AI-Powered Algorithm-Centric Quantum Processor Topology Design

Tian Li[1], Xiao-Yue Xu[1], Chen Ding[1], Tian-Ci Tian[1], Wei-You Liao[1], Shuo Zhang[1], He-Liang Huang[1]*

[1]Henan Key Laboratory of Quantum infommation and Cryptography
Zhengzhou, Henan, 450000, China.
quanhhl@ustc.edu.cn(corresponding author)

Abstract

Quantum computing promises to revolutionize various fields, yet the execution of quantum programs necessitates an effective compilation process. This involves strategically mapping quantum circuits onto the physical qubits of a quantum processor. The qubits' arrangement, or topology, is pivotal to the circuit's performance, a factor that often defies traditional heuristic or manual optimization methods due to its complexity. In this study, we introduce a novel approach leveraging reinforcement learning to dynamically tailor qubit topologies to the unique specifications of individual quantum circuits, guiding algorithm-driven quantum processor topology design for reducing the depth of mapped circuit, which is particularly critical for the output accuracy on noisy quantum processors. Our method marks a significant departure from previous methods that have been constrained to mapping circuits onto a fixed processor topology. Experiments demonstrate that we have achieved notable enhancements in circuit performance, with a minimum of 20% reduction in circuit depth in 60% of the cases examined, and a maximum enhancement of up to 46%. Furthermore, the pronounced benefits of our approach in reducing circuit depth become increasingly evident as the scale of the quantum circuits increases, exhibiting the scalability of our method in terms of problem size. This work advances the co-design of quantum processor architecture and algorithm mapping, offering a promising avenue for future research and development in the field.

Code — https://github.com/qclab-quantum/Qtailor

1 Introduction

The advent of quantum computing marks a significant leap forward in computational capabilities (Wu et al. 2021; Nielsen and Chuang 2010; Shor 1997; Deutsch and Jozsa 1997; Shor 1994; Zhu et al. 2022), offering the potential to outperform classical computing in solving specific intricate problems. The field has witnessed remarkable progress in recent decades (Arute et al. 2019), having now entered the Noisy Intermediate-Scale Quantum (NISQ) era(Huang et al. 2023, 2020; Wu et al. 2021). Despite this progress, the realization of quantum advantage in practical applications, such as quantum machine learning (Biamonte et al. 2017; Beer et al. 2020; Liu et al. 2021; Huang et al. 2021; Ding, Bao, and Huang 2022) and quantum chemistry (Bauer et al. 2020; Cao et al. 2019), remains a huge challenge. This is primarily attributed to the fragile nature of physical qubits on noisy quantum processors (Schlosshauer 2019), making it difficult to execute complex quantum algorithms reliably. A pivotal strategy to mitigate the impact of noise is to minimize the depth of quantum circuits, which underscores the imperative for an efficient quantum circuit compilation process. This process involves mapping theoretical quantum algorithms onto the physical qubits within a processor, taking into account the native gate set and the processor's topology, with the goal of achieving the shallowest possible circuit depth.

Previous works have predominantly concentrated on circuit mapping for fixed processor topologies (Zhang et al. 2021; Li, Ding, and Xie 2019; Wille, Burgholzer, and Zulehner 2019; Zulehner, Paler, and Wille 2018; Siraichi et al. 2018; Li, Zhou, and Feng 2021; Zhou, Feng, and Li 2022; Nannicini et al. 2022), such as linear or two-dimensional grid configurations. The topology of a processor significantly influences the depth of the mapped circuits, with higher connectivity generally leading to shallower depths. However, for certain quantum algorithms, a one-size-fits-all topology is not always necessary; instead, the processor can be tailored to the algorithm for optimal performance. This raises an intriguing question: Can tailoring the topology to specific circuits lead to further reductions in circuit depth? If affirmative, what methodologies can be employed to effectively engineer such tailored topologies? Our work addresses these questions by harnessing machine learning to simultaneously optimize the quantum processor topology and map quantum circuits for the intended quantum algorithm, thereby significantly reducing the depth of the mapped circuits. We summarize our key contributions as follows:

- We introduce *Qtailor*, a novel framework that utilizes Reinforcement Learning (RL) to determine the optimal quantum processor topology for a specific quantum algorithm, within the constraints of a processor's connectivity. Experimental tests with small-scale circuits indicate that *Qtailor* can significantly reduce circuit depth, with enhancements reaching up to 46% over the current state-of-the-art model, *Qiskit* (Contributors and IBM

*Corresponding author.

2024). Moreover, as circuit size escalates, the advantages of *Qtailor* become progressively pronounced.

- We develop a reward-replay mechanism for training *Qtailor*, which recycles rewards from previous actions to reduce the need for time-consuming action evaluations, thus enhancing training efficiency and making *Qtailor* well-suited for addressing large-scale problems.
- We propose a directed-force layout technique that significantly minimizes edge crossings in the designed quantum computing processor, reducing crosstalk noise and optimizing the layout for the grid-like architecture commonly found in modern superconducting quantum computers.

2 RL for Processor Topology Design with Restricted Connections

In the context of this study, we represent physical qubits and their connections using an undirected graph, with qubits as vertices (**v**) and their connections as edges (**e**). As elaborated in Appendix A.4, our empirical evidence indicates that variances between two graph can lead to a significant divergence in circuit depth. It is, therefore, crucial to identify an optimal graph for circuits. Considering a set of n vertices, the total number of possible edges is determined by the number of ways to select 2 vertices from n, hence, there are $2^{\frac{n(n-1)}{2}}$ potential graphs, with varying structures that markedly affects the circuit. Due to the exponential growth in potential graphs with the number of vertices, it is hard for heuristic search algorithms to traverse the extensive solution space.

Reinforcement Learning (Silver et al. 2017; François-Lavet et al. 2018) is particularly well-suited to this domain because its adaptive, incremental learning process, a trade-off between exploration and exploitation, enabling effective exploration in vast solution space. Consequently, our approach leverages RL to develop an agent that can dynamically guide topology design, moving beyond mere static memorization of solutions. Nevertheless, integrating RL into this domain presents two primary challenges: **(a)** It requires the formulation of the topology design problem within the confines of a reinforcement learning framework. This involves developing novel action space and a reward function that are compatible with reinforcement learning algorithms. **(b)** There is a critical need to improve the training efficiency of models designed for large-scale quantum circuits. This challenge arises from the inherent computational demands and the complexity of quantum circuits, which can significantly hinder the speed and effectiveness of model training. These challenges are addressed in Sections 2.1 and 2.2, respectively.

