Technical Reflection & Description

Jialin Chen，Ke Xu，Zhiqiang Cai, Di Wu, Yunjin Jiao
Team QuantumLoop
Big Data Institute, Fudan University, Shanghai, China

1. Overview

The algorithm we utilized to search Variational Quantum Circuit (VQC) structure is based on Monte Carlo Tree Search (MCTS), which we term QASC (**Q**uantum **A**rchitecture **S**earch for **C**hemistry). The core idea of QASC is to build a binary tree to gradually distinguish good structures from bad ones for a specific task. After that, we utilize Qiskit's built-in functions to do ciruit-transpiling and Pauli-grouping. In the stage of error mitigation, a modified extrapolation method is proposed to infer the final result.

The rest of reports is organized as follows. In section 2, we describe the workflow of QASC. Then, we provide our insights of constructing hardware efficient ansatze and corresponding encodes for circuit structures. At the end of the section 3, we display the results of QASC algorithm that verify the efficacy of the search method. Post-optimization techniques including Pauli grouping, transpiling optimization and error mitigation are introduced in section 4, 5 and 6 respectively. In the last two sections, we display our experimental results and conclude our work with a summary and outlook.

2. Workflow of QASC

QASC is an iterative algorithm, with each iteration composed of a learning phase and a search phase. The learning phase builds a partition tree for sampling in the search phase, while the search phase provides the learning phase with trained samples. An overview of an iteration of QASC is illustrated in Fig. 1.

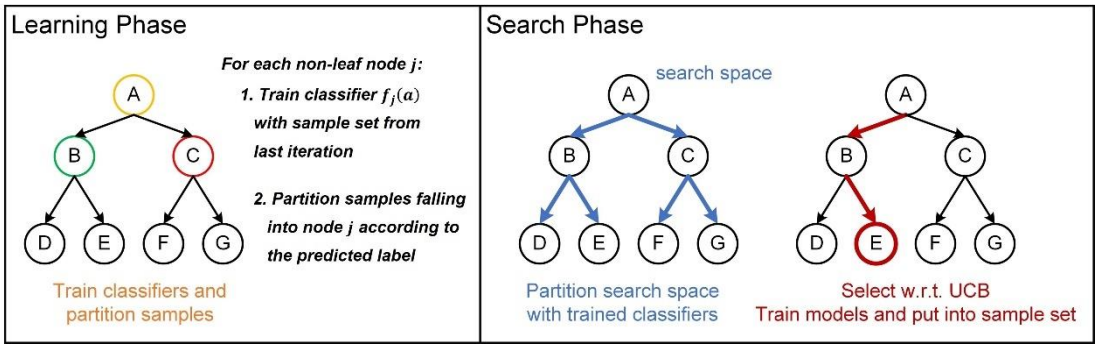

Fig. 1 An overview of an iteration of QASC

The goal of the ***learning phase*** is to recursively partition the entire search space such that we can group the architectures with similar performance together in the same region. To this end, a full binary tree is used to initialize the algorithm where the root represents the entire search space. For each node of the tree, we learn a classifier with the samples falling into the node represented region, which is then used to split the region into two disjoint subregions containing the half of good models

and bad ones respectively. If we partition the architecture space recursively from the root of the tree with height n, we can finally get 2^{n-1} subregions represented by the leaves, where the leftmost leaf contains the best group of models and the further to the right, the models perform worse.

In the ***search phase***, we sample architectures, evaluate them, and use them to refine the boundaries in the learning phase of next iteration with the UCB score, a measure of both the value possessed and the visits of a node. The MCTS procedure will keep running until an optimal architecture with good enough performance is found for the task.

Note that before the search program, a prerequisite is that we need to determine our search space and structure encoding.

