SA-DQAS: Self-attention Enhanced Differentiable Quantum Architecture Search

Yize Sun [1 2 3] **Jiarui Liu** [2] **Zixin Wu** [2] **Zifeng Ding** [2 3] **Yunpu Ma** [1 2 3] **Thomas Seidl** [2] **Volker Tresp** [1 2 3]

Abstract

We introduce SA-DQAS in this paper, a novel framework that enhances the gradient-based Differentiable Quantum Architecture Search (DQAS) with a self-attention mechanism, aimed at optimizing circuit design for Quantum Machine Learning (QML) challenges. Analogous to a sequence of words in a sentence, a quantum circuit can be viewed as a sequence of placeholders containing quantum gates. Unlike DQAS, each placeholder is independent, while the self-attention mechanism in SA-DQAS helps to capture relation and dependency information among each operation candidate placed on placeholders in a circuit. To evaluate and verify, we conduct experiments on job-shop scheduling problems (JSSP), Max-cut problems, quantum chemistry and quantum fidelity. Incorporating self-attention improves the stability and performance of the resulting quantum circuits and refines their structural design with higher noise resilience and fidelity. Our research demonstrates the first successful integration of self-attention with DQAS.

1. Introduction

In recent years, quantum computing (QC) has developed rapidly and has achieved remarkable progress in different areas, such as image classification, circuit architecture search, quantum reinforcement learning, knowledge graph embedding, and approximate optimization problem [1]–[5]. However, the performance of the quantum algorithms is often different from expected on noisy devices. Variational quantum algorithms (VQAs) are considered the leading strategy in the NISQ era [6].

[1]Munich Center for Machine Learning (MCML), Munich, Germany [2]Institute of Informatik, LMU, Munich, Germany [3]Siemens AG, Munich, Germany. Correspondence to: Yunpu Ma <cognitive.yunpu@gmail.com>.

Published at the 2nd Differentiable Almost Everything Workshop at the 41st International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

Various quantum architecture search (QAS) algorithms can automatically search for the near-optimal quantum circuit architecture [7]–[15]. Differentiable quantum architecture search (DQAS) [9] is a gradient-based QAS algorithm. It designs circuits by replacing placeholders with operation candidates. Instead of evaluating all possible architecture, DQAS only samples a batch of candidates for updating architecture parameter α and circuit parameters θ. The loss $\mathcal{L}$ and the local loss L are given as:

$$\mathcal{L} = \sum_{k \in P(k,\alpha)} \frac{P(k,\alpha)}{\sum_{k' \in P(k,\alpha)} P(k',\alpha)} L(k) \quad , \tag{1}$$

where

$$P(k,\alpha) = \prod_{i=1}^{p} p(k_i, \alpha_i) \quad , \tag{2}$$

and k determines a circuit structure in a probabilistic model.

Since DQAS assumes that each placeholder is independent of other placeholders, each operation candidate has little overall view of operations in other placeholders. In order to tackle this issue, we choose the architecture parameter α as a tokenized "sentence" and extend the DQAS by adding a transformer encoder [16], which compresses architecture parameter and positional information into a low-dimension vector to provide a global view of the operation candidates in different placeholders. Unlike prior methods [9] that the selection of operation for each placeholder has little information regarding operation candidates in other placeholders, we present a DQAS method enhanced with the self-attention mechanism SA-DQAS to let the operation candidates have a global view.

We summarise our contributions as follows:

1. We present a high-performing QAS algorithm SA-DQAS by extending the DQAS with an attention module. A transformer encoder can transform architecture parameters by handling the fitness and positional information of operation candidates.

2. We conduct sufficient experiments for comparing our new method with the original DQAS in different tasks. The SA-DQAS provides outperforming results and can

find high-performing circuit architectures efficiently on a quantum simulator.

3. We test SA-DQAS in a noisy environment. By considering idle errors, we show that SA-DQAS can still find noise-resilient circuit architecture by keeping entanglements.

2. Methodology

Our work is based on DQAS, which uses a gradient-based method to construct a quantum circuit by replacing these placeholders with one possible candidate operation from the operation pool. We denote the operation pool as $\mathcal{O}$. A circuit is a sequence of p placeholders described as $U = \prod_{i=0}^{p} u_i(\theta_i)$, where u_i and $o_i \in \mathcal{O}$ stand for unitary placeholder and operation candidate, and θ_i are the parameters of trainable gates. The architecture parameter matrix is $(\alpha_{ij}) \in \mathbb{R}^{p\times l}$ with $l = |\mathcal{O}|$. The probability is calculated by Eq 2 . This setting makes the search process continue and focuses on operations in one placeholder but ignores the relationship among operation candidates placed on different placeholders.

