuit in a single layer would require eight  $e^{-i\gamma Z_i Z_j}$ ,  $i, j \in E$  entangling gates to encode the cost Hamiltonian, as well as a layer of single-qubit rotation gates for the mix Hamiltonian, while our circuit requires just several single-qubit rotation gates and no more than five CNOT gates to find one of the optimal solutions.



**FIG. 24.** Two different optimal solutions found by the circuits in Fig. 23(a) and (b), respectively. These two different solutions are both optimal for the given problem. Figure (a) corresponds to solution 0110010, where nodes 0, 4, 3, and 6 are colored red and nodes 1, 2, and 5 are colored blue for the two partitions of the graph. Figure (b) corresponds to solution 0111010, where nodes 0, 4 and 6 are colored red and nodes 1, 2, 3, and 5 are colored blue for the two partitions of the graph.



**FIG. 25.** Search and fine-tune rewards for the circuits in Fig. 23. (a) Change of rewards w.r.t. search iteration during the search for the ansatz [in Fig. 23(a)] that gives the solution 0110010 [in Fig. 24(a)]. To reduce the amount of time for searching, we stopped the algorithm after the search reward exceeded 6.5. (b) Change of loss w.r.t. optimization iteration during the finetune for the ansatz [in Fig. 23(a)] that gives the solution 0110010 [in Fig. 24(a)]. We can see that the final loss is very close to -7, indicating that the circuit we found can produce an optimal solution. (c) Change of rewards w.r.t. search iteration during the search for the ansatz [in Fig. 23(b)] that gives the solution 0111010 [in Fig. 24(b)]. To reduce the amount of time for searching, we stopped the algorithm after the search reward exceeded 6.5. (d) Change of loss w.r.t. optimization iteration during the finetune for the ansatz [in Fig. 23(b)] that gives the solution 0111010 [in Fig. 24(b)]. We can see that the final loss is close to -7, indicating that the circuit we found can produce an optimal solution.

## ACKNOWLEDGMENT

With assistance from Casey R. Myers and Muhammad Usman, Peiyong Wang proposed and developed the research idea. Peiyong Wang implemented the algorithm and carried out the numerical experiments. All authors participated in the analysis of results. Peiyong Wang wrote the article with input from all authors.