2.1 System Overview

Consider a Graph with n vertices labeled $v_1, v_2, ..., v_n$, which correspond to n qubits in a quantum circuit. Let E represent the complete set of potential edges, where each edge e, signifying a potential link between any two vertices (or qubits), belongs to this set, i.e., $e \subseteq E$. A Graph is thus defined as a subset of E with each element of this subset representing an edge in the Graph. The depth of a circuit (c) mapped onto a Graph (G) can be computed using the function $f_{depth}(G, c)$. In our experimental setup, we utilized a famous quantum development kit *Qiskit* (Contributors and IBM 2024) to implement the computation of $f_{depth}(G, c)$. The objective is to identify a G to minimizes the circuit depth while adhering to the maximum connectivity degree constraint for its vertices, formally stated as:

$$\begin{aligned} &\min f(G, c), G \subseteq E \\ &s.t.\ degree(v) \leq 4 \end{aligned} \tag{1}$$

In the context of the RL framework, we conceptualize G as the **environment**, wherein each action executed by the agent can modify G, propelling G towards the desired state. The G is represented by an adjacency matrix, denoted by M, in which the presence of an edge between vertices v_i and v_j is indicated by $M_{ij} = 1$. It is important to note that Graph G is undirected; consequently, an edge constitutes an unordered vertex pair, rendering $e\langle v_i, v_j\rangle$ and $e\langle v_j, v_i\rangle$ as equivalent representations of the same edge. In light of this symmetry, we use only the lower triangular portion of M where $i < j$, effectively reducing the state space by over half. As illustrated in Figure 2, matrix M is subsequently flattened into a one-dimensional array to serve as the **state** (input) for the agent. The **agent** employed in our study is borrowed from the Proximal Policy Optimization (PPO) as detailed in (Schulman et al. 2017).

The extensive search space inherent in graphs makes it challenging for an agent to directly identify an optimal graph, especially during the initial stages when both the agent's policy and output are nearly random. This randomness results in sparse positive rewards, complicating the model's convergence. To address this, we simplify the huge action space into incremental steps. At each step, the agent only decide whether to build an edge between two specified vertices, consequently, the **action** can be denoted by $\langle v_i, v_j\rangle$. Given n vertices the action space is effectively reduced to $\frac{n(n-1)}{2}$, a significant contraction from the entire Graph's search space. Reward is quantified as a scalar value indicates the improvement of circuit depth (the lower the better) by action. An efficient reward function is detailed in Section 2.1. Furthermore, to enhance the efficiency of training, a **Reward-Replay** method is introduced in Section 2.2.

Reward Function Our objective is for the reward mechanism to guide the agent towards optimizing the topology towards an ideal state. We start by denoting the initial depth of the circuit as D_0, During the $i-th$ iteration, the change in depth relative to the previous iteration is represented by $\Delta(D_{i-1}, D_i) = D_i - D_{i-1}$, Our objective is to ensure that this change, $\Delta(D_{i-1}, D_i)$, is negative, indicating a reduction in depth. Moreover, we strive for the depth at any iteration D_i to be lower than the initial depth D_0, highlighting a consistent trend towards optimization. Consequently, the reward is computed based on both $\Delta(D_0, D_i)$ and $\Delta(D_{i-1}, D_i)$. At any given time t, we assess depth change Δ from step $t-1$ and the initial step to step t respectively:

$$\Delta D_{t\to 0} = \frac{D_t - D_0}{D_0} \tag{2}$$

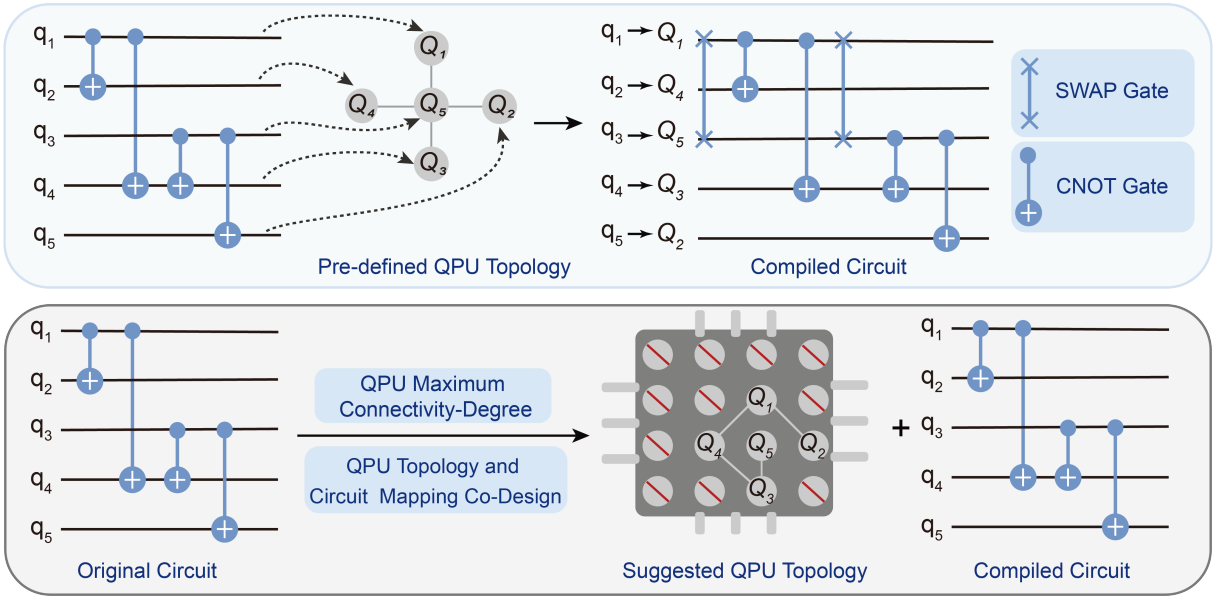


Figure 1: Comparative between the existing method and our method: **(top)** Prior research has focused on the development of mapping algorithms for predefined topologies, primarily aimed at allocating qubits in a way that satisfies connectivity requirements. The result of these methodologies is a fixed mapping protocol. **(bottom)** Our study employs Reinforcement Learning model to suggest an topology that aligns with the circuit's characteristics under the limitations imposed by restricted connectivity. Subsequently, qubits are mapped in a sequential manner instead of a complex mapping algorithms.

$$\Delta D_{t\to t-1} = \frac{D_t - D_{t-1}}{D_{t-1}} \tag{3}$$

According to Eq. (1) and Eq. (2), we formulate the reward as follows:

$$r = \begin{cases} \left((1+\Delta_{t\to 0})^2 - 1\right)\left|1+\Delta_{t\to t-1}\right|, \Delta_{t\to 0} < 0 \\ -\left((1-\Delta_{t\to 0})^2 - 1\right)\left|1-\Delta_{t\to t-1}\right|, \Delta_{t\to 0} \geq 0 \end{cases} \tag{4}$$

Our designed reward function incorporates both the initial state and the state from the preceding step, directing the agent to modify the topology at each step towards reducing the circuit depth. This ensures that the cumulative adjustments are aligned with the desired direction.

2.2 Reward-Replay Proximal Policy Optimization

The reward function and action space operates independently of RL algorithms, enabling a variety of RL to tackle the mapping problem. Among these, Proximal Policy Optimization (PPO) (Schulman et al. 2017) is preferred due to its benefits in promoting stable training processes and efficient exploration strategies. Nonetheless, there still a significant obstacle that existing RL algorithms struggle to overcome: the evaluation of the action (i.e. assigning rewards to actions) at each training step relies on computing the circuit's depth using f_{depth}, a process that can take several seconds to minutes. This evaluation process significantly reduces the training efficiency, consuming approximately 68% of the total training time. This highlights the crucial need to reduce this duration to improve training efficiency.