3. Hardware efficient ansatz and structure encoding

As is known to all, the Hilbert space of 12 qubits is very large. As a result, to improve the efficiency of finding a high-performance model, the following insights observed by us is taken account when constructing ansatz:

- 12 trainable parameters are sufficient for the task. Circuits with less than 11 trainable parameters will lead to a significant deterioration in performance.
- The first qubit plays an important role in the accuracy of the energy estimation, thus we always place quantum gates on the first qubit.
- Considering that all quantum gates in the circuit will finally be decomposed into a set of basic operations: SX, RZ and CNOT, we employ two building blocks consisting of these basic gates to construct our circuits. As illustrated in Fig. 2, the single-qubit block is a concatenation of two SX and a RZ that operate on the same qubit. The two-qubit block is a CNOT gate followed by a RZ rotation on the target qubit. Using the basic gates to construct our circuit can ensure hardware efficiency, since other quantum gates still need decompositions and the complexity of final circuit is difficult to control.

Here is an example to illustrate the construction of circuits that make up our search space and their corresponding encoding, as depicted in Fig. 2. According to the insight, we assume that each qubit can be applied by one single-qubit block and one two-qubit block at most once, which leads to a list of 24 variables. The column “Pauli Rotations” of table in Fig.2 reflects the positional encoding of the single-qubit blocks, where the symbol $'n'$ means that the block in the corresponding qubit will be discarded and the symbol $'0'$ otherwise. In the last column, the symbol $'n'$ has the same meaning, while other symbols indicate the target input of the block. Note that We do not exclude the case where the target input and control input are identical. When this happen, we simply discard the CNOT gate in the block and retain the PauliZ rotation. Under the circuit hypotheses described above, there are 83M circuits for the entire search space.