Encoder $F1$**&**$F2$ As shown in Figure 1, each architecture parameters vector α_i is viewed as a word, and the architecture matrix α is viewed as a sentence. The output of the encoder is the architecture parameters, including relationship features, and is added to the original architecture parameter matrix α:

$$\alpha' = \alpha + \beta F(\alpha), \tag{3}$$

where F is the encoder and β is the scaling coefficient.

From different perspectives, by viewing a placeholder as a token, we propose one encoder with two types of input: F_1 and F_2. Regarding the encoder F_1, we use sinusoidal position embedding α_i^{pos} for N layers-encoder. Each layer consists of two sublayers:

$$\alpha_i^{n_1} = \text{LayerNorm}(\alpha_i^{a}), \tag{4}$$
$$\alpha_i^{\text{a}} = \text{MultiHead}(\alpha_i^{pos}, \alpha_i^{pos}, \alpha_i^{pos}) + \alpha_i^{pos}. \tag{5}$$

The second sub-layer is a position-wise fully connected feed-forward network:

$$\alpha_i^{n_2} = \text{LayerNorm}(\alpha_i^{a_2}), \tag{6}$$
$$\alpha_i^{a_2} = \alpha_i^{n_1} + g(\alpha_i^{n_1}), \tag{7}$$
$$g(\alpha_i^{n_1}) = W_2\text{max}(0, W_1\alpha_i^{n_1} + b_1) + b_2. \tag{8}$$

And α is updated by:

$$\alpha_1' = \alpha + \beta F_1(\alpha^{trans}), \tag{9}$$
$$\alpha^{trans} = \alpha \times \alpha^T \times \alpha. \tag{10}$$

For $F2$, we view the original architecture matrix α as an input for the encoder. The output of this method is $\alpha_2' = \alpha + \beta F_2(\alpha)$. We expect that the self-attention mechanism can give us additional features without strongly reducing the effect of the original architecture parameters. As a result, we select a sufficiently small β as a scaling coefficient.

Loss function The loss function for the whole process of circuit architecture search is described in Eq.1. In order to get a gradually fixed architecture parameter, we select the distance of the output of the encoder between the current training step and one previous training step as our loss metric. For each training step t, the encoder parameters are updated by

$$w_t = w_t - \eta\Delta_w L_{\text{encoder}} \tag{11}$$
$$L_{\text{encoder}} = \max_{1\le i\le p}\max_{1\le j\le l} |F_{t-m} - F_t(\alpha_{ij}^{trans})|, \tag{12}$$

where F_t is the output matrix of F in training step t. We set $m = 1$ in our experiment and $F(\alpha^0)$ is initialized by 0.

3. Experiments

3.1. Experiment settings

Our experiments test our framework based on JSSP and Max-cut problems. We define four different types of operation pools, $\mathcal{O}_1$-$\mathcal{O}_4$. Appendix A describes the operation pool construction rules. The basic settings in the training process and types of gates contained in $\mathcal{O}_1$-$\mathcal{O}_4$ are given in Appendix E.

3.2. Experiments on JSSP

In this section, we conduct experiments with five qubits using SA-DQAS for the simplest JSSP task defined in paper[4] with five qubits. Their manually designed circuit, represented in Figure 2b, is selected as our baseline. In the experiments, all qubits are initialized with the state $|0\rangle$ and applied `rx` gates with a rotation angle of π to form the encoding block. There is only one parameterized block containing four placeholders. For each learning step, we update all placeholders. The energies E are scaled to the range $[0, 1]$:

$$e = \frac{E - E_{\min}}{E_{\max} - E_{\min}} \in [0, 1] \quad , \tag{13}$$

where $E_{\min}$ and $E_{\max}$ are the minimum and maximum energy. When $e \approx 0$, we reach the minimal energy and find the optimal solution.

From Table 1, nearly all circuits designed by SA-DQAS converge faster than those without self-attention. When circuits generated with SA-DQAS contain fewer parameterized gates than those with DQAS. There are still some searches that do not find optimal circuits.

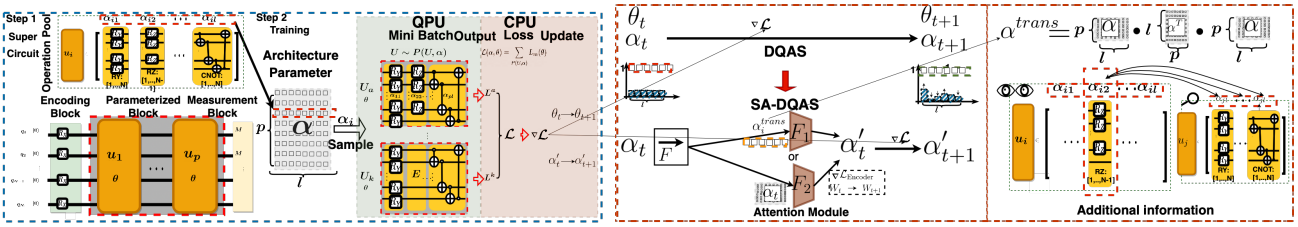


Figure 1: The illustration of the proposed framework SA-DQAS. On the left side is the common part of both algorithms (Step 1 and Step 2). On the right side, we show the differences between both algorithms. SA-DQAS uses the attention modules F_1 or F_2 to improve the architecture parameters α, such that each operation shares positional and fitness information among operation candidates in different placeholders.