## REFERENCES

- [1] A. Peruzzo et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Commun.*, vol. 5, no. 1, pp. 1–7, 2014, doi: [10.1038/ncomms5213](#).
- [2] M. Schuld and F. Petruccione, *Machine Learning With Quantum Computers*. Berlin, Germany: Springer, 2021, doi: [10.1007/978-3-030-83098-4](#).
- [3] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Rev. Mod. Phys.*, vol. 92, Mar. 2020, Art. no. 015003, doi: [10.1103/RevModPhys.92.015003](#).
- [4] N. Stamatopoulos et al., "Option pricing using quantum computers," *Quantum*, vol. 4, Jul. 2020, Art. no. 291, doi: [10.22331/q-2020-07-06-291](#).
- [5] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, "Qvector: An algorithm for device-tailored quantum error correction," 2017, *arXiv:1711.02249*, doi: [10.48550/arXiv.1711.02249](#).
- [6] X. Xu, S. C. Benjamin, and X. Yuan, "Variational circuit compiler for quantum error correction," *Phys. Rev. Appl.*, vol. 15, no. 3, Nov. 2021, Art. no. 034068, doi: [10.1103/PhysRevApplied.15.034068](#).
- [7] A. Kandala et al., "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017, doi: [10.1038/nature23879](#).
- [8] J. Lee, W. J. Huggins, M. Head-Gordon, and K. B. Whaley, "Generalized unitary coupled cluster wave functions for quantum computation," *J. Chem. Theory Computation*, vol. 15, no. 1, pp. 311–324, 2019, doi: [10.1021/acs.jctc.8b01004](#).
- [9] J. Jumper et al., "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, doi: [10.1038/s41586-021-03819-2](#).
- [10] A. Davies et al., "Advancing mathematics by guiding human intuition with AI," *Nature*, vol. 600, no. 7887, pp. 70–74, Dec. 2021, doi: [10.1038/s41586-021-04086-x](#).
- [11] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, *arXiv:1806.09055*, doi: [10.48550/arXiv.1806.09055](#).
- [12] C. Liu et al., "Progressive neural architecture search," *Computer Vision—ECCV* (ser. Lecture Notes in Computer Science), Cham, Switzerland: Springer, 2018, vol. 11205, pp. 19–35, doi: [10.1007/978-3-030-01246-5\\_2](#).
- [13] L. Wang, Y. Zhao, Y. Jinnai, Y. Tian, and R. Fonseca, "Neural architecture search using deep neural networks and Monte Carlo tree search," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innov. Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, 2020, pp. 9983–9991, doi: [10.1609/aaai.v34i06.6554](#).
- [14] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: [10.1038/nature16961](#).
- [15] H. Huang, X. Ma, S. M. Erfani, and J. Bailey, "Neural architecture search via combinatorial multi-armed bandit," in *Proc. Int. Joint Conf. Neural Netw.*, 2021, pp. 1–8, doi: [10.1109/IJCNN52387.2021.9533655](#).
- [16] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Differentiable quantum architecture search," *Quantum Sci. Technol.*, vol. 7, no. 4, Aug. 2022, Art. no. 045023, doi: [10.1088/2058-9565/ac87cd](#).
- [17] Y. Shu, W. Wang, and S. Cai, "Understanding architectures learnt by cell-based neural architecture search," Sep. 2019, *arXiv:1909.09569*, doi: [10.48550/arXiv.1909.09569](#).
- [18] P. Zhou, C. Xiong, R. Socher, and S. C. H. Hoi, "Theory-inspired path-regularized differential network architecture search," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 8296–8307, doi: [10.48550/arXiv.2006.16537](#).
- [19] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Neural predictor based quantum architecture search," *Mach. Learn.: Sci. Technol.*, vol. 2, no. 4, Oct. 2021, Art. no. 045027, doi: [10.1088/2632-2153/ac28dd](#).
- [20] E.-J. Kuo, Y. L. L. Fang, and S. Y.-C. Chen, "Quantum architecture search via deep reinforcement learning," 2021, *arXiv:2104.07715*, doi: [10.48550/arXiv.2104.07715](#).
- [21] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104. [Online]. Available: <http://proceedings.mlr.press/v80/pham18a.html>
- [22] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, vol. 12, 1999. [Online]. Available: <http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf>
- [23] Z. He, C. Chen, L. Li, S. Zheng, and H. Situ, "Quantum architecture search with meta-learning," *Adv. Quantum Technol.*, vol. 5, Jun. 2022, Art. no. 2100134, doi: [10.1002/qute.202100134](#).
- [24] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao, "Quantum circuit architecture search for variational quantum algorithms," *npj Quantum Inf.*, vol. 8, no. 1, pp. 1–8, May 2022, doi: [10.1038/s41534-022-00570-y](#).
- [25] K. Linghu et al., "Quantum circuit architecture search on a superconducting processor," Jan. 2022, *arXiv:2201.00934*, doi: [10.48550/arXiv.2201.00934](#).
- [26] F.-X. Meng, Z.-T. Li, X.-T. Yu, and Z.-C. Zhang, "Quantum circuit architecture optimization for variational quantum eigensolver via Monte Carlo tree search," *IEEE Trans. Quantum Eng.*, vol. 2, 2021, Art. no. 3103910, doi: [10.1109/TQE.2021.3119010](#).
- [27] S. Ontañón, "The combinatorial multi-armed bandit problem and its application to real-time strategy games," in *Proc. 9th AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2013, pp. 58–64, doi: [10.1609/ai-ide.v9i1.12681](#).
- [28] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000, doi: [10.1017/CBO9780511976667](#).
- [29] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017, doi: [10.1038/nature24270](#).
- [30] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo tree search: A new framework for game AI," in *Proc. 4th AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2008, pp. 216–217, doi: [10.1609/ai-ide.v4i1.18700](#).
- [31] T. Cazenave, "Nested Monte-Carlo search," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 456–461. [Online]. Available: <https://www.ijcai.org/Proceedings/09/Papers/083.pdf>
- [32] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Mar. 2003. [Online]. Available: <https://www.jmlr.org/papers/volume3/auer02a/auer02a.pdf>
- [33] J. Roffe, "Quantum error correction: An introductory guide," *Contemporary Phys.*, vol. 60, no. 3, pp. 226–245, 2019, doi: [10.1080/00107514.2019.1667078](#).
- [34] V. Bergholm et al., "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2020, *arXiv:1811.04968*, doi: [10.48550/arXiv.1811.04968](#).
- [35] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, "Variational quantum linear solver," 2019, *arXiv:1909.05820*, doi: [10.48550/arXiv.1909.05820](#).
- [36] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, Oct. 2009, Art. no. 150502, doi: [10.1103/PhysRevLett.103.150502](#).
- [37] Variational quantum linear solver – PennyLane, Accessed: Apr. 24, 2022. [Online]. Available: [https://pennylane.ai/qml/demos/tutorial\\_vqls.html](https://pennylane.ai/qml/demos/tutorial_vqls.html)
- [38] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: [10.1038/s41586-019-1666-5](#).
- [39] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Phys. Rev. X*, vol. 7, no. 2, 2017 Art. no. 021050, doi: [10.1103/PhysRevX.7.021050](#).
- [40] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New J. Phys.*, vol. 18, no. 2, 2016, Art. no. 023023, doi: [10.1088/1367-2630/18/2/023023](#).
- [41] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Phys. Rev. A*, vol. 92, no. 4, 2015, Art. no. 042303, doi: [10.1103/PhysRevA.92.042303](#).