Typically, the evaluation associated with a particular action depends on the current state (s), indicating that identical actions may yield divergent outcomes across different states. This concept is encapsulated within the reward function $r(s,a)$. A significant challenge emerges due to the extensive array of potential states, making the storage of every possible state-action pair $\langle s,a\rangle$ impractical. However, our experiments have yielded a notable observation: across numerous distinct pairs (s_p,a) and (s_q,a) where $s_p \neq s_q$,the results demonstrate a notable similarity in outcomes. In light of this observation, we simplify our model by adopting the assumption that $(a|s_1) \approx (a|s_2) \ldots \approx (a|s_m)$, thereby enabling us to approximate the reward function for a state-action pair $r(s,a)$ as simply $r(a)$. This simplification is rationalized by the premise that if an edge significantly contributes to minimizing the circuit's depth by enabling the requisite connections, then, regardless of the current state (i.e., the graph), this edge consistently ensures the necessary connectivity, thus uniformly facilitating a reduction in the circuit's depth. Building on this approximation, we introduce the Reward-Replay method, which involves recycling the reward associated with an a (represented as the pair $\langle r,a\rangle$) rather than $\langle r,a,s\rangle$, where the latter signifies a nearly infinite array of possibilities. This strategy allows for the immediate retrieval of rewards from memory upon any subsequent execution of the action, eliminating the need for repetitive reward calculations and consequently diminishing computational demands.

Utilizing $r(a)$ to approximate the $r(a,s)$ may introduce inaccuracy. To counteract the potential accumulation of errors during training, a replay threshold is implemented for

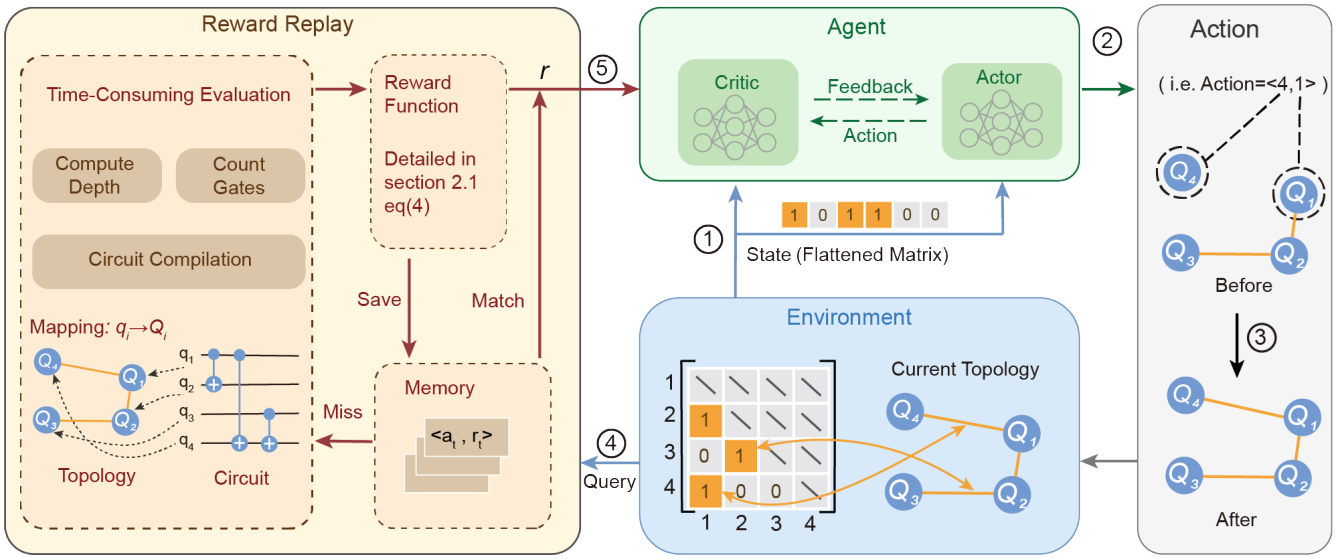


Figure 2: Overview of proposed *Qtailor*: **(1)** The agent acquires state from the environment; state are represented by a flattened matrix that denotes the current topology, where $M_{ij} = 1$ indicates that Q_i and Q_j are connected. **(2)** Subsequently, the agent outputs an action (a), that establish an connection between which two qubits. **(3)** The action is then applied to the topology. **(4)** Using the action as a key, we query the reward (r) from memory, which stores pairs of $\langle a, r \rangle$. If a match is found, the corresponding reward will be directly provided to the agent, otherwise, an evaluation involving circuit compilation, computation of depth, and gates is conducted. The reward function is then applied based on the depth or gates, and this reward is stored in memory as a pair of $\langle a, r \rangle$. This process is referred to as reward replay, detailed in Section 2.2. **(5)** Finally, The agent receives the reward and continues to the subsequent iteration.

retaining rewards in the buffer. For instance, setting a threshold of 2 means that the reward associated with action will be discarded after its second use. The balance between training efficiency and performance can be adjusted by modifying the replay threshold. Details regarding the hyper-parameters and the pseudo-code for the Reward-Replay Proximal Policy Optimization (RR-PPO) are provided in the Appendix A.5. It is important to emphasize that the framework we propose is not dependent on a specific reinforcement learning algorithm. Additionally, it does not require the use of a particular quantum software as a backend for processing quantum circuits.

3 Topology Adjustment for Industrial Manufacturing

Reinforcement learning offers promising topology for circuits, however, translating these topologies into practical, industrially viable implementations encounters significant challenges: **(a)** Electromagnetic interference between crossing wires can cause a cross-talk(Ding et al. 2020) problem (Sarovar et al. 2020), leading to decoherence and low fidelity in quantum operations, necessitating a topology design that minimizes wire crossovers to alleviate such problems. **(b)** Over past decades, the quantum processor architecture based on superconducting qubits has become the leading candidate platform (Huang et al. 2020). Given the predominant grid-like structure of these superconducting processor topologies (Nishio et al. 2020; Zulehner, Paler, and Wille 2018),

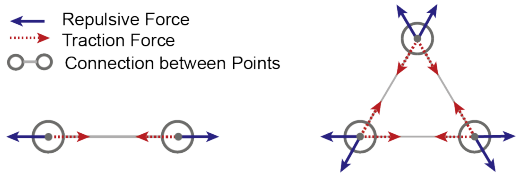


Figure 3: The forces acting between two and three vertex, each vertex undergoes both repulsive and attractive forces from the other vertex, with these forces counterbalancing each other to maintain the stability of vertex positions.

our goal is to maintain this uniform grid pattern. This approach aims to ensure compatibility with existing technologies, thereby facilitating a smoother integration of reinforcement learning-derived topologies into the current framework of quantum computing.