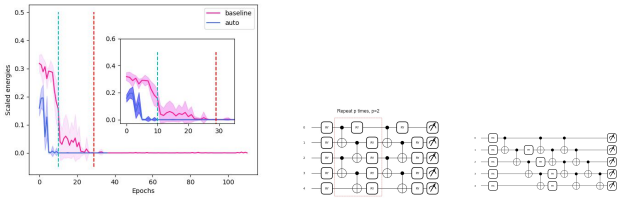
Table 1: Summary of circuits designed with and without self-attention from different operation pools for JSSP

Operation pool	#Gates***		#Param. Gate***		Quantum cost***		ASP**	
Baseline	18		10		8		29	
	Y*	N*	Y	N	Y	N	Y	N
op1-1	15	14	15	14	0	0	25	26
op1-2	13	9	9	9	4	0	18	11
op1-3	13	19	9	14	4	0	20	39
op1-4	14	20	5	15	4	0	13	38
op2-1	13	9	9	9	4	0	13	16
op2-2	15	16	7	12	8	4	13	28
op2-3	16	18	8	18	8	0	13	40
op2-4	15	18	10	5	0	8	19	14
op3-1	15	13	7	9	8	4	18	15
op3-2	15	13	11	13	4	0	17	32
op3-3	16	16	4	12	12	4	10	16
op3-4	13	20	5	20	8	0	14	39
op4-1	7	12	7	12	0	4	33	72
op4-2	12	15	12	15	8	8	60	108
op4-3	16	16	16	16	8	8	89	96
op4-4	16	17	16	17	16	12	88	91

* Y indicates circuits searched by SA-DQAS and N indicates by DQAS

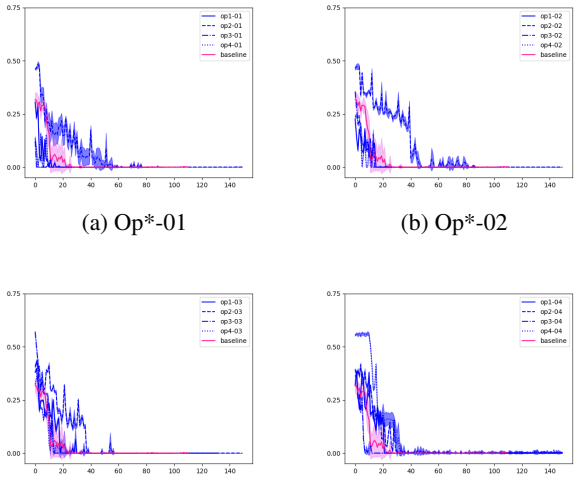
** Average solving point (ASP) refers to the number of iterations required to find the minimum energy without subsequent fluctuations.

*** Columns 1-3 indicate the number of total gates, trainable gates and controlled gates in the parameterized blocks.



(a) Comparison (b) Baseline (c) Circuit$_{auto}$

Figure 2: Evaluation of newly found architectures on the simulator without noise. The results are averaged over 10 trials with the different initial parameters.



(a) Op*-01 (b) Op*-02

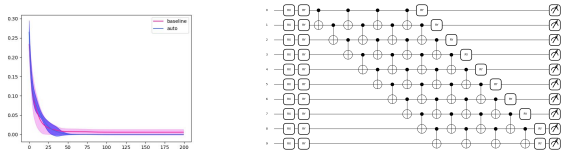
(c) Op*-02 (d) Op*-04

Figure 3: Comparison of circuits generated using operation pools replacing ry with rx. We also generate operation pools by replacing ry with rx gates.

Figure 2 shows the evaluation of newly found architectures on the simulator without noise. The best-performing circuits converge faster than the baseline and have high reliability. More controlled gates indicate a higher building cost of the quantum circuit, while controlled gates provide quantum features. The discovered circuit contains fewer gates than the baseline. The impact of placeholders and parameterized blocks are shown in Appendix B.

In Figure 5, we show the circuit evaluation on the noisy simulator. The designed circuits are noise-resilient and more reliable in most cases, although the learned circuits, including the baseline, do not find the accurate minimum energy in some trials because of the influence of noise.

From Figure 6, we know that random information to DQAS



(a) Performance JSSP-10-qubits (b) Circuit JSSP-10-qubits

Figure 4: Top performing circuit for 10 qubits JSSP problem. The baseline (red) does not converge to the bottom, while the automatically design circuit can within 300 training Epochs.

