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

Digital Object Identifier 10.1109/TQE.2020.DOI

Automated Quantum Circuit Design with Nested Monte Carlo Tree Search

FIRST A. AUTHOR[1], (Fellow, IEEE), SECOND B. AUTHOR[2], AND THIRD C. AUTHOR, JR.[3], (Member, IEEE)

[1]National Institute of Standards and Technology, Boulder, CO 80305 USA (email: author@boulder.nist.gov)
[2]Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (email: author@lamar.colostate.edu)
[3]Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA

Corresponding author: First A. Author (email: author@ boulder.nist.gov).

This paragraph of the first footnote will contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

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 ansätzs remain key challenges. In this work, we report an algorithmic framework based on nested Monte-Carlo Tree Search (MCTS) coupled with the combinatorial multi-armed bandit (CMAB) model for the automated design of quantum circuits. Through numerical experiments, we demonstrated our algorithm applied to various kinds of problems, including the ground energy problem in quantum chemistry, quantum optimisation on a graph, solving systems of linear equations, and finding encoding circuit for quantum error detection codes. Compared to the existing approaches, the results indicate that our circuit design algorithm can explore larger search spaces and optimise quantum circuits for larger systems, showing both versatility and scalability.

INDEX TERMS Enter key words or phrases in alphabetical order, separated by commas. For a list of suggested keywords, send a blank email to keywords@ieee.org or visit http://www.ieee.org/organizations/pubs/ani_prod/keywrd98.txt

I. INTRODUCTION

The variational quantum circuit (VQC, also known as parameterised quantum circuit, PQC) approach, first proposed for solving the ground state energy of molecules [1], have 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 depend on the choice of a suitable ansätze, 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 ansätze. For problems involving physical systems such as in quantum chemistry, we can rely on the well-defined properties of molecular systems for ansätz designing, like the hardware efficient ansätze [7] and physical-inspired ansätze, such as k-UpCCGSD [8]. However, this cannot be generalised to other areas such as designing variational error correction circuits or quantum optimisation problems. For example, in [6], when developing a variational circuit that can encode logical states for the 5-qubit quantum error correction code, the authors adopted an expensive approach by randomly searching over a large number (order of 10000) 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 ansätzes. Therefore, it is important to develop efficient methods for the automated design of variational quantum circuits. Here we focus on the development of 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 mechanism for the bonding region of the protein and inhibitors [9] with competitive accuracy on predicting the three-dimensional 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 variational quantum circuits, 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 efficiently search suitable network architectures in a large search space. Famous algorithms for neural architecture search (NAS) include the DARTS algorithm [11], which models the choice of operations placed in different layers as an independent categorical probabilistic model that can be optimised via gradient descent methods, and the PNAS algorithm [12], which models the search process with sequential model-based optimisation (SMBO) strategy. Tree-based algorithms were also proposed for NAS, such as AlphaX [13], which models the search process similarly as the search stage of AlphaGo [14]. Recently, a new NAS algorithm based on tree search and combinatorial multi-armed bandits, proposed in [15], outperforms other NAS algorithms, including the previously mentioned algorithms.

Based on progress in neural architecture search algorithms, efforts have been made on developing similar approaches for Quantum Ansätz (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 will update the parameters in the VQC as well as the probabilistic model. However, it has been shown in NAS literature that DARTS tend to assign fast-converge architectures with high probability during sampling [17], [18]. Also, the off-the-shelf probabilistic distributions for modelling 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 parameterised quantum circuits 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 proved through many experiments that one neural network architecture can act as a backbone feature extractor for many downstream tasks, the structures of variational quantum circuits for different problems often vary a great deal with different problems, casting some doubts on the generalisation 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 optimised by the advantage actor-critic and proximal policy optimisation algorithms. However, NAS algorithms based on policy gradient reinforcement learning have been shown to get easily stuck in local minimal, producing less optimal solutions [21], [22]. Also, the data size for training a reinforcement learning agent will explode 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 neural architecture search, 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 optimisation. Later Linghu *et.al* [25] applied similar techniques on search to a classification circuit on a physical quantum processor. Meng *et.al* [26] applied Monte-Carlo tree search to ansätz optimisation for problems in quantum chemistry and condensed matter physics. However, these studies often restrict their demonstrations within one or two types of problems and small-sized systems.