In this study, we adopt the Force-directed Layout method (Kobourov 2012) to minimize edge crossings by emulating physical forces, specifically repulsion and attraction. As depicted in figure 3, vertexes are treated as charged particles, and edges are treated as springs, each vertex experiences two types of forces: **(a)** Repulsive force defined as $F_{repulsive} = \frac{k_1}{r^2}$, k_1 is a constant, r is the distance between vertexes; **(b)** Attractive force: defined as $F_{traction} = k_2 \Delta x$, k_2 is a constant and Δx is the value of the increase of the distance between vertexes compared to the initial distance. The resultant force acting on each vertex is determined by

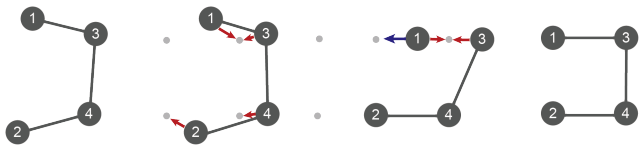


Figure 4: Workflow of the Force-directed Grid Layout: **(a)** Acquire an initial force-directed layout using existing method (Kobourov 2012). **(b)** Initialization the grid point around the point, the point are attracted to the nearest surrounding grid point, vertexes determine the nearest grid point via Euclidean distance, followed by occupation. **(c)** In instances where v_1 and v_3 share the closest grid point, they both move to that point. However, an increase in the repulsive force between them as they draw closer ensures that they are effectively repelled. **(d)** Upon occupation of a grid point by a point, the nearest unoccupied grid points is selected as the new target for other points.

the sum of all repulsive and attractive forces. Subsequently, the vertex adjusts its position in response to this resultant force. This adjustment process is iteratively repeated until the system reaches equilibrium, as detailed in Appendix A.6. Equilibrium is characterized by a state of low entropy, leading to minimal edge crossings. While the layout method does not guarantee an absolute minimum number of crossings, its cost-effectiveness allows for multiple executions to identify the most optimal result. However, this method faces challenges in achieving a grid-like arrangement of vertices. To address this issue, we have incorporated a grid component into the existing force-directed method. This additional component directs vertices towards the nearest grid points, ensuring that vertices settle into a coherent grid pattern. The workflow of this component is illustrated in Figure 4.

4 Experiments

In this section, we present the experimental design aimed at addressing the following research questions: **(RQ1)** How does the performance of *QTailor* compare to the state-of-the-art approach, and what are the reasons for the observed performance differences? **(RQ2)** How effective is *QTailor* as the circuit scales up? **(RQ3)** How does our proposed Reward-Replay impact the efficiency of PPO?

4.1 Experiment Setup

Benchmarks and Compared Method . Building upon the foundation established by previous research, we have selected MQT Bench (Quetschlich, Burgholzer, and Wille 2023) as our benchmark toolkit. This toolkit offers a diverse array of platform-independent circuits tailored for various quantum computing tasks. To monitor the training process and evaluate efficiency, we utilized TensorBoard (Abadi et al. 2015) Our experiments were conducted using *Qiskit* (Contributors and IBM 2024) as the backend framework, a renowned quantum computing development kit provided by IBM. Notably, *Qiskit* incorporates Sabre (Li, Ding, and Xie 2019) as its mapping algorithm, which is recognized as the state-of-the-art. Sabre utilizes a sophisticated mapping algorithm that operates on a SWAP-based Bidirectional heuristic search approach.

For our study, Sabre was chosen to ensure a fair and intuitive comparison. By employing Sabre, we maintain consistency across all phases of our experiment, with the exception of the mapping phase. This approach allows us to utilize the same configurations for post-mapping optimizations and circuit depth calculations, thereby ensuring a standardized and controlled environment for our comparative analysis.

Evaluation Protocol. For the Sabre algorithm, a topology consisting of a square grid with 100 nodes (10×10) is provided, where nodes having a connectivity degree of 4. For a fair comparison, the maximum connectivity degree for the *QTailor's* topology is similarly constrained to 4. Given that the calculation of circuit depth involves randomness, we conducted the calculation three times and took the average to mitigate the effects of this randomness. The detailed hyperparameters are given in Appendix A.5.

4.2 Performance Evaluation (RQ1)

In this section, we compare the depth, fidelity, and total gates of circuits driven by *QTailor* and *Qiskit*. The circuits therein is from public dataset. Our analysis indicates that *QTailor* yields better outcomes than *Qiskit* across a majority of tasks, reducing circuit depth ranging from 5% to 46% as presented in Figure 5, over 60% of the samples exhibit a reduction in circuit depth by 20%. The results shows that the topology suggest by *QTailor* for circuits can also significantly reduce total gates of compiled circuits, which can mitigate the accumulation of quantum errors, leading to higher overall fidelity. Our method can be easily optimized for different metrics by making a minor adjustment to the reward function. Specifically, we can replace the parameter depth (denoted by ΔD) with the number of gates. Similar to depth, the number of gates should be minimized. Conversely, if the metric in question should be maximized, we need to use its negative value in the reward function.

The statistics presented in Table 1 provide insight into the factors contributing to the reduced depth of the line. The idle ratio, which represents the proportion of time a qubit remains inactive in gate operations, is calculated using the following formula:

$$Idle = 1 - (\frac{gates}{qubits \times depth}) \tag{5}$$

We observe that the idle ratio from *Qtailor* is lower than that of *Qiskit*, with a maximum observed reduction of 31.68%, regardless of whether *Qtailor's* gate count is higher or lower than *Qiskit's*. A lower idle ratio indicates enhanced parallel processing capabilities of the quantum processor and contributes to a reduction in circuit length. If the focus shifts towards reducing the number of gates, a minor adjustment can be made to the reward function by replacing the depth parameter (denoted by ΔD) while other components remain unchanged. The result are shown in Figure 6.

The Quantum gates in circuits are applied to qubits to perform computations. Each gate takes a certain amount of

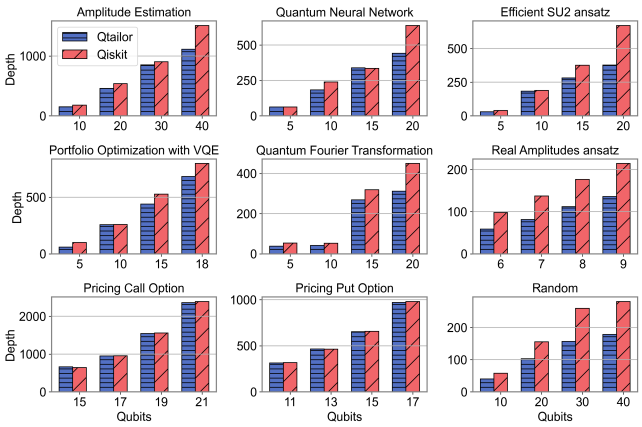


Figure 5: A comparative evaluation involving *QTailor* and *Qiskit*. The x-axis represents the circuit size quantified by the number of qubits, while the y-axis denotes the circuit depth after the mapping process.