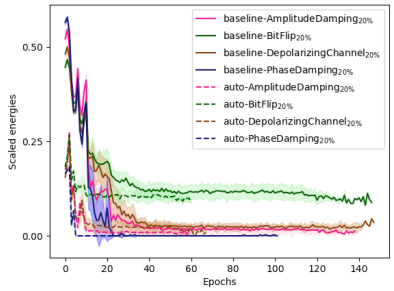


Figure 5: Performance evaluation in noisy environments. We study the resistance of auto-generated circuits to noise. We form noise models by adding 20% noise of different types on each end of the qubit in the generated circuit with the best performance.

Table 2: Summary of circuit architectures designed with and without self-attention from different operation pools for solving different Max-cut problem.

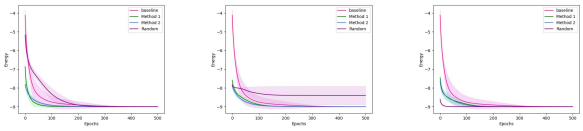
Circuits	#Gates**			#Param. G.			#Con. G.		
	M_1^*	M_2^*	N^*	M_1	M_2	N	M_1	M_2	N
$\text{Circuit}^{***}_{R-\text{op4}-7_4}$	23	22	29	23	22	29	8	16	8
$\text{Circuit}_{L-\text{op3}-3_4}$	23	27	22	15	19	14	8	8	8
$\text{Circuit}_{B-\text{op3}-6_4}$	29	30	23	21	22	23	8	8	0

* M_1^* indicates circuits searched by Method 1, M_2^* by Method 2 and N indicates by DQAS.

** Columns 1-3 indicate the number of total gates, controlled gates, and trainable parameterized gates in the parameterized blocks.

*** The suffix indicates that the circuits are derived from which operation pool and used to solve the Max-cut problem in which graph. R denotes Random graph, L means Ladder and B is Barbell. The suffix op4-4 means the circuits are generated from operation op4-4.

can lead to the instability of the final circuit performance. However, self-attention can help build stable circuit architectures through additional information. There are more comparison results in terms of randomness, Max-Cut and error mitigation problem in Table 2 and Appendix C, and noisy test in Appendix D.



(a) op2-7_4 (b) op3-4_4 (c) op3-6_4

Figure 6: Evaluation of performances of circuits generated by SA-DQAS and DQAS with random information. Method 1 and 2 use the encoder F_1 and F_2, respectively. The line is the average results of 10 trials with different initial parameters. The suffix means four placeholders are used.

Table 3: Experiments on hydrogen. We solve the Hamiltonian of H_2 and compare our method to other algorithms. The value indicates the error level compared with ground state energy.

Algorithm	H_2 (Energy Error)
UCCSD	10^{-11}
quantumDARTS	10^{-6}
SA-DQAS	10^{-4}
DQAS	10^{-4}
QGAS	10^{-2}
Randomness	10^{-2}

3.3. Experiments on quantum chemistry

In Table 3, we compared SA-DQAS with different algorithms in terms of hydrogen. The SA-DQAS can find optimal circuit architecture that outperforms QGAS [7], RS but still has limited performance by comparing with UCCSD [17] and quantumDARTS [2]. SA-DQAS here may only find the local optimum due to a very shallow circuit with only 4 placeholders. More molecules and a large circuit should be considered by comparison in the next step.

4. Conclusion

Our method, SA-DQAS, improves DQAS by using the self-attention mechanism. We show the effectiveness of this method through various experiments for JSSP, quantum chemistry, and max-cut problems. SA-DQAS still has limited performance in quantum chemistry. However, JSSP experimental results show that SA-DQAS can find more stable circuit structures than DQAS in ideal and noisy environments.

However, the circuit depth and width are fixed, and being especially small during a search may limit the search performance. Some search results still hardly depend on the size and selected gate types of the operation pool, and a larger operation pool increases the search space exponentially, leading to tremendous training costs. Addressing these issues could be the next step.

Acknowledgements

The project of this conference paper is based on was supported with funds from the German Federal Ministry of Education and Research in the funding program Quantum Reinforcement Learning for industrial Applications (QLindA) and the Federal Ministry for Economic Affairs and Climate Action in the funding program Quantum-Classical Hybrid Optimization Algorithms for Logistics and Production Line Management (QCHALLenge) - under project number 13N15644 and 01MQ22008B. The sole responsibility for the paper's contents lies with the authors.

References

[1] M. Alam, S. Kundu, R. O. Topaloglu, and S. Ghosh, "Quantum-classical hybrid machine learning for image classification (iccad special session paper)," *arXiv preprint arXiv:2109.02862*, 2021.

[2] W. Wu, G. Yan, X. Lu, K. Pan, and J. Yan, "Quantumdarts: Differentiable quantum architecture search for variational quantum algorithms," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, Honolulu, Hawaii, USA: JMLR.org, 2023.

