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

PEI-YONG WANG[1], MUHAMMAD USMAN[2,3], UDAYA PARAMPALLI[1] (Seniro Member, IEEE), LLOYD C. L. HOLLENBERG[2] and CASEY R. MYERS[1,4]

[1]School of Computing and Information Systems, Faculty of Engineering and Information Technology, The University of Melbourne, Melbourne VIC 3010, Australia
[2]School of Physics, The University of Melbourne, Parkville, VIC 3010, Australia
[3]Data61, CSIRO, Clayton, Victoria, Australia
[4]Silicon Quantum Computing Pty Ltd., Level 2, Newton Building, UNSW Sydney, Kensington, NSW 2052, Australia

Corresponding author: Pei-Yong Wang (email: peiyongw@student.unimelb.edu.au).

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 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 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

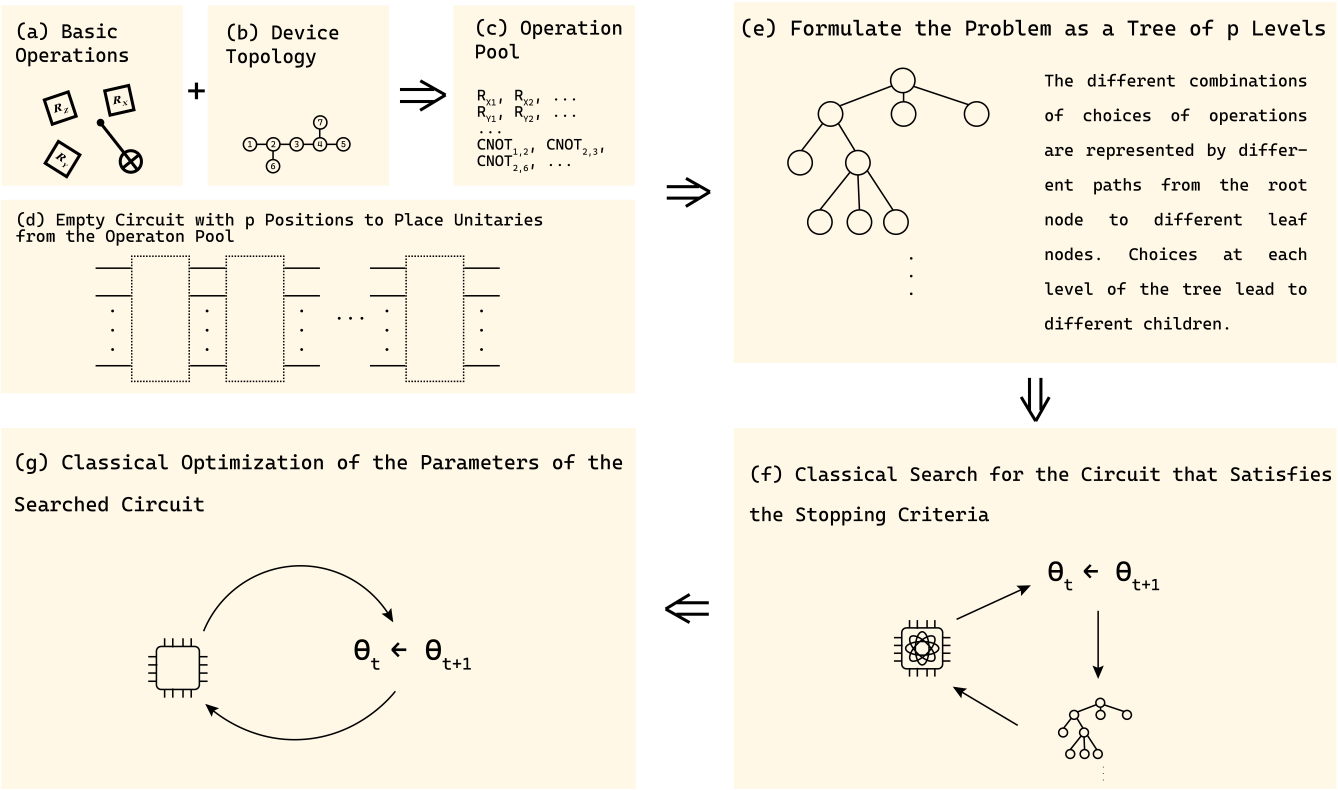


FIGURE 1: An overview of the algorithmic framework proposed in this paper. The operation pool (c) is obtained by tailoring the basic operations (a) with respect to the device topology (b). After that, we formulate the combinations of different choices of operations at different layer position in the 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..