Circuit	*Qiskit* Depth	*Qiskit* Idle(%)	*Qtailor* Depth	*Qtailor* Idle(%)	Reduction(%)
qnn_5	92	76.52	63	66.70	31.52↓
qnn_10	191	75.29	184	76.47	3.66↓
qnn_15	483	84.89	330	77.52	31.68↓
qnn_20	610	83.67	445	74.04	27.05↓
vqe_10	257	74.09	253	67.39	9.04↓
vqe_15	507	79.20	428	69.66	12.04↓
ae_10	191	84.29	162	81.67	3.11 ↓
ae_20	527	74.65	459	70.04	6.18 ↓
ae_40	1353	58.47	1069	40.79	30.24 ↓
ansatz_6	95	86.32	55	81.27	5.85 ↓
ansatz_9	193	83.63	136	81.84	2.14 ↓
su2_5	62	87.10	22	74.55	14.41 ↓
su2_20	480	64.21	394	41.24	35.77 ↓

Table 1: Statistics of depth, and idle ratio on circuits. Results suggests that minor variations in idling rates can culminate in substantial differences in circuits depths.

time to execute, the gates must be executed within an operational window dictated by the qubits coherence times, such as T1 (relaxation time) and T2 (dephasing time). The T1 and T2 define the duration for which a qubit can maintain its state reliably, if operations exceed these time limits, qubits may lose their state, leading to errors. Fewer gates results in shorter operation times, which is crucial for enhancing fidelity—a metric indicating the accuracy of a quantum operation or circuit performance. The improvement in fidelity achieved through our method is depicted in Figure 7. We employ *Qtailor* to optimize and reduce the quantum gate count in the circuit. Next, we evaluate the fidelity of the circuit using Equation 6, with P_i and Q_i denoting the probabilities of the i-th quantum state for the ideal and noise-affected scenarios, respectively. The findings reveal that as the circuit size grows, the fidelity enhancement becomes more significant. In a 10-qubit circuit, there is an average fidelity improvement of 10.91%, with a maximum enhancement of 26%.

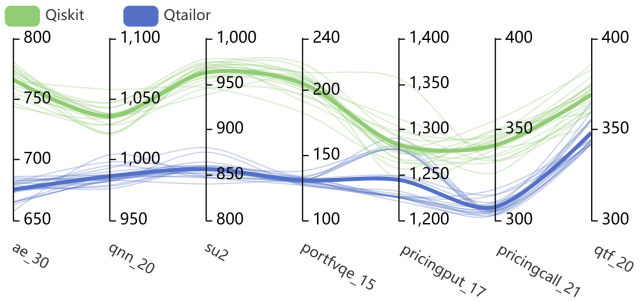


Figure 6: Statistics on the number of gates. Each curve represents a single experiment conducted across all circuits, with the bold curve indicating the mean value. Our approach reduced the total number of gates by 4.78% to 36.39%.

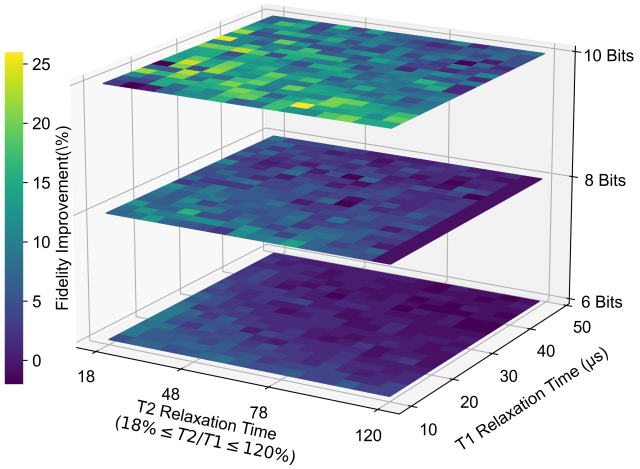


Figure 7: Fidelity improvement with gates reduction, the surfaces from top to bottom represent circuits of 6, 8, and 10 bits of Real Amplitudes ansatz task.

$$F(P,Q) = \left(\sum_i \sqrt{P(i)Q(i)}\right)^2 \tag{6}$$

4.3 Evaluation of Circuits for Scaling Up (RQ2)

Figure 8 shows how the depth changes with size of the circuits increase. In a comparative analysis focusing on the depth of quantum circuits comprising 6 qubits, *Qtailor* consistently exhibits a reduced circuit depth when compared to *Qiskit*. As the gate count increases from 60 (6×10) to 210 (6×35), the depth reduction achieved by *Qtailor* compared to *Qiskit* are **31.03%, 31.52%, 33.87% , 39.11%, 39.65% and 40.74%**, respectively. This trend indicating a systematic efficiency improvement in circuit depth with *Qtailor* as circuits size increases. The capability to efficiently handle large-scale circuits is crucial for unlocking the full potential of quantum computing, thereby enabling the solution of problems that are intractable for classical computing.

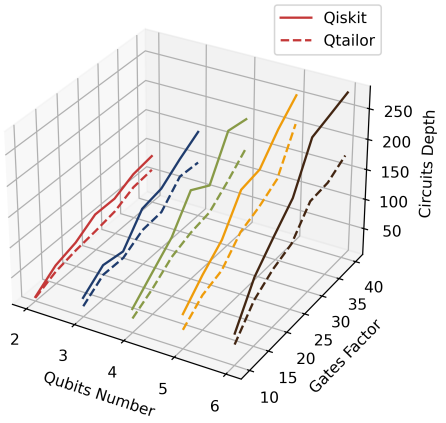


Figure 8: Depth of mapped circuits of varied sizes. The circuit's size is determined by both the number of qubits and gates. The number of gates is determined by multiplying the number of qubits by a factor, such as 2 on the axis indicating that the number of gates equals twice the number of qubits.

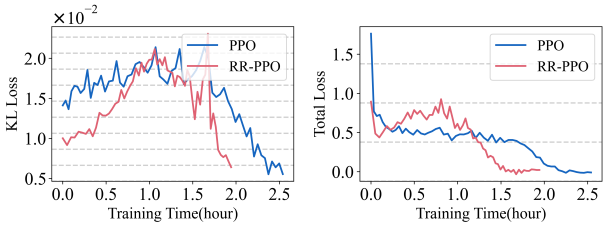


Figure 9: Comparison of the loss curves for PPO and RR-PPO. After a period of training, RR-PPO requires less time for sampling , resulting in a decrease in the time consumed per iteration.

4.4 Ablation Study (RQ3)

Figure 9 shows the Kullback-Leibler (KL) divergence loss and total loss during the training. Besides the Reward-Replay module, the experiments are conducted under uniform conditions, including hardware and hyper-parameters. We observe that: **(left)** Our method shows a more fluctuating curves for both KL and total loss, yet reaches convergence faster by 24% . This efficiency can be attributed to the Reward-Replay module, which minimizes unnecessary evaluations on circuits. **(right)** The losses curves eventually reach a nearly a near-identical value, because the forgetting mechanism avoids the errors accumulation and thus has a limited effect on training accuracy.