[3] Y. Ma, V. Tresp, L. Zhao, and Y. Wang, "Variational quantum circuit model for knowledge graph embedding," *Advanced Quantum Technologies*, vol. 2, no. 7-8, p. 1 800 078, 2019.

[4] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano, and M. Fiorentini, "A case study of variational quantum algorithms for a job shop scheduling problem," *EPJ Quantum Technology*, vol. 9, no. 1, p. 5, 2022.

[5] Y. Sun, J. Liu, Y. Ma, and V. Tresp, "Differentiable quantum architecture search for job shop scheduling problem," *arXiv preprint arXiv:2401.01158*, 2024.

[6] M. Cerezo, A. Arrasmith, R. Babbush, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[7] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao, "Quantum circuit architecture search for variational quantum algorithms," *npj Quantum Information*, vol. 8, no. 1, pp. 1–8, 2022.

[8] F.-X. Meng, Z.-T. Li, X.-T. Yu, and Z.-C. Zhang, "Quantum circuit architecture optimization for variational quantum eigensolver via monto carlo tree search," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–10, 2021.

[9] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Differentiable quantum architecture search," *Quantum Science and Technology*, vol. 7, no. 4, p. 045 023, 2022.

[10] L. Ding and L. Spector, "Evolutionary quantum architecture search for parametrized quantum circuits," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 2190–2195.

[11] L. Ding and L. Spector, "Multi-objective evolutionary architecture search for parameterized quantum circuits," *Entropy*, vol. 25, no. 1, p. 93, 2023.

[12] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, "Quantum circuit optimization with deep reinforcement learning," *arXiv preprint arXiv:2103.07585*, 2021.

[13] E.-J. Kuo, Y.-L. L. Fang, and S. Y.-C. Chen, "Quantum architecture search via deep reinforcement learning," *arXiv preprint arXiv:2104.07715*, 2021.

[14] E. Ye and S. Y.-C. Chen, "Quantum architecture search via continual reinforcement learning," *arXiv preprint arXiv:2112.05779*, 2021.

[15] M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko, "Reinforcement learning for optimization of variational quantum circuit architectures," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 182–18 194, 2021.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[17] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz," *Quantum Science and Technology*, vol. 4, no. 1, p. 014 008, 2018.

[18] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Physical Review A*, vol. 94, no. 5, p. 052 325, 2016.

[19] A. Zlokapa and A. Gheorghiu, "A deep learning model for noise prediction on near-term quantum devices," *arXiv preprint arXiv:2005.10811*, 2020.

A. Operation Pool

We define four root operation pools $\mathcal{O}_1$-$\mathcal{O}_4$ to create concrete operation pools. Each root contains only the type of quantum gates with a generic range. We define 'RZ+RY+CZ+H' ($\mathcal{O}_1$) and 'U3+CU3+H' as two essential roots since the circuits based on these two design spaces perform well in quantum optimization problems. Furthermore, we define two additional roots by making a small change in 'RZ+RY+CZ'. One is by extending it with CNOT ($\mathcal{O}_3$), and the other is created by replacing CZ with CNOT ($\mathcal{O}_2$).

The smallest and largest operation pool of $\mathcal{O}_4$ with five qubits in JSSP for each type is shown as follows:

$$\begin{aligned}
\text{op4-1} = \{&\texttt{U3}:[0,1,2,3,4],\\
&\texttt{U3}:[0,1,2,3],\texttt{U3}:[1,2,3,4],\\
&\texttt{U3}:[0,1,2],\texttt{U3}:[1,2,3],\texttt{U3}:[2,3,4],\\
&\texttt{U3}:[0,1],\texttt{U3}:[1,2],\texttt{U3}:[2,3],\texttt{U3}:[3,4],\\
&\texttt{CU3}:[0,1,2,3],\texttt{E}:[0,1,2,3,4]\}\\
\text{op4-4} = \{&\texttt{U3}:[0,1,2,3,4],\texttt{H}:[0,1,2,3,4],\\
&\texttt{CU3}:[0,1,2,3],\texttt{E}:[0,1,2,3,4]\}
\end{aligned} \tag{14}$$

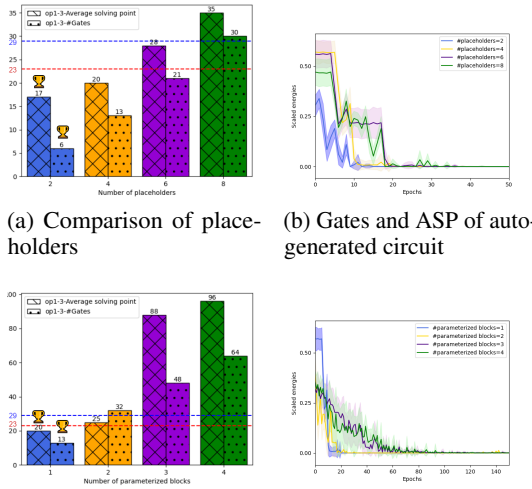
The last operation E means that add an identical gate to each qubit. We change the size following this rule: every time we reduce the size, we delete all single-gate operations with the current smallest length of the working range. For example, the largest operation pool of $\mathcal{O}_4$ contains U3 with a length of working range of 2. The smallest length of working range in the smallest pool of single-qubit gates is 5. As a result, for each type, with N qubits, we will form $N-1$ different sizes of operation pools, and we number them in order from largest to smallest. In our experiments, our operation pools of different qubits are formed to obey this rule. In some experiments, we change the operation pool slightly, the working range of entangle gates [0,1,2,3,4,5,6,7].