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.

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).

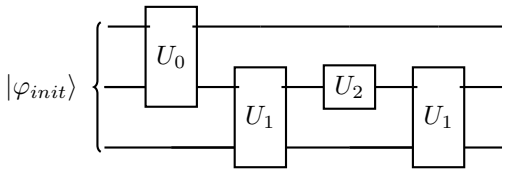


FIGURE 2: An example of the circuit corresponding to the series of unitaries applied to $|\varphi_{init}\rangle$ in Eqn.3.

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 ith layer can be referred as k_i. For example, in the quantum circuit above, we have $k_2 = 1$.

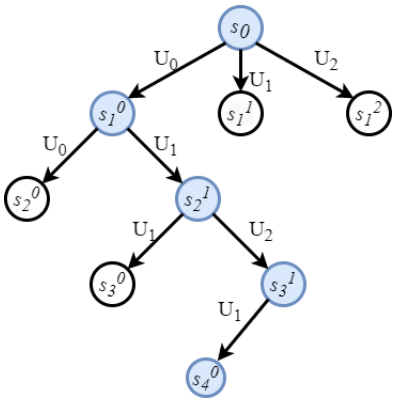


FIGURE 3: The tree representation (along the arc with blue-shaded circles) of the unitary described in Eqns. 3 and 4 as well as Fig. 2. The circle with s_0 is the root of the tree, which represents an empty circuit. Other circles with s_i^j in it denote the jth node at the ith level of the tree. i can 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 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 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.

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.

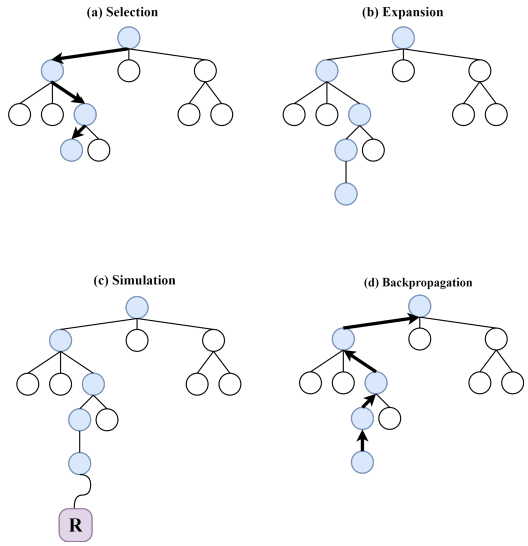


FIGURE 4: Four stages of Monte Carlo tree search. From left to right, up to down: Selection: Go down from the root node to a non fully expanded leaf node; Expansion: Expand the selected node by taking an action; Simulation: Simulate the game, which in our case is the quantum circuit, to obtain reward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

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.

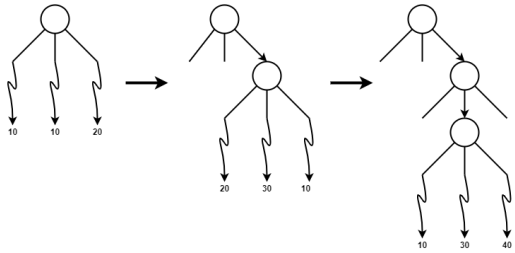


FIGURE 5: Nested Monte Carlo tree search. Left: The root node has three possible actions, which in this case are unselected initially. We perform MCTS on all three children nodes (generated by the three possible actions) to update their reward information. After one iteration of MCTS with each child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

III. NUMERICAL EXPERIMENTS AND RESULTS

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 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:

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

The corresponding encoding circuit is shown in Fig.6.

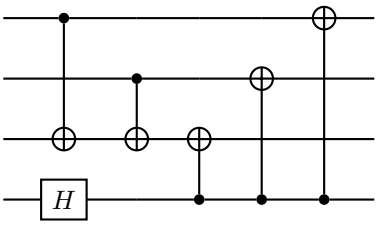


FIGURE 6: Encoding circuit of the [[4,2,2]] code [33] to detect X- and Z-errors. It needs 4 physical qubits for 2 logical qubits and has a code distance 2. By our settings, the number of layers equals to the number of operations in the circuit. In this figure, the number of layers is 6.