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 combinatorial multi-armed bandit (CMAB) model as well as Monte-Carlo Tree Search (MCTS). In order to explore extremely large search spaces compared to previous work in the literature, the working 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 naïve assumption [27]. This enabled our strategy to work on larger systems, more than 7 qubits, whereas the existing examples [16], [16], [19], [20], [23], [24] are restricted to typically 3 or 4 qubits, with the largest being 6 qubits. To demonstrate the working of our method, we showed 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 ansätz for solving optimisations problems. Our work confirms that the automated quantum architecture search based on the MCTS+CMAB approach exhibits great versatility and scalability, and therefore should provide an efficient solution and new insights to the problems of designing variational quantum circuits.

This paper is organised as follows: Section II introduces the basic notion of Monte-Carlo tree search, as well as other techniques required for our algorithm, including nested MCTS and naïve assumptions from the CMAB model. Section **??** 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 ansätz circuit for finding the ground state energy of different molecules, as well as circuits for solving linear system of equations and optimisation. In Section **??** we discuss the results and conclusions.

II. METHODS

A. PROBLEM FORMULATION

In this paper, we formulate the quantum ansätz search problem, which is aimed to automatically design variational quantum circuits 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

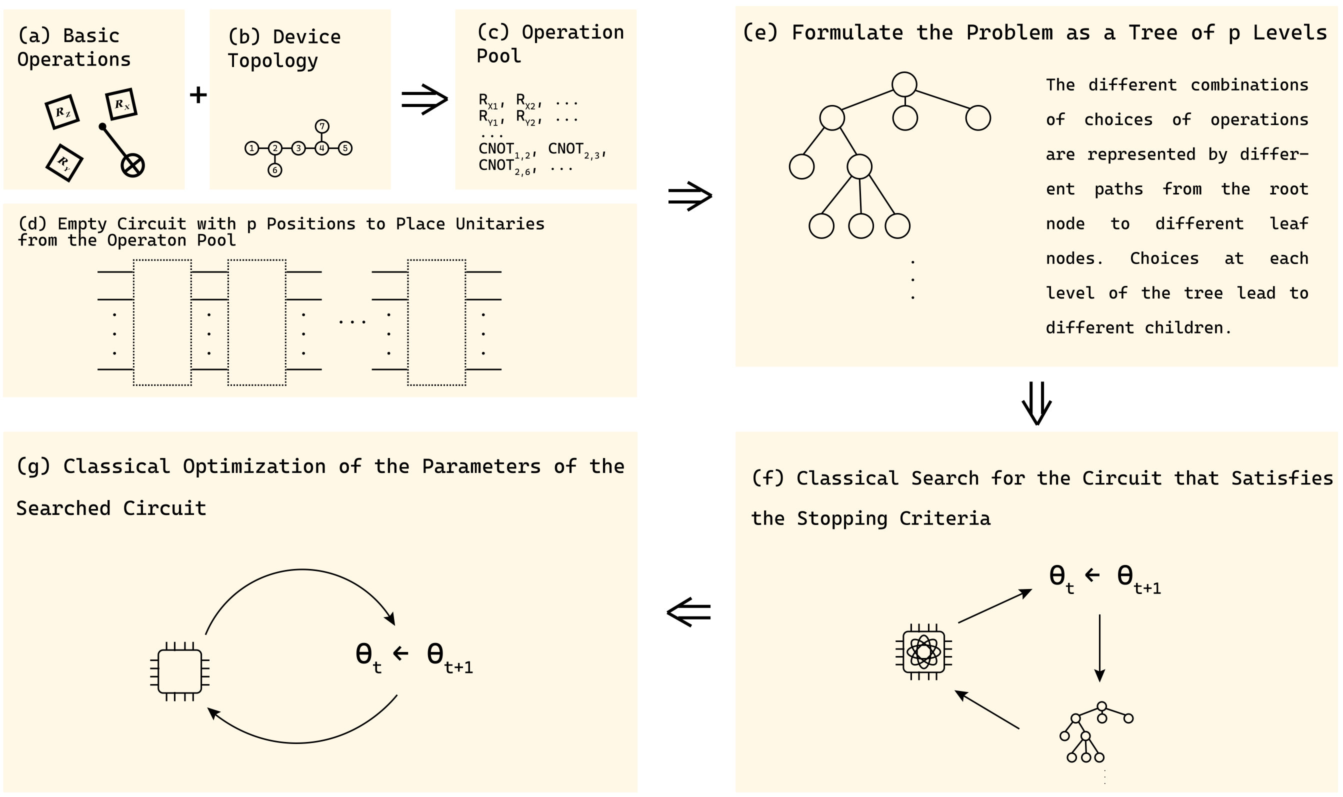
(a) Basic Operations
+
(b) Device Topology
(c) Operation Pool
R_X1, R_X2, ...
R_Y1, R_Y2, ...
CNOT_1,2, CNOT_2,3,
CNOT_2,6, ...
(d) Empty Circuit with p Positions to Place Unitaries from the Operaton Pool
(e) Formulate the Problem as a Tree of p Levels
The different combinations of choices of operations are represented by different paths from the root node to different leaf nodes. Choices at each level of the tree lead to different children.
(f) Classical Search for the Circuit that Satisfies the Stopping Criteria
θt ← θt+1
(g) Classical Optimization of the Parameters of the Searched Circuit
θt ← θt+1