B. JSSP

The impact of placeholders and parameterized blocks during generating circuits is studied. In Figure 7, the red line indicates the number of gates of the baseline is 23 and the ASP is 29. The evaluation results show that as the number of placeholders and parameterized blocks increase, the number of gates in the generated circuits and the depth of the circuits increase and converge more slowly when solving a simple JSSP. The deeper the circuits are, the more parameterized gates need to be trained, resulting in more complex quantum calculations and slower convergence.

C. Max-cut Problem

SA-DQAS adds self-attention in two methods for the Max-cut problem in the benchmark and compares their performances. We select some generated circuits to make a comparison. The results are shown in Figure 9. In Figure 9b, circuits created with method 1 converge faster to the minimum energy than circuits searched by method 2. Figures 9c and 9e show that circuits searched by method 1 outperform those by method 2. Some circuits generated with these two methods perform similarly for the Max-cut problem, as shown in Figure 9d and 9f. Furthermore, in Figure 9a, although the circuit from method 2 converges fast, it has a large deviation. The circuits generated by the two methods



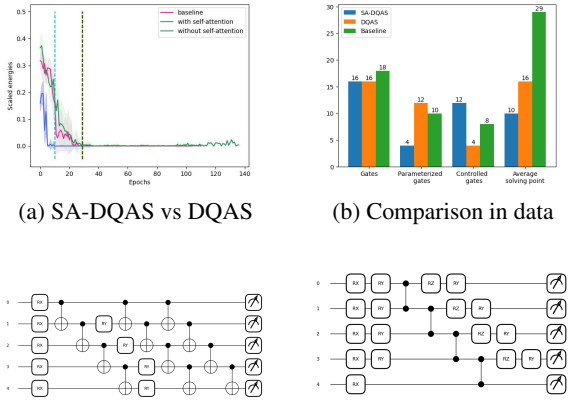
(a) Comparison of placeholders

(b) Gates and ASP of auto-generated circuit

(c) Comparison of parameterized blocks

(d) Gates and ASP of auto-generated circuit

Figure 7: (a) and (b) illustrate the impact of placeholders when generating circuits with op1-3. (c) and (d) illustrate the impact of parameterized blocks when generating circuits with op1-3.



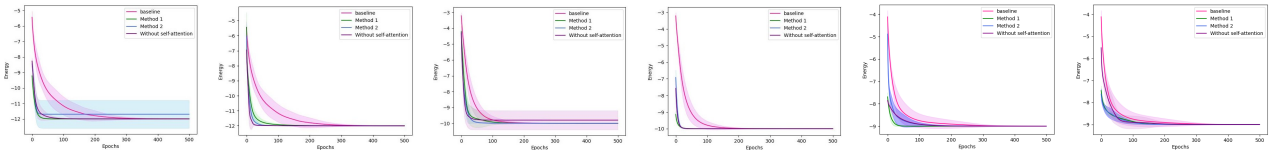
(a) SA-DQAS vs DQAS

(b) Comparison in data

(c) Circuit by SA-DQAS

(d) Circuit by DQAS

Figure 8: Comparison of circuits generated using SA-DQAS and DQAS. (a) performances for solving JSSP, (b) comparison in data, (c) one circuit generated with SA-DQAS, (d) one circuit generated with DQAS.



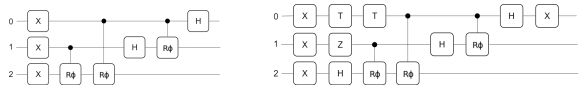
(a) W<WO op2-4_4 (b) W>WO op4-7_4 (c) W≈WO op4-5_4 (d) W≈WO op3-3_4 (e) W<WO op3-1_4 (f) W>WO op3-6_4

Figure 9: A comparison of the performance of circuits generated with (methods 1 and 2) and without self-attention in solving different Max-cut problems. (a),(b) are evaluation results for Random graph, (c), (d) are for Ladder graph and (e), (f) are for Barbell graph. W means generating circuits with self-attention, and WO means without self-attention.

perform similarly in solving different Max-cut problems, and no circuit generated by one method will always perform better. However, the circuits generated in both ways find the optimal solution faster than the baseline. Moreover, the circuit generated by method 1 has a more stable and reliable performance and is not sensitive to different initial parameters.