Figure 10 provide insights into the faster training process of RR-PPO, The results shows that the time consumed by RR-PPO gradually decreases and stabilization on both sampling and iteration phrase. Yet, the traditional PPO exhibits a contrasting trend. As training progressed, more reward can be replayed to avoid time-consuming evaluations. We also notice that the trends in the sampling time **(left)** and iteration time **(right)** curves exhibit strong similarity. Additionally, it is noteworthy that the iteration time encompasses the

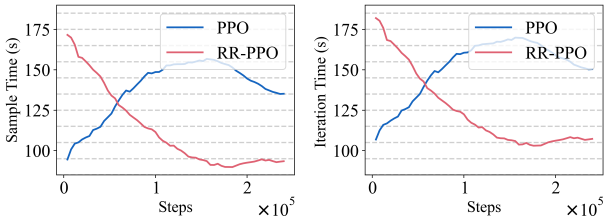


Figure 10: Time consuming. The time required to collect one sample **(left)** and to complete one iteration **(right)**. An iteration comprises sampling, gradient descent, and updating the policy. The results indicate a rapid decrease in the time expended by RR-PPO over time.

sampling time, implying that the total time to complete an iteration is mainly determined by the evaluation in sampling process.

5 Discussion and Conclusion

The proposed *Qtailor*, a framework that pioneers the synergistic design of quantum processor topology and circuit mapping, offers several remarkable features: 1) Compared to state-of-the-art methods, *Qtailor* significantly reduces the depth of mapped circuits, with a minimum of 20% reduction observed in 60% of the examined cases and a maximum enhancement reaching up to 46%. This reduction is pivotal for high-fidelity quantum computing, as circuit depth is a critical factor affecting the fidelity of quantum operations. This work, therefore, highlights the transformative potential of machine learning techniques in advancing quantum computing by transcending the limitations of conventional compilation methods. 2) Our method demonstrates excellent scalability across problem sizes and training efficiency. As illustrated in Section 4.3, the advantages of our approach become more pronounced with larger problem sizes, indicating its effectiveness for complex quantum circuits. Furthermore, as detailed in Section 4.3, the reinforcement learning framework we designed, which encompasses problem modeling and a problem-oriented training strategy called reward-replay proximal policy optimization, has yielded significant improvements in training efficiency. 3) We have also integrated a force-grid layout technique to optimize the topology on the processor, enhancing compatibility with current manufacturing technologies and mitigating issues related to crossing connections. This integration is crucial for the practical realization of quantum processors and addresses a key challenge in the transition from theoretical designs to manufacturable devices.

In the current study, we have focused on the connectivity constraints of quantum processors, an essential factor in processor design. However, different quantum physical systems or varying fabrication processes for quantum processors may entail distinct constraints. While our work provides a preliminary framework, it sets the stage for future research that can explore these nuances more deeply.

A Appendix

A.1 Related work

Circuit mapping involves the judicious allocation of qubits on a quantum processor which is known to be NP-complete (Boixo et al. 2018). One common approach is to reformulate the mapping issue as an mathematic problem and utilize a software solver (Lye, Wille, and Drechsler 2015; Oddi and Rasconi 2018; Venturelli et al. 2017, 2018; Bhattacharjee and Chattopadhyay 2017). Wille et al. (Wille, Burgholzer, and Zulehner 2019) addressed the mapping problem by incorporating it into a Satisfiability Modulo Theory(SMT) framework (Moura and Bjørner 2009) and they have successfully utilized the Z3 boolean satisfiability solver (de Moura and Bjørner 2008) to to obtain precise results for mapping quantum circuits onto IBM QX architectures. Another proposed method, BIPMapping (Nannicini et al. 2022) by Nannicini et al. models the mapping problem as an Integer Linear Problem (ILP) (Schrijver 1986) and and employs IBM CPLEX (IBM 2022) as a slover, the goal is to minimize a linear objective function subject to a set of linear constraints. BIPMapping consists three possible objective functions including error minimize, depth minimize and crosstalk minimize. Notably, BIPMapping outperforms Sabre in terms of reducing the number of CNOTs and exhibits higher fidelity.

A significant limitation of aforementioned methods is that a general solver is suffer from long runtimes and are only applicable to small-scale cases.As a solution, researchers have explored the use of search-based methods to find optimal mapping schemas (Zhang et al. 2021, 2023; Burgholzer, Schneider, and Wille 2022; Li et al. 2023; Zhou, Feng, and Li 2022; Zhou, Li, and Feng 2020; Li, Zhou, and Feng 2021; Fu et al. 2017; Zhang et al. 2020). Siraichi et al. (Siraichi et al. 2018) conducted a study on qubit mapping problems with IBM QX2 and QX3 processors, proposing a search algorithm based on dynamic programming to identify optimal solutions. However, this optimal algorithm requires exponential time and space for execution and is constrained to circuits with a maximum of 8 qubits. In 2019, Siraichi et al. (Siraichi et al. 2019) further developed an improved algorithm and evaluated its performance on IBM Q20 Tokyo. They employed a Bounded Mapping Tree (BMT) to explore the complex landscape of qubit mapping by breaking the problem down into subgraph isomorphism and token swapping problems.

Zulehner et al. (Zulehner, Paler, and Wille 2018) employed A^* Search algorithm (Zeng and Church 2009) along with a heuristic cost function to optimize the arrangement of two-qubit gates in separate layers, similar to Sabre. The main objective was to minimize the overall distance between coupled qubits in each layer while simultaneously reducing the circuit's depth. This approach taking only a few minutes to execute on small-scale circuits. However, it requires the analysis of all possible combinations of concurrent SWAP gates, which result in an excessively long runtime.

IBM introduced Sabre (Li, Ding, and Xie 2019) a state-of-the-art mapper with optimized heuristic search,Sabre utilizes the Floyd-Warshall algorithm to calculate the All Pairs Shortest Path and generate the matrix D. In this matrix, $D[i][j]$ represents the minimum number of swaps required to transfer a logical qubit from physical qubit Q_i to Q_j. The computation complexity of this step is $O(N^3)$. Afterward, Sabre traverses the entire circuit to create a Directed Acyclic Graph (DAG) that visualizes gate dependencies, with a complexity of $O(g)$. The DAG is then divided into independent layers, enabling concurrent execution of gates within each layer.

Liu et al. (Liu, Li, and Zhou 2022) proposed an optimization-aware algorithm that utilizes a similar mapping strategy as Sabre. However, their algorithm calculates the reduction in CNOT gates for each SWAP candidate in the routing process and selects the best swap candidate based on a cost function. Experimental results demonstrate that their algorithm reduces the number of CNOT gates by an average of 21.30% and the circuit depth by an average of 7.61% compared to Sabre.

Another important perspective is that circuit mapping should consider noise, as real-world processors exhibit significant variability in the error rates of qubits and their connections (Nishio et al. 2020). Tannu et al. (Tannu and Qureshi 2019) introduced a Variation-Aware Qubit Allocation (VQA) approach, which optimizes the movement and allocation of qubits towards stronger qubits and connection, thereby enhancing the reliability of NISQ systems by up to 2.5 times. Similarly, Murali et al. (Murali et al. 2019) conducted mappings on a real system using daily calibration data provided by IBM to prioritize qubit positioning, resulting in a reduced likelihood of communication (SWAP) errors. The results demonstrate that proper placement can lead to over 10 times improvement in the success rate of execution. However, this method suffers from the limitation that the initial qubit layout, which is optimal in terms of noise characteristics, may not be ideal for later qubit routing through swap mapping to the target topology. To overcome this issue, Nation et al. proposed MAPOMATIC (Nation and Treinish 2023) which involves remapping quantum circuits after qubit routing or post-compilation, to match low-noise subgraphs. They utilize output circuits generated with a SWAP-efficient layout and routines that are insensitive to device parameters,such as the the SABRE (Li, Ding, and Xie 2019), and then score each mapping result using heuristic cost functions derived from calibration data. Finally, the circuits are remapped to the lowest error subgraph before execution. The results demonstrate that using better qubit selection can recover, on average, nearly 40% of missing fidelity.