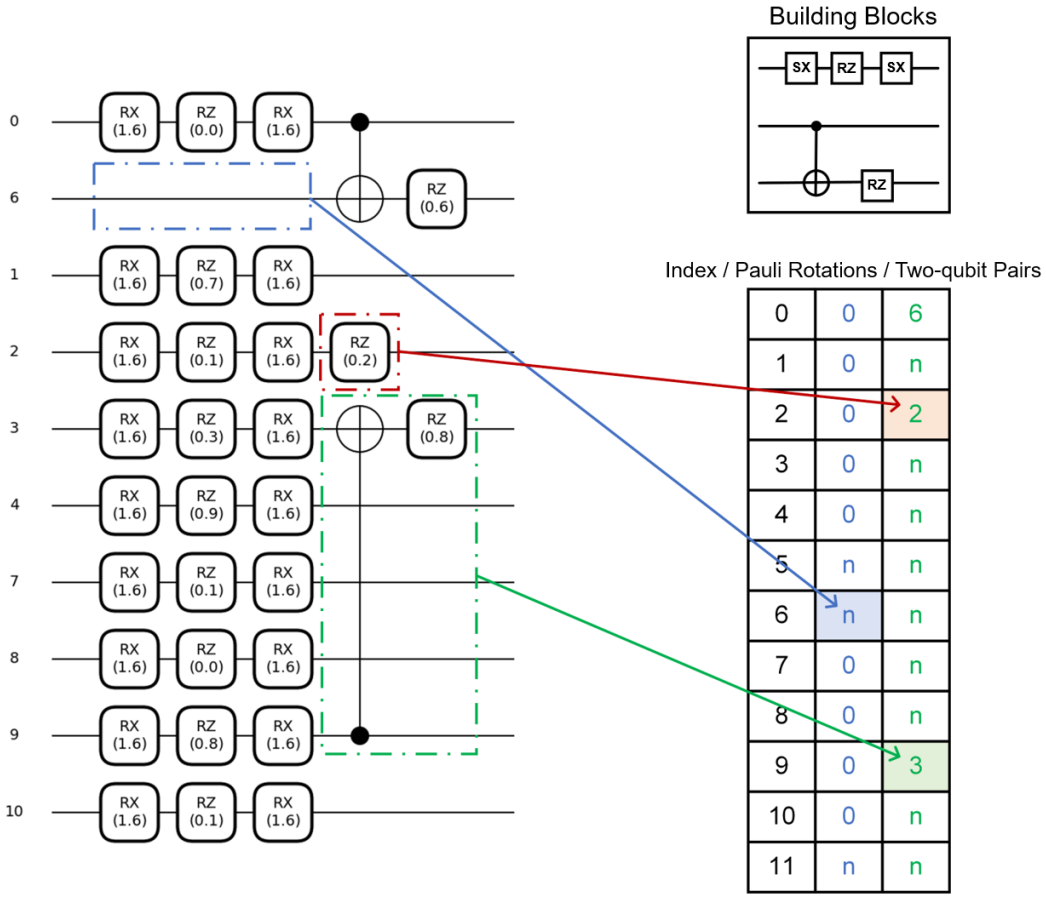

Fig. 2 Circuit hypotheses and structure encoding

Due to the unique structure encoding, QASC is efficient to partition the search space based on model performance. We run the procedure for 20 iterations. After each iteration, 100 circuits are randomly sampled in every leaf node to evaluate the performance of the tree. The average results of the last 10 iterations are illustrated in Fig. 3. We can observe from the figure that the circuits in the search space have been categorized according to their performance on the task. Subsequently, we randomly select a number of circuits in the leftmost leaves, among which the best-performing circuit is determined for our final results.

The code of QASC is available at https://github.com/zqcai19/chemistry-OH

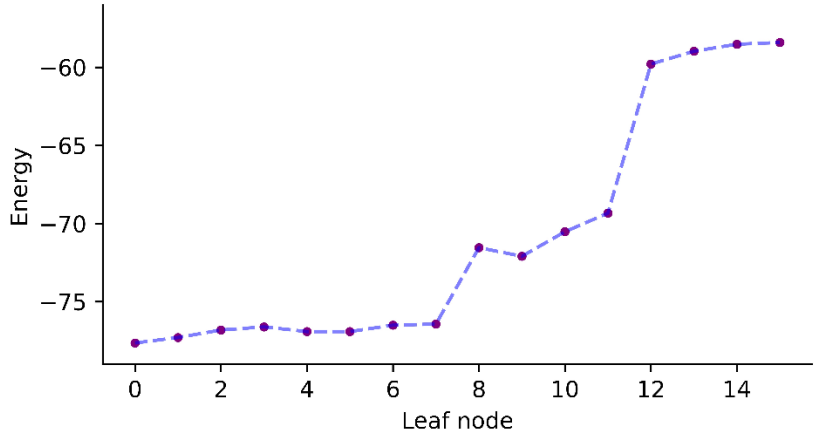

Fig. 3 Results of QASC algorithm

4. Pauli Grouping

When measuring the expectation value of a Hamiltonian, it is desirable to group the list of Pauli strings into commuting subsets in order to minimize the number of measurements. As shown in Fig. 4, for the hydroxyl cation's Hamiltonian having 631 Pauli strings, we use Qiskit's built-in function, *group_commuting*, to shrink the list of Pauli strings down to 39, and reduce the total number of shots to $39 \times shots$.

```
from qiskit.quantum_info import SparsePauliOp

observable = SparsePauliOp(labels, coeffs)

# Perform Pauli grouping using Qiskit built-in function: group_commuting()
grouped_observable = observable.group_commuting()     # grouping type: fully commuting
print("The length of original Pauli strings: ", len(observable))
print("The length of grouped Pauli strings: ", len(grouped_observable))
```

```
The length of original Pauli strings:  631
The length of grouped Pauli strings:  39
```

Fig. 4 Results of Pauli-grouping

5. Transpiling Optimization

Since we limit the complexity of searched circuits to have at most two CNOT gates, our circuit is simple enough to use Qiskit's default transpilation option which carries out the heaviest optimization level (3). As shown in Fig. 5, the backend model is *FakeMontreal* and the required algorithm seeds are passed into the transpiler via the *evaluate* function in our code.

```
from qiskit import *
from qiskit.providers.fake_provider import *

def evaluate(noise_model, circuit, seeds, shot, zne=None):
    system_model = FakeMontreal()
    transpiled_circuit = transpile(circuit, backend=system_model, seed_transpiler=seeds)
```

Fig. 5 Transpiling optimization method

6. Error mitigation: Zero Noise Extrapolation

Zero-noise extrapolation (ZNE) is an error mitigation technique in which the ideal expectation value is inferred by extrapolating the measured results at different levels to the zero-noise limit. Below lists the kind of noise scaling method and extrapolation method we adopted.