In this part, we compare the structural and performing differences between the circuits generated with SA-DQAS and DQAS and the performances in solving different Max-cut problems in the benchmark. When we look into the results shown in Figures 9a and 9e, we can know that circuits generated by SA-DQAS, no matter which method is used, these circuits outperform those created without self-attention. And circuits in Figures 9c, indicating these circuits don't perform stably. Circuits created by DQAS, shown in other figures, perform better than those designed with SA-DQAS. However, if we look into their circuit architectures, the data are given in Table 2. Circuit$_{R-\mathtt{op4}-7_4}$ with method 2 contains the fewest gates, especially parameterized ones. Next, we look into the evaluation results of the Ladder graph. Circuits derived from op3-3 by these three methods show similar performances, converging to the minimum energy almost simultaneously and containing a similar number of each gate type. The circuit generated by DQAS for the Barbell graph doesn't contain any controlled gates. Controlled gates are essential in quantum circuits because they allow us to create interactions and entanglement between quantum qubits, enabling various quantum computation and quantum information processing tasks. A circuit without controlled gates is indistinguishable from a classical circuit. However, the hardware costs for entanglement in circuits designed by SA-DQAS are higher.

From these comparison results, we can know that by adding self-attention to DQAS, a better circuit structure that keeps the quantum features by containing entanglements can be found, and the circuit shows better stability and repeatability when solving problems and is insensitive to initial parameters. Moreover, the number of gates in the generated circuit is less, which saves the hardware resources needed to build the circuit and reduces the errors and hardware noise generated by the quantum gate itself to a certain extent.



(a) Ideal circuit (b) Circuit from SA-DQAS

Figure 10: Evaluation of newly found architectures on the simulator without noise. The results are averaged over 10 trials with the different initial parameters.

D. Fidelity

In this section, we test our algorithm for error mitigation and study the fidelity between the pre-defined circuits and circuits created by SA-DQAS. An experiment has been done in [9] which proved that DQAS can be used for error mitigation. We repeat this experiment but generate new circuits with SA-DQAS. The pre-defined circuit is a three-qubit QFT circuit. An `x` gate is added on each qubit at the beginning for encoding. In the searching process, we set up six placeholders in the circuit, half before the `QFT` part and half after it, just like the original experimental setting in [9]. The operation pool used in this experiment is $\mathcal{O}_{f1}$ (shown in Appendix E), which only contains single-qubit operations. A noise model assumes that there are 2% BitFlip errors between two gates and 20% BitFlip errors when a qubit is idle. We design circuits by SA-DQAS in an ideal environment and pick the one with the highest fidelity to the ideal circuit in a noisy environment. The fidelity between the auto-generated circuit by SA-DQAS (illustrated in 10a) and the ideal circuit is 0.66, which is a little higher than the fidelity (0.6) between the ideal circuit and the circuit derived from DQAS. It indicates that SA-DQAS outperforms DQAS in error mitigation.

We build three more noise models based on BitFlip. The first type is designed by adding BitFlip errors with different occurrence probabilities on each qubit at the end. The second type adds 2% BitFlip errors after each operation in the non-empty (the chosen operation is not "`E`") placeholders of the generated circuit and adds a 20% BitFlip to each qubit at the

Table 4: A table that records the fidelity between the generated circuit and the ideal circuit under different probabilities of BitFlip noise when the placeholders are placed at different positions. The circuits are automatically generated under an ideal environment ($B_{0.0}$) and 20% BitFlip noise ($B_{0.2}$).

*/**		$B_{0.0}$	$B_{0.1}$	$B_{0.2}$	$B_{0.3}$
Back	$B_{0.0}$	0.94	0.79	0.68	0.57
	$B_{0.2}$	0.97	0.81	0.69	0.57
Back and Front	$B_{0.0}$	0.97	0.79	0.68	0.57
	$B_{0.2}$	0.94	0.78	0.67	0.57
Front	$B_{0.0}$	0.99	0.77	0.65	0.54
	$B_{0.2}$	0.99	0.79	0.66	0.55

* The column headings of the table indicate the environments when searching for a circuit with SA-DQAS. The row headers indicate the environment the generated circuits are in when calculating the fidelity between them and the ideal circuit.

** We search five architectures in each environment and calculate the average fidelity under different environments.

end of the circuit. The third one uses the same noise model given in [9]. Now we calculate the fidelity between the ideal circuit and generated circuits in environments where BitFlip errors occur with different probabilities.

From Table 4 we get the results. As the probability of the occurrence of BitFlip increases, the average fidelity decreases. When placeholders are all behind or before the `QFT` part, the circuits generated in a noisy environment have higher average fidelity than those created in an ideal environment. While the `QFT` part is in the middle of the circuit, the situation is precisely the opposite. Circuits are more stable and reliable when created in a noisy environment. Their structures are better and suffer less from BitFlip noise, making them more noise-resilient. Moreover, the noise resistance of the generated circuit is also related to the circuit structure.