Quantum error detection and correction is 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 non-parametric 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 6 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 Section 4.3 of [33] (also shown in Fig. 6) when input states taken from the set of Pauli operator eigenstates and the magic state $|T\rangle$ are used:

$$\mathcal{S} = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+i\rangle, |-i\rangle, |T\rangle\} \tag{12}$$

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

The input states (initialised on all four qubits) are

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

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 $U_{Searched\ [[4,2,2]]}$, which is a function of the structure $\mathcal{P}_{Searched\ [[4,2,2]]}$. The loss and reward function can then be expressed as:

$$L_{[[4,2,2]]} = 1 - \frac{1}{|\mathcal{I}_{[[4,2,2]]}|} \sum_{|\psi_i\rangle \in \mathcal{I}_{[[4,2,2]]}} \langle\psi_i| U^\dagger_{Searched\ [[4,2,2]]} O_{[[4,2,2]]}(|\psi_i\rangle) U_{Searched\ [[4,2,2]]} |\psi_i\rangle \tag{14}$$

$$R_{[[4,2,2]]} = 1 - L_{[[4,2,2]]} \tag{15}$$

where

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

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 numbers of iterations (Fig. 7), although the actual circuit are different from each other, as shown in Fig. 8. The search process that gave the circuit in Fig.8a met the early-stopping criteria in four iterations, and the search process that gave the circuit in Fig.8b met the early-stopping criteria in eight iterations, as shown in Fig. 7.

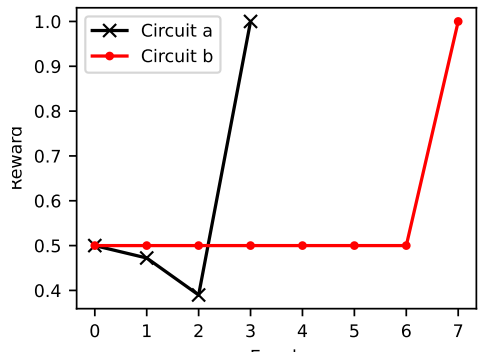


FIGURE 7: Rewards when searching for encoding circuits of the [[4,2,2]] code. We can see that in both cases the algorithm was able find the encoding circuit that generated the required code words in just a few iterations. 'Circuit a' refers to the search rewards for the circuit in Fig.8a and 'Circuit b' refers to the search rewards for the circuit in Fig.8b.

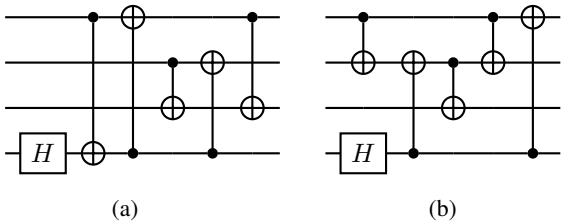


FIGURE 8: Two different encoding circuits of the [[4,2,2]] code produced by the search algorithm.

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 like 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

$$A\,|x\rangle \propto |b\rangle \tag{17}$$

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

$$A = \sum_l c_l A_l \tag{18}$$

where A_l are unitaries, and $|b\rangle = H^{\otimes n}\,|\mathbf{0}\rangle$. We will also adopt the local cost function C_L described in [35]:

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

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:

$$A = \zeta I + JX_1 + JX_2 + \eta Z_3 Z_4 \tag{20}$$

$$|b\rangle = H^{\otimes 4}\,|0\rangle \tag{21}$$

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

$$\mathcal{R} = e^{-10C_L} - \lambda \tag{22}$$

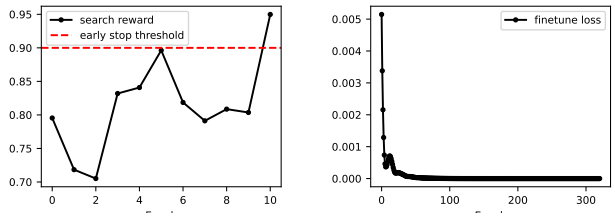
where λ is a penalty term depending on the number of Placeholder gates in the circuit. The operation pool consists of CNOT gates between neighbouring two qubits as well as the first and fourth qubits, the Placeholder and the single qubit rotation gate Rot [28]:

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

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

2) Results

The search rewards as well as fine-tune losses are shown in Fig 9. We can see that the search algorithm can produce a circuit with high reward (exceeds the threshold) quickly and the loss of the optimised parameters can reach close to 0. Although facing a large search space, our algorithm can still find a circuit (shown in Fig 10) that minimises the loss function (Fig 9b) and leads us to results close to the classical solution. A comparison of the results obtained by directly solving the linear equation $Ax = b$ and the results obtained by sampling the state $|x\rangle$ produced by the searched circuit is shown in Fig 11.



(a) Search rewards for VQLS. The change of rewards with respect to the iterations is shown. 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}.