FIGURE 1: An overview of the algorithmic framework proposed in this paper. 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 position in the circuit (d) as a search tree (e). In (f), we evaluate our circuit on a quantum processor or quantum simulator to get 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 variational quantum circuit by classical-quantum hybrid computing..

circuit, we fill the layers with operations chosen by the search algorithm, from the first to the final layer.

A quantum circuit is represented as a (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

$$\mathcal{C} = \{U_0, U_1, \cdots, U_{c-1}\}, \tag{1}$$

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 parameterised (e.g. the $R_Z(\theta)$ gate), or non-parameterised (e.g. the Pauli gates). A quantum circuit with four layers could, for instance, be represented as:

$$\mathcal{P} = [U_0, U_1, U_2, U_1], \tag{2}$$

where, according to the search algorithm, the operations chosen for the first, second, third and fourth layer are U_0, U_1, U_2, U_1. In this case, $p = 4$ and the size of the operation pool $|\mathcal{C}| = c$. The search tree is shown in Fig. 3 In this paper, we will only deal with unitary operations or unitary channels.

The output state of such a quantum circuit can then be written as:

$$|\varphi_{out}\rangle = U_1 U_2 U_1 U_0 |\varphi_{init}\rangle, \tag{3}$$

where $|\varphi_{init}\rangle$ 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, like the mixing Hamiltonians often seen in typical QAOA circuits, or just single- and two-qubit gates).

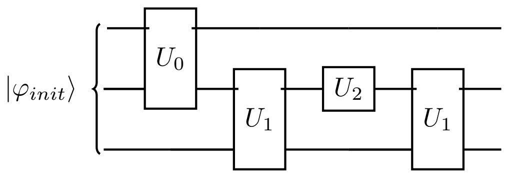

FIGURE 2: An example of the circuit corresponding to the series of unitaries applied to $|\varphi_{init}\rangle$ in Eqn.3.

For example, the quantum circuit from Eqn. 3 can be written as:

$$\mathcal{P} = [0, 1, 2, 1] \tag{4}$$

and the operation at the i^{th} layer can be referred as k_i. For example, in the quantum circuit above, we have $k_2 = 1$.

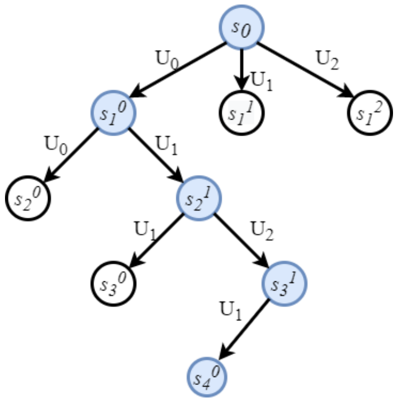

FIGURE 3: The tree representation (along the arc with blue-shaded circles) of the unitary described in Eqns. 3 and 4 as well as Fig. 2. The circle with s_0 is the root of the tree, which represents an empty circuit. Other circles with s_i^j in it denote the j^{th} node at the i^{th} level of the tree. 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 labelled s_2^0, indicating that it is the 0^{th} 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 2 layers. We can also see that some of the possible branches along the blue-node path are pruned, leading to the size of operation pool at some node smaller than the total number of possible choices $c = |\mathcal{C}|$.