While all of the aforementioned methods have produced notable outcomes. they have primarily focused on circuit mapping for fixed processor topologies. None of these previous studies have specifically customized the processor to the algorithm in order to achieve optimal performance.

A.2 More Background

Qubits: In the field of quantum computing, the qubit, or quantum bit, functions as the counterpart to the classical bit. Diverging from the classical bit's limitation to representing solely '1' or '0', a qubit can simultaneously exist in a coher-

ent superposition of both states. This feature establishes it as a two-state quantum system that encapsulates the distinctive attributes of quantum mechanics (Nielsen and Chuang 2010). An exemplary illustration of this concept is the electron spin, characterized by its two potential states: spin-up and spin-down.

Quantum Gates: Quantum computing operations fundamentally rely on two primary categories of quantum gates. The first category includes single-qubit gates, which are unitary operations that can be conceptualized as rotations around the axis of the Bloch sphere, representing the state space of a single qubit (Nielsen and Chuang 2010). The second category consists of multi-qubit gates, which are crucial for executing more complex quantum computations. Notably, complex quantum operations can be decomposed into sequences of single-qubit gates—namely, H, S, T gates—and the two-qubit Controlled NOT (CNOT) gate. The CNOT gate, due to its critical role, will be the primary focus of this discussion. It operates on two qubits, designated as the control and the target. The principle of operation for the CNOT gate is simple: if the control qubit is in the state '1', it flips the state of the target qubit; if the control qubit is '0', the target qubit remains unchanged.

Quantum Circuit: A quantum circuit consists of a collection of qubits and a sequence of operations applied to these qubits. The representation of quantum circuits can be achieved through several methods. One such method employs the quantum assembly language known as OpenQASM, introduced by IBM (Cross et al. 2022). Alternatively, circuit diagrams can be used, where qubits are depicted as horizontal lines and quantum operations are illustrated by various blocks placed along these lines.

A.3 Comparison with Tket Compiler

Qtailor utilizes *Qiskit* to assess circuits depth and gate count. However, it is important to note that *Qtailor* is not limited to a specific compiler; rather, it serves as a framework that is compatible with multiple compilers. The QPU topology suggested by *Qtailor* is generic. To illustrate this flexibility, we performed a comparative analysis with *Tket* (Sivarajah et al. 2021). Aside from switching compilation platforms from the *Qiskit* to *Tket*, no further modifications were necessary. The topology used in the experiments remained consistent with those in Section 4, specifically a fixed grid-like topology and the suggested topology from the RL module. Consequently, there is no requirement to retrain the RL model when changing compilers.

Qtailor achieves a maximum circuit depth reduction of 39% compared to Tekt, as illustrated in Figure 11. The results indicate that employing *Qtailor*'s suggested topology is more effective for reducing circuit depth, and highlights Qtailor's compatibility with various compilers.

A.4 Graph Isomorphic: an Example

Figure 12 illustrates the concept of isomorphic and non-isomorphic graphs. Isomorphism refers to the property of a graph where a single graph can exist in multiple forms. In other words, two distinct graphs can possess identical numbers of edges, vertices, and edge connectivity. Such graphs

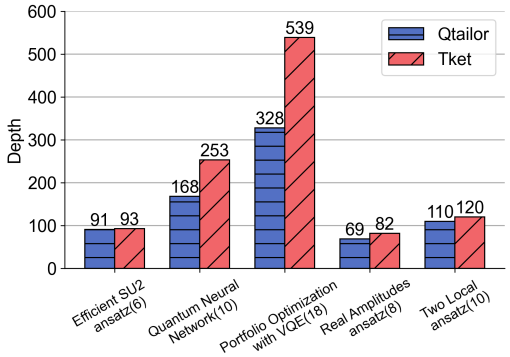


Figure 11: Comparing with the *TKet* compiler, where the horizontal axis represents the types of circuits, with the number of bits indicated in parentheses.

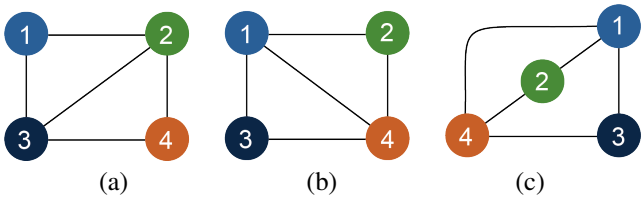


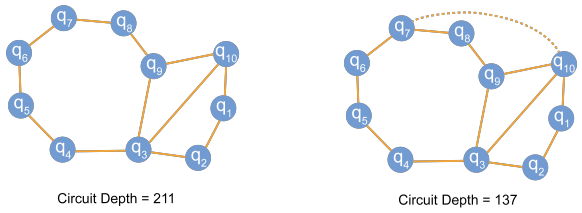
Figure 12: A Example of Non-Isomorphism and Isomorphism. Figure (a) and Figure (b) are non-isomorphic because they possess different edges (specifically, edge $\langle 1, 3\rangle$ and $\langle 1, 4\rangle$). On the other hand, Figure (b) and Figure (c) are isomorphic. Despite their dissimilar forms, they meet the aforementioned five conditions that have been defined.

are referred to as isomorphic graphs. For two graphs to be considered isomorphic, they must satisfy the following conditions: (a) Graphs must have the same number of vertices and edges, and their degree sequences must match. (b) If one graph forms a cycle of length k using vertices $v_1, v_2, v_3, \ldots, v_k$, the other graph must also form the same cycle of length k using vertices $v_1, v_2, v_3, \ldots, v_k$. (c) The adjacent matrices of both the graphs are the same.

In our experiments, we found that the ones that have an effect on the depth of the circuit are non-isomorphic graphs because the non-isomorphic graphs have different adjacency matrices, which means that the connectivity of the nodes in the graphs is different, and Figure 13 shows the effect of two non-isomorphic graphs with very small differences on the depth of the circuit.

A.5 Implementation Details

The benchmark circuits utilized in this study are sourced from the Munich Quantum Toolkit Benchmark Library (MQT Bench) (Quetschlich, Burgholzer, and Wille 2023). These circuits are stored in a .txt file in Qasm (Cross et al. 2022) format. A brief description of the circuits can be found on the MQT Bench website (MQT Bench 2023). All experiments were conducted on the Ubuntu 23.04 operating system using an Intel Xeon CPU (3.10GHz). The reinforcement learning model was implemented in Python 3.10 with



(a) (b)

Figure 13: Mapped Circuits Depth on Non-Isomorphism Graphs: Figure (b) exhibits an additional edge (represented by a dashed line) in comparison to Figure (a). Upon mapping Quantum Fourier Transformation circuits onto the graph, the depths of the two figures are determined to be 211 and 137, respectively. This signifies a notable reduction in depth of 35.07% when compared to the former configuration.