(b) Fine-tune loss for the VQLS circuit.After the searched stopped at iteration 10 as shown in Fig 9a, the structure of the circuit is left unchanged and its parameters are optimised to achieve smaller losses. The final loss of the optimised parameters is very close to 0.

FIGURE 9: The search rewards and fine-tune loss for VQLS experiment.

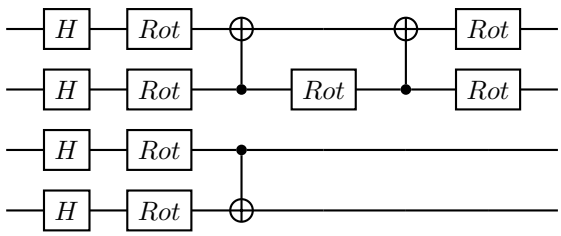


FIGURE 10: Circuit searched for the VQLS problem. $Rot(\phi,\theta,\omega) = R_Z(\omega)R_Y(\theta)R_Z(\phi)$. The four Hadamard gates at the beginning of the circuit are to put everything in an equal superposition, and not included when constructing the search tree, i.e. the composed circuits will always 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.

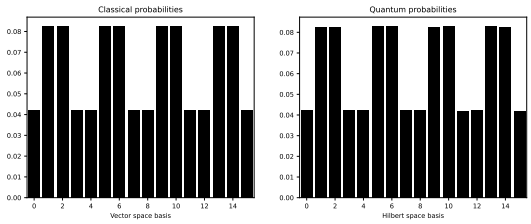


FIGURE 11: Comparison between classical probabilities, which obtained from solving the matrix equation with the classical method, i.e. $x = A^{-1}b$, of the normalised solution vector $\frac{x}{||x||}$ for $Ax = b$ (left), and the probabilities obtained by sampling the state $|x\rangle$ produced by the trained circuit in Fig 10 (right). The number of shots for measurement is 10^6. We can see that the quantum results is very close to the classically obtained ones, showing that our algorithm can be indeed applied to finding variational ansätz for VQLS problems.

C. SEARCH FOR QUANTUM CHEMISTRY ANSÄTZE

Recently, there has been a lot of progress made on finding the ground state energy of a molecule on a quantum computer with variational circuit, both on theoretical [38]–[40] and experimental [1], [41]–[45] front. Normally, when designing the ansätz for the ground energy problem either a physically plausible or a hardware efficient ansätz needs to be found. However, our algorithm provides an approach which can minimise the effort needed to carefully choose an ansätz 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]:

$$E_0 \leq \frac{\langle\tilde{0}|H|\tilde{0}\rangle}{\langle\tilde{0} \mid \tilde{0}\rangle} \tag{24}$$

where H is the system Hamiltonian, $|\tilde{0}\rangle$ is the "trail ket" [46], or ansätz, trying to mimic the real wave function at ground state with energy E_0, which is 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 θ:

$$\left|\tilde{0}\right\rangle = |\varphi(\theta)\rangle = U(\theta)\left|0^{\otimes n}\right\rangle \tag{25}$$

Given an ansätz, the goal of optimisation is to find a set of parameters θ that minimises the right hand side of Eqn 24. However, in our research, the form of the trail wave function will no longer be fixed. We will not only vary the parameters, but also the circuit structure that represent the ansätz.

1) Experiment settings

a: Search an ansätz for finding the ground energy of H_2:

In this experiment, we adopted the 4-qubit Hamiltonian $H_{hydrogen}$ for the hydrogen molecule H_2 generated by the Pennylane-QChem [34] package, when the coordinates of the two hydrogen atoms are $(0, 0, -0.66140414)$ and $(0, 0, 0.66140414)$, respectively, in atom units. The goal of this experiment is to find an ansätz that can produce similar states as the four-qubit Givens rotation for single and double excitation. The unitary operator [1] that performs single excitation on a subspace spanned by $\{|01\rangle, |10\rangle\}$ can be written as

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

And the transformation of the double excitation on the subspace spanned by $\{|1100\rangle, |0011\rangle\}$ is[2] :

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

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

$$L_{\mathrm{H}_2} = \langle\psi_0|U^{\dagger}_{SearchedAnsatz} H_{Hydrogen} U_{SearchedAnsatz}|\psi_0\rangle \tag{28}$$

$$R_{\mathrm{H}_2} = -L_{\mathrm{H}_2} \tag{29}$$

The operation pool consists of Placeholder gates, Rot gates and CNOT gates with a linear entanglement topology (nearest neighbour interactions). The maximum number of layers is 30, with maximum number of CNOT gates $30/2 = 15$, and no penalty term for the number of Placeholder gates:

$$\mathcal{R}_{\mathrm{H}_2, Pool\ 1} = R_{\mathrm{H}_2} \tag{30}$$

Such settings of operation pool and 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 ansätz for finding the ground energy of LiH