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 $\boldsymbol{\theta}$:

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

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

where λ is some penalty function that may only appear when certain circuit structures appear, as well as other kinds of penalty terms, like penalty on the sum of absolute value of weights or the number of certain type of gates in the circuit; L and R are the loss/reward before applying the penalty. The purpose of the penalty term λ is 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 $U3$ gate [28]:

$$U3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{7}$$

as well as its controlled version $CU3$ 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 is the same 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 $\mathcal{P}' = [U_0, U_1, U_1, \cdots]$, 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 the U_0 and U_1 in both circuits, respectively. Such a strategy is often called "parameter-sharing" or "weight-sharing" in the neural architecture search literature.

As shown in Fig 3 and mentioned earlier, the process of composing or searching a circuit can be formulated in the form of the 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 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.

The process of choosing operations at each layer can be viewed as a both a *local* and *global* multi-armed bandit (MAB). A multi-armed bandit, just as its name indicates, is similar to a bandit, or 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 (maybe Squidward) 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 maximise his rewards from the machine(s). If we consider the whole circuit search problem as an MAB (the global MAB, 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 of the circuits 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 number of layers increase makes it impossible to perform an informed search for suitable solutions while training every circuit we encountered during the search process. Since our circuit is basically a combination of different choice 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 from 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, 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 next section, we will introduce the naïve 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.

- *Local MAB*: The choice of unitary operations at each layer can be considered a *local* MAB. That is, different unitary operations can be treated as different “arms” of the bandit;
- *Global MAB*: We can also treat the composition of the entire quantum circuit as a *global* MAB. That is, different quantum circuits can be viewed as different “arms” of the global bandit.

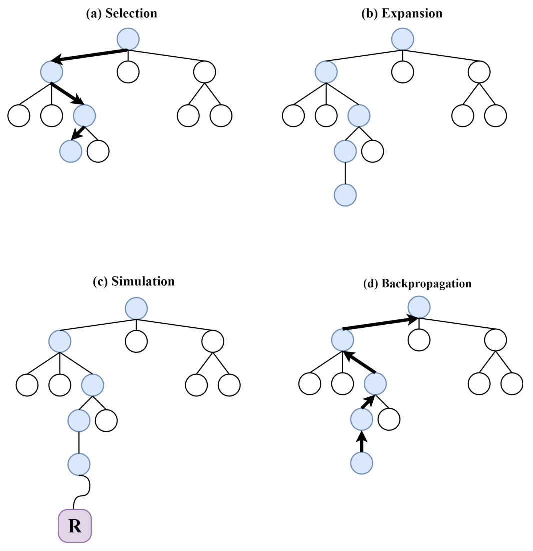

FIGURE 4: Four stages of Monte Carlo tree search. From left to right, up to down: Selection: Go down from the root node to a non fully expanded leaf node; Expansion: Expand the selected node by taking an action; Simulation: Simulate the game, which in our case is the quantum circuit, to obtain reward information **R**; Backpropagation: Back-propagation of the reward information along the path (arc) taken.

B. MONTE CARLO TREE SEARCH (MCTS), NESTED MCTS AND THE NAÏVE ASSUMPTION

Monte Carlo tree search (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 four stages in a single iteration of MCTS (see Fig. 4) [30]:

- Selection:(Fig.4(a)) In the selection stage, the algorithm will, starting 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 as the “selection policy”, until a non fully 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, then we can progress to the next stage;
- Expansion:(Fig.4(b)) In the expansion stage, at the node selected in the previous stage, we choose a previously unvisited child 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;
- Simulation:(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 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 loss $\mathcal{L}$ (or reward $\mathcal{R}$). Usually, the loss $\mathcal{L}$ is required to update the parameters in the circuit;
- Backpropagation:(Fig.4(d)) In this stage, the reward information obtained from the simulation stage is back-propagated through the arc leading from the root of the tree to the leaf node, and the number of visits as well as the (average) reward for each node along the arc is be 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 sub-trees 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.