Hyper-parameters	**Values**
Critic network	256, 256
Policy network	256, 256
Discount factor	0.99
Batch size	4000
SGD Minibatch size	128
Learning rate	0.00005
Optimizer	SGD
Use KL loss	True

Table 2: Hyper-parameters for RR-PPO

the RLlib framework(Liang et al. 2018), which is a scalable and flexible reinforcement learning library built on the Ray distributed execution framework Detailed model hyper-parameters are listed in Table 2. Since Most existing QPU exhibit maximum degree ranging from 2 to 4, represented by structures like linear grids and hexagons, we set the maximum degree as 4.

A.6 More Details of Force-Directed Grid Layout

In Section 3, it was previously mentioned that the vertex moves in the direction of the resultant force, which represents the combined effect of all the component forces acting on an object. Component forces refer to the individual forces that contribute to the resultant force. It is important to note that the vertex moves incrementally, changing its forces in accordance with its position. The new forces are then re-calculated, taking into account the small change in position. Subsequently, the vertex moves again based on the updated combined forces. This iterative process is repeated several hundred times until the forces acting on the entire system reach a state of equilibrium.

A.7 Recommended Topologies for Circuits

In this section, we present the recommended topology for the Amplitude Estimation circuit with 40 qubits using *Qtailor*. We assign serial numbers to the qubits in the circuit, where the first qubit is denoted as 1. Initially, we illustrate

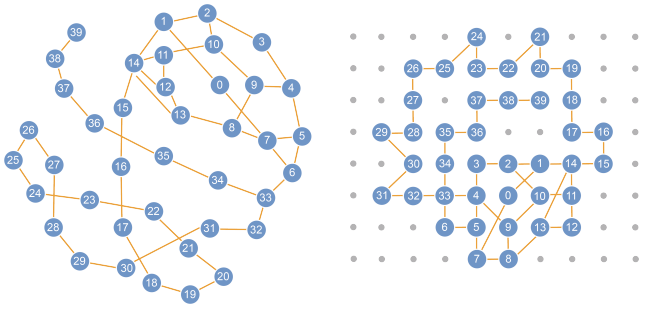
Algorithm 1: Reward-Replay Proximal Policy Optimization (PPO) Algorithm

Require: Initial policy parameters θ_0, initial value function parameters ϕ_0

for each $iteration$ **do**
  Collect Trajectories(Algorithm 2) $\mathcal{T}_{\theta_{old}}$ using policy $\pi_{\theta_{old}}$
  Compute rewards-to-go $\hat{R}_t$ and advantage estimates
  for each $epoch$ **do**
    for minibatch of size M sampled from $\mathcal{T}_{\theta_{old}}$ **do**
      Compute ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
      Compute clipped surrogate objective using ratio $r_t(\theta)$:
      Update policy parameters θ by maximizing $L(\theta)$ via stochastic gradient ascent
      Update value function parameters ϕ by minimizing the value function loss:
    end for
  end for
  $\theta_{old} \leftarrow \theta$
end for

Algorithm 2: Collect Trajectories with Reward-Repaly

for each episode until batch is full **do**
  Reset environment and get initial state s_0
  Initialize trajectory $\tau = \emptyset$
  while episode not done **do**
    Select action a_t based on policy $\pi_{\theta_{old}}(a_t|s_t)$
    if The reward r_{a_t} corresponding to the a_t is available in the memory **then**
      set reward $r_t = r_{a_t}$
      if $threshold_{a_t} <= 0$ **then**
        clear the pair $\langle r_{a_t}, a_t >\rangle$ in memory
      end if
    else
      Execute the action a_t and run a evaluation on circuit
      Use the reward function to calculate the reward value and assign it to r_t
      Append (a_t, r_t) to memory
      Set $threshold_{a_t}$ to a positive integer
    end if
    Update state $s_t \leftarrow s_{t+1}$
    Append (s_t, a_t, r_t, s_{t+1}) to τ
  end while
  Add trajectory $\tau_{\theta_{old}}$ to the batch of trajectories
end for



(a) (b)

Figure 14: Topology recommend for 40-bit Amplitude_Estimation circuits by RR-PPO model and a force-gird layout for this topology.

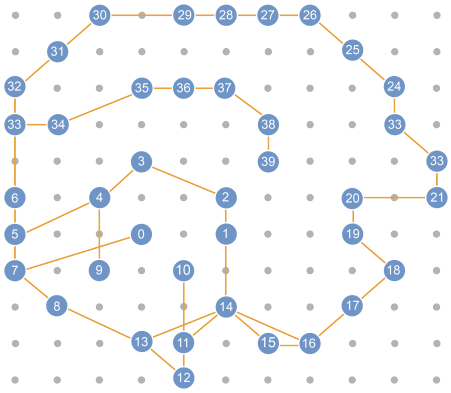


Figure 15: A sparse force-direct gird layout for 40-bit Amplitude_Estimation circuits

the topology suggested by RR-PPO (refer to Figure 14(a)). Subsequently, the qubits are precisely arranged on the grid using the force-grid layout algorithm (refer to Figure 14(b)). The force-directed grid layout allows for spacing the qubits more sparsely, reducing potential interference, as shown in Figure 15. This can be easily achieved by modifying the coefficient of the repulsion force.

Acknowledgments

H.-L.H. acknowledges support from the National Natural Science Foundation of China (Grant No. 12274464), and Natural Science Foundation of Henan (Grant No. 242300421049). We gratefully acknowledge Hefei Advanced Computing Center for hardware support with the numerical experiments.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J. C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F. G. S. L.; Buell, D. A.; Burkett, B.; Chen, Y.; Chen, Z.; Chiaro, B.; Collins, R.; Courtney, W.; Dunsworth, A.; Farhi, E.; Foxen, B.; Fowler, A.; Gidney, C.; Giustina, M.; Graff, R.; Guerin, K.; Habegger, S.; Harrigan, M. P.; Hartmann, M. J.; Ho, A.; Hoffmann, M.; Huang, T.; Humble, T. S.; Isakov, S. V.; Jeffrey, E.; Jiang, Z.; Kafri, D.; Kechedzhi, K.; Kelly, J.; Klimov, P. V.; Knysh, S.; Korotkov, A.; Kostritsa, F.; Landhuis, D.; Lindmark, M.; Lucero, E.; Lyakh, D.; Mandrà, S.; McClean, J. R.; McEwen, M.; Megrant, A.; Mi, X.; Michielsen, K.; Mohseni, M.; Mutus, J.; Naaman, O.; Neeley, M.; Neill, C.; Niu, M. Y.; Ostby, E.; Petukhov, A.; Platt, J. C.; Quintana, C.; Rieffel, E. G.; Roushan, P.; Rubin, N. C.; Sank, D.; Satzinger, K. J.; Smelyanskiy, V.; Sung, K. J.; Trevithick, M. D.; Vainsencher, A.; Villalonga