- [42] P. J. J. O'Malley et al., "Scalable quantum simulation of molecular energies," *Phys. Rev. X*, vol. 6, no. 3, 2016 Art. no. 031007, doi: [10.1103/PhysRevX.6.031007](https://doi.org/10.1103/PhysRevX.6.031007).
- [43] J. Colless et al., "Implementing a variational quantum eigensolver using superconducting qubits," in *Proc. Quantum Inf. Meas.*, 2017, Paper QF6A.2, doi: [10.1364/QIM.2017.QF6A.2](https://doi.org/10.1364/QIM.2017.QF6A.2).
- [44] J. I. Colless et al., "Computation of molecular spectra on a quantum processor with an error-resilient algorithm," *Phys. Rev. X*, vol. 8, no. 1, 2018 Art. no. 011021, doi: [10.1103/PhysRevX.8.011021](https://doi.org/10.1103/PhysRevX.8.011021).
- [45] E. F. Dumitrescu et al., "Cloud quantum computing of an atomic nucleus," *Phys. Rev. Lett.*, vol. 120, no. 21, 2018, Art. no. 210501, doi: [10.1103/PhysRevLett.120.210501](https://doi.org/10.1103/PhysRevLett.120.210501).
- [46] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2017, doi: [10.1017/9781108499996](https://doi.org/10.1017/9781108499996).
- [47] A. Delgado, "A brief overview of VQE," Jan. 2020, Accessed: Jun. 13, 2023. [Online]. Available: [https://pennylane.ai/qml/demos/tutorial\\_vqe.html](https://pennylane.ai/qml/demos/tutorial_vqe.html)
- [48] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," *Nat. Commun.*, vol. 10, no. 1, Jul. 2019, Art. no. 3007, doi: [10.1038/s41467-019-10988-2](https://doi.org/10.1038/s41467-019-10988-2).
- [49] K. Bharti et al., "Noisy intermediate-scale quantum algorithms," *Rev. Mod. Phys.*, vol. 94, no. 1, Feb. 2022, Art. no. 015004, doi: [10.1103/RevModPhys.94.015004](https://doi.org/10.1103/RevModPhys.94.015004).
- [50] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*, doi: [10.48550/arXiv.1411.4028](https://doi.org/10.48550/arXiv.1411.4028).
- [51] Q. Sun et al., "PySCF: The python-based simulations of chemistry framework," *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, vol. 8, no. 1, Jan. 2018, Art. no. e1340, doi: [10.1002/wcms.1340](https://doi.org/10.1002/wcms.1340).
- [52] Q. Sun et al., "Recent developments in the PySCF program package," *J. Chem. Phys.*, vol. 153, no. 2, Jul. 2020, Art. no. 024109, doi: [10.1063/5.0006074](https://doi.org/10.1063/5.0006074).