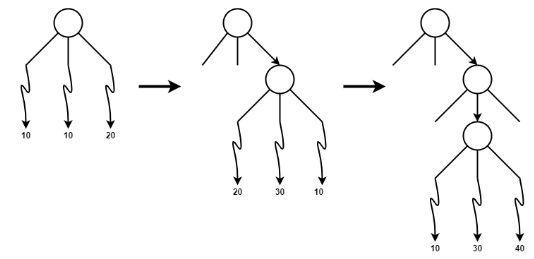

FIGURE 5: Nested Monte Carlo tree search. Left: The root node has three possible actions, which in this case are unselected initially. We perform MCTS on all three children nodes (generated by the three possible actions) to update their reward information. After one iteration of MCTS with each child as root node for the search tree that MCTS performed on, the rewards of these three actions leading to the three child nodes are 10, 10, 20, respectively. In this case, the right child 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.

We denote a quantum circuit with p layers $\mathcal{P} = [k_1, \cdots, k_p]$, 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). Then each choice for layer k_i is 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 naïve assumption [27], the global reward R_{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, then the local rewards can be approximated from the global reward. With the naïve assumption, we can have a linear relationship between the global reward and local rewards:

$$R_{global} = \frac{1}{p}\sum_{i=1}^{p} R_i \tag{8}$$

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

$$R_i \approx R_{global} \tag{9}$$

where R_i is the reward for pulling an arm at *local MAB_i* and R_{global} is the reward for the global arm. Also, if we use the naïve assumption, we will not need to directly optimise on 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.

In the original work on nested MCTS [31], a random policy was adopted for sampling. In this paper we will instead change it to the famous UCB policy [32]. Given a local MAB_i, with the set of all the possible choices k_i, the UCB policy can be defined as:

$$UCB : \underset{arm_j \in k_i}{\operatorname{argmax}} \bar{R}(k_i, arm_j) + \alpha\sqrt{\frac{2\ln n_i}{n_j}} \tag{10}$$

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

For small α, the actual reward from the bandit will play a more important role in the UCB modified rewards, which will lead to selecting actions with previously observed high rewards. When α is large enough, the second term, which will be relatively large if MAB_i has been visited many times but arm_j of 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 NAÏVE MCTS

Generally, a single iteration for the search algorithm will include two steps for non-parameterised circuits, and two more parameter-related steps for parameterised quantum circuits. 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 2.1. 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 parameterised quantum circuits (PQC), it is common practice to "warm-up" the parameters by randomly sampling a batch of quantum circuits, calculating the averaged gradient, and update 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, we have:

1) Sample a batch of quantum circuits from the super circuit with Algorithm 1;
2) (For PQCs) Calculate the averaged gradients of the sampled batch, add noise to the gradient to guide the optimiser 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 Algorithm 2.

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

With the naïve 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.

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
    $curr \leftarrow GetRoot(Tr)$    ▷ Starting from the root node of the tree $Tr$
    $i \leftarrow 0$    ▷ Counter
    while $i < N$ do
        $ExecuteSingleRound(curr, Policy, param)$
        $i \leftarrow i + 1$
    end while
    while $curr$ is not leaf node do
        $curr \leftarrow SelectNode(curr, Policy)$
    end while
    $\mathcal{P} \leftarrow GetListRepresentation(curr)$
```

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
    $curr \leftarrow GetRoot(Tr)$    ▷ Starting from the root node of the tree $Tr$
    while $curr$ is not leaf node do
        $i \leftarrow 0$    ▷ Counter
        while $i < N$ do
            $ExecuteSingleRound(curr, Policy, param)$
            $i \leftarrow i + 1$
        end while
        $curr \leftarrow SelectNode(curr, Policy)$
    end while
    $\mathcal{P} \leftarrow GetListRepresentation(curr)$
```

Algorithm 3 SelectNode

```
Input: current node $n$, selection policy $Policy$
Output: selected node $n'$
    if $n$ is fully expanded then
        $PruneChild(n)$    ▷ Prune children nodes according to certain threshold
        $n' \leftarrow GetBestChild(n, Policy)$    ▷ Select the best child
    else
        $n' \leftarrow ExpandChild(n)$    ▷ 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
    $R \leftarrow Simulation(n', param)$    ▷ Obtain reward from simulation
    $Backpropagate(n', R)$    ▷ Back-propagate the reward information along the arc
```

REFERENCES

[1] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. Nature communications, 5(1):1–7, 2014.