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

$$L_{\mathrm{LiH}} = \langle\psi_0|U^{\dagger}_{SearchedAnsatz} H_{\mathrm{LiH}} U_{SearchedAnsatz}|\psi_0\rangle \tag{31}$$

$$R_{\mathrm{LiH}} = -L_{\mathrm{LiH}} \tag{32}$$

and the initial state is also the vacuum state $|\psi_0\rangle = |0\rangle^{\otimes 10}$. The Hamiltonian is obtained at bond length 2.969280527 Bohr, or 1.5712755873606 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 neighbouring qubits on a line topology.

[1] see https://pennylane.readthedocs.io/en/latest/code/api/pennylane.SingleExcitation.html
[2] see https://pennylane.readthedocs.io/en/latest/code/api/pennylane.DoubleExcitation.html

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 ansätz for finding the ground energy of H_2O

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

$$L_{\mathrm{H_2O}} = \langle\psi_0|U^{\dagger}_{SearchedAnsatz} H_{\mathrm{H_2O}} U_{SearchedAnsatz}|\psi_0\rangle \tag{33}$$

$$R_{\mathrm{H_2O}} = -L_{\mathrm{H_2O}} \tag{34}$$

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:

$$\begin{aligned} &\mathrm{H} : (0., 0., 0.); \\ &\mathrm{O} : (1.63234543, 0.86417176, 0); \\ &\mathrm{H} : (3.36087791, 0., 0.) \end{aligned} \tag{35}$$

Units are in Angstrom. Active electrons is set to 4 and active orbitals is set to 4. The size of the operation pool $c = |\mathcal{C}| = 30$, including Rot gates, Placeholder and CNOT gates operating on neighbouring 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

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 trade-offs. J. Mach. Learn. Res., 3:397–422, Mar 2003.
[33] Joschka Roffe. Quantum error correction: an introductory guide. Contemporary Physics, 60(3):226–245, 2019.
[34] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2020.
[35] Carlos Bravo-Prieto, Ryan LaRose, M. Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J. Coles. Variational quantum linear solver, 2019.
[36] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Phys. Rev. Lett., 103:150502, Oct 2009.
[37] Variational quantum linear solver — PennyLane. https://pennylane.ai/qml/demos/tutorial_vqls.html. Accessed: 2022-4-24.
[38] Ying Li and Simon C Benjamin. Efficient variational quantum simulator incorporating active error minimization. Physical Review X, 7(2):021050, 2017.
[39] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. New Journal of Physics, 18(2):023023, 2016.
[40] Dave Wecker, Matthew B Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. Physical Review A, 92(4):042303, 2015.
[41] Peter JJ O'Malley, Ryan Babbush, Ian D Kivlichan, Jonathan Romero, Jarrod R McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, et al. Scalable quantum simulation of molecular energies. Physical Review X, 6(3):031007, 2016.
[42] James Colless, Vinay Ramasesh, Dar Dahlen, Machiel Blok, Jarrod McClean, Jonathan Carter, Wibe A de Jong, and Irfan Siddiqi. Implementing a variational quantum eigensolver using superconducting qubits. In Quantum Information and Measurement, pages QF6A–2. Optical Society of America, 2017.
[43] 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, 2017.
[44] James I Colless, Vinay V Ramasesh, Dar Dahlen, Machiel S Blok, Mollie E Kimchi-Schwartz, Jarrod R McClean, Jonathan Carter, Wibe A de Jong, and Irfan Siddiqi. Computation of molecular spectra on a quantum processor with an error-resilient algorithm. Physical Review X, 8(1):011021, 2018.
[45] Eugene F Dumitrescu, Alex J McCaskey, Gaute Hagen, Gustav R Jansen, Titus D Morris, T Papenbrock, Raphael C Pooser, David Jarvis Dean, and Pavel Lougovski. Cloud quantum computing of an atomic nucleus. Physical review letters, 120(21):210501, 2018.
[46] J. J. Sakurai and Jim Napolitano. Modern Quantum Mechanics. Cambridge University Press, 2 edition, 2017.
[47] PennyLane dev team. A brief overview of vqe, Jul 2021.