In [9] they only use single-qubit operations in the operation pool. Such quantum error mitigation tricks can transfer the coherent errors to the easily handled Pauli errors [9], [18], [19]. We add controlled gates into the operation pool form a new operation pool $\mathcal{O}_{f2}$ (given in Appendix E). In this way, the automatically designed parameterized blocks in the circuit architectures can consist of entangled gates. Together with the operation pool containing single-qubit gates, we generate circuits from both operation pools in an ideal environment or add 20% BitFlip on each qubit at the end of the entire circuit. The average fidelity of the searched circuits is calculated in an ideal environment or with BitFlip 2 and 3 The comparisons are shown in Table 5.

When circuits (except the `QFT` part) contain controlled gates, whether in ideal or noisy environments, the average fidelity is lower than those that only contain single-qubit gates. Controlled gates connect multiple qubits, and when one of them, especially the control qubit, is perturbed by an error, it affects the state of the other qubits. So, the controlled gates suffer from noise errors with a larger margin of error, resulting in lower fidelity of the final circuit. Furthermore, more controlled gates in a circuit can cause qubits to last longer in idle states. When a qubit is idle, it has a higher probability of suffering from errors called idling errors.

Table 5: A table that records the fidelity between the generated circuit from operation pools with and without controlled gates and the ideal under different environments when the placeholders are in three positions of the `QFT` part.

		Ideal		BitFlip 2		BitFlip 3	
		Y*	N*	Y	N	Y	N
Back	$B_{0.0}$	0.91	0.94	0.66	0.68	0.56	0.65
	$B_{0.2}$	0.94	0.97	0.67	0.69	0.63	0.65
Back and Front	$B_{0.0}$	0.89	0.97	0.63	0.68	0.53	0.66
	$B_{0.2}$	0.89	0.94	0.64	0.67	0.51	0.65
Front	$B_{0.0}$	0.99	0.99	0.65	0.65	0.50	0.64
	$B_{0.2}$	0.99	0.99	0.64	0.66	0.47	0.64

* Y indicates the operation pool contains controlled gates, and N indicates only single-qubit gates are in the operation pool.

E. Settings

Experiments settings:

Table 6: Basic settings in different experiments.

Experiments	#Qubits	#Placeholders	#Param. blocks*
Max-cut	8	4	1
JSSP	5	4	1
Fidelity	3	6	1

* #Param. blocks indicate the number of parameterized blocks used in the training process.

Operation pools:

Table 7: The types of gates in operation pools $\mathcal{O}_1$-$\mathcal{O}_4$

Operation pool	Gate type
$\mathcal{O}_1$	`RY`, `RZ`, `H`, `CZ`
$\mathcal{O}_2$	`RY`, `RZ`, `H`, `CNOT`
$\mathcal{O}_3$	`RY`, `RZ`, `H`, `CZ`, `CNOT`
$\mathcal{O}_4$	`H`, `U3`, `CU3`

Benchmark graphs of Max-cut problem:

Baseline in Max-cut problem:

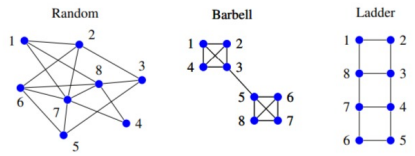


Figure 11: Benchmark graphs used for Max-cut problem.

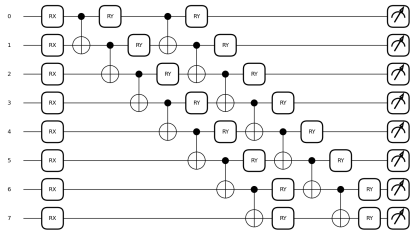


Figure 12: Baseline used for Max-cut problem.

Operation pools used for study the fidelity:

$$
\begin{aligned}
\mathcal{O}_{f1} = \{&\texttt{X}:[0],\texttt{X}:[1],\texttt{X}:[2],\\
&\texttt{T}:[0],\texttt{T}:[1],\texttt{T}:[2],,\\
&\texttt{E}:[0,1,2]\\
\mathcal{O}_{f2} = \{&\texttt{X}:[0],\texttt{X}:[1],\texttt{X}:[2],\\
&\texttt{T}:[0],\texttt{T}:[1],\texttt{T}:[2],\\
&\texttt{CNOT}:[0],\texttt{CNOT}:[1],\texttt{CNOT}:[2],\texttt{CNOT}:[0,1,2],\\
&\texttt{CZ}:[0],\texttt{CZ}:[1],\texttt{CZ}:[2],\texttt{CZ}:[0,1,2],\\
&,\texttt{E}:[0,1,2,3,4]\}
\end{aligned}
\tag{15}
$$