[2] Maria Schuld and Francesco Petruccione. Machine learning with Quantum Computers. Springer, 2021.

[3] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. Rev. Mod. Phys., 92:015003, Mar 2020.

[4] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option pricing using quantum computers. Quantum, 4:291, Jul 2020.

[5] Peter D. Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and Alán Aspuru-Guzik. Qvector: an algorithm for device-tailored quantum error correction, 2017.

[6] Xiaosi Xu, Simon C Benjamin, and Xiao Yuan. Variational circuit compiler for quantum error correction. Physical Review Applied, 15(3), November 2021.

[7] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature, 549(7671):242–246, Sep 2017.

[8] Joonho Lee, William J. Huggins, Martin Head-Gordon, and K. Birgitta Whaley. Generalized unitary coupled cluster wave functions for quantum computation. Journal of Chemical Theory and Computation, 15(1):311–324, 2019.

[9] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873):583–589, August 2021.

[10] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding human intuition with AI. Nature, 600(7887):70–74, December 2021.

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[12] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, Computer Vision – ECCV 2018, pages 19–35, Cham, 2018. Springer International Publishing.

[13] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. Neural architecture search using deep neural networks and monte carlo tree search. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 9983–9991. AAAI Press, 2020.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. Nat., 529(7587):484–489, 2016.

[15] Hanxun Huang, Xingjun Ma, Sarah M. Erfani, and James Bailey. Neural architecture search via combinatorial multi-armed bandit. In International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021, pages 1–8. IEEE, 2021.

[16] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search, 2021.

[17] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. September 2019.

[18] Pan Zhou, Caiming Xiong, Richard Socher, and Steven C H Hoi. Theory-inspired path-regularized differential network architecture search. In Proceedings of the 34th International Conference on Neural Information Processing Systems, number Article 695 in NIPS'20, pages 8296–8307, Red Hook, NY, USA, December 2020. Curran Associates Inc.

[19] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Neural predictor based quantum architecture search. Machine Learning: Science and Technology, 2(4):045027, Oct 2021.

[20] En-Jui Kuo, Yao-Lung L. Fang, and Samuel Yen-Chi Chen. Quantum architecture search via deep reinforcement learning, 2021.

[21] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4095–4104. PMLR, 10–15 Jul 2018.

[22] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. Adv. Neural Inf. Process. Syst., 12, 1999.

[23] Zhimin He, Chuangtao Chen, Lvzhou Li, Shenggen Zheng, and Haozhen Situ. Quantum architecture search with meta-learning. Adv. Quantum Technol., page 2100134, June 2022.

[24] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. npj Quantum Information, 8(1):1–8, May 2022.

[25] Kehuan Linghu, Yang Qian, Ruixia Wang, Meng-Jun Hu, Zhiyuan Li, Xuegang Li, Huikai Xu, Jingning Zhang, Teng Ma, Peng Zhao, Dong E Liu, Min-Hsiu Hsieh, Xingyao Wu, Yuxuan Du, Dacheng Tao, Yirong Jin, and Haifeng Yu. Quantum circuit architecture search on a superconducting processor. January 2022.

[26] Fan-Xu Meng, Ze-Tong Li, Xu-Tao Yu, and Zai-Chen Zhang. Quantum circuit architecture optimization for variational quantum eigensolver via monto carlo tree search. IEEE Transactions on Quantum Engineering, 2:1–10, 2021.

[27] Santiago Ontañón. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE'13, page 58–64. AAAI Press, 2013.

[28] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[29] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. Nat., 550(7676):354–359, 2017.

[30] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE'08, page 216–217. AAAI Press, 2008.

[31] Tristan Cazenave. Nested monte-carlo search. In Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09, page 456–461, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[32] Peter Auer. Using confidence bounds for exploitation-exploration tradeoffs. J. Mach. Learn. Res., 3:397–422, Mar 2003.