- **Noise scaling method**. Unitary local folding is used to amplify the noise by performing $U \rightarrow UU^{+}U$ for each gate locally. We set scaling factors (1, 3, 5), (1, 3, 5, 7, 9) and (3, 5, 7) for three circuits respectively. It should be noticed that the scaling factor (3, 5, 7) is proved better than (1, 3, 5, 7) for *circuit_3*. We guess the reason is that when the scale factor is 1, the circuit_3 remains unchanged and its short duration (only 320) causes the strong qubit-environment interaction, which is negative for the performance of the following extrapolation.

- **Extrapolation method**. Two extrapolation methods are used: Richadson extrapolation (Rich) and a modified linear extrapolation proposed by us, called average slope (AvS)

method. The idea of AvS is simple, i.e. to average the slopes of all pairs between first points x_1 and others. Generally, if there are m points, we define the average slop l is as follows

$$l = \frac{1}{m}\sum_{i=2}^{m} \frac{x_1 - x_i}{i-1}$$

then the zero-noise limit is calculated by $x_0 = x_1 + l$.

As shown in Fig. 6, both extrapolation methods behave similarly and have its own advantages in some cases. In practice, we choose **AvS** for noise model 1 and 3 and **Rich** for model 2 to obtain the best results.

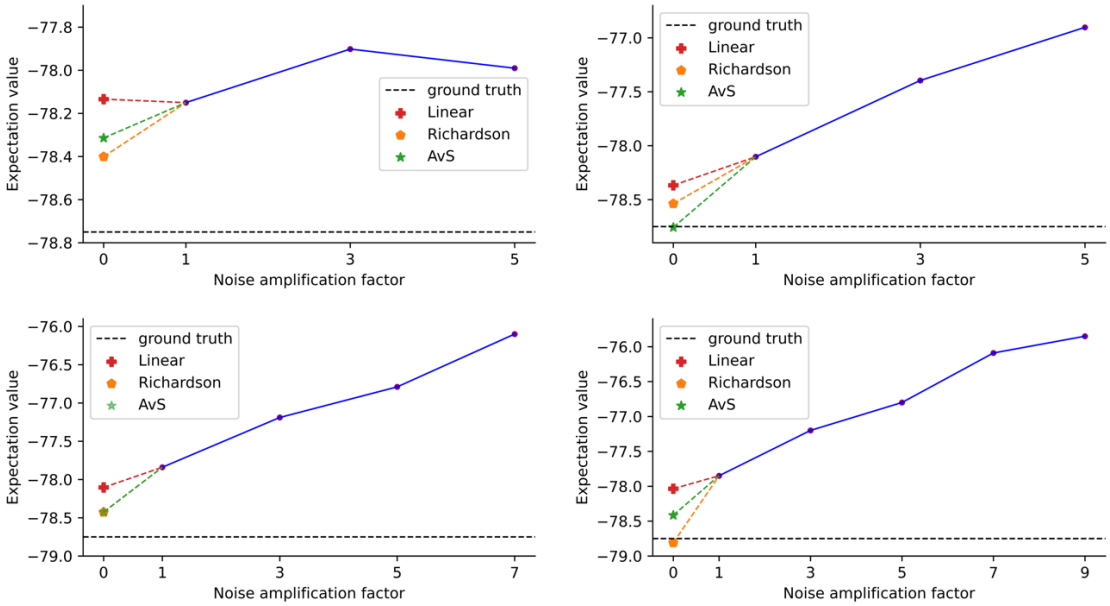

Fig. 6 Comparisons of noiseless results inferred by three extrapolation methods

7. Experimental Results

By deploying the above search strategy and corresponding post-optimization techniques, three circuits in Fig. 7, which term as *circuit_1*, *ciruit_2* and *circuit_3* are found for three noise model respectively. The number of shots in three noise experiments are 6000, 6000 and 1000 respectively and the *noise_factor* of ZNE are (1,3,5), (1,3,5,7,9) and (3,5,7).

From experimental results in Table 1, it is surprised that at most two CNOT gates are needed and even tensor product states also work well. Moreover, it can be found that two qubits q_5 and q_{11} are least important in this task from the Fig.7, so their state is always $|00\rangle$.

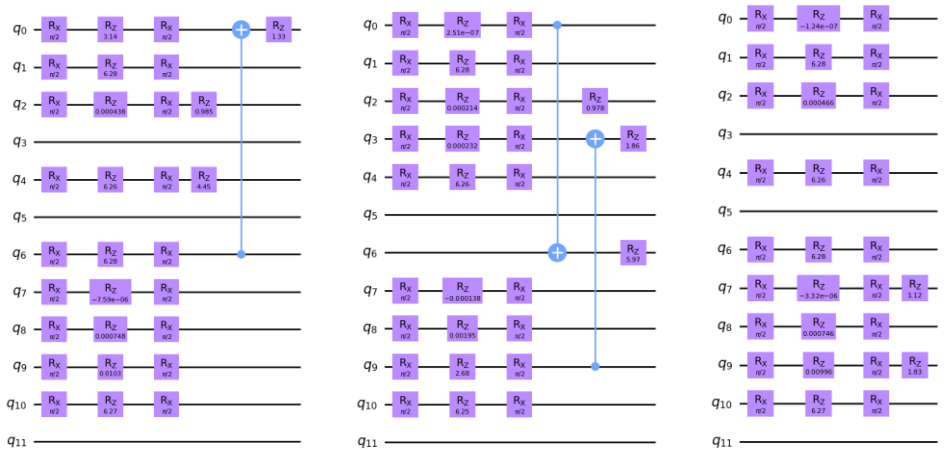

Fig. 7 Diagram of three circuits

Table 1. Experiment results for the three provided noise models

seed #	Noise Model 1: fakekolkata	Noise Model 2: fakemontreal	Noise Model 3: fakecairo
20	-78.75	-78.88	-78.71
21	-78.75	-78.84	-78.71
30	-78.76	-78.81	-78.71
33	-78.75	-78.81	-78.72
36	-78.76	-78.81	-78.72
42	-78.76	-77.96	-78.71
43	-78.76	-78.79	-78.71
55	-78.76	-78.72	-78.71
67	-78.77	-78.80	-78.74
170	-78.79	-77.83	-78.59
Average	**-78.76**	**-78.63**	**-78.70**
Accuracy	**99.99%**	**99.84%**	**99.93%**
Duration	**2,080**	**2,105.6**	**320**

It is interesting to see the ideal expectation values of these circuits, which we list below. These values are obtained by *qiskit-aer* simulator without any noise. We can see that by using appropriate error mitigation method, the performance under noise condition will be even better.

Table 2. Ideal $\langle H \rangle$ of three circuits

Circuit_1	Ciruit_2	Circuit_3
-78.67	**-78.66**	**-78.69**

8. Conclusions

In the contemporary NISQ era, it is critical for us to deploy well-tailored search strategy and post-optimization techniques to overcome the detrimental effects of the noise. With QASC and certain extrapolation method, like AvS, we are able to find much simple quantum circuits, or even tensor product states without any two-qubit gates to achieve the promising results. As the experimental results shown, the circuits continuously achieve accuracies over 99%, which are even better than values under noiseless situation. Hence, instead of reducing noise only, we can also utilize noise processes to boost the performance, since noise is the nature of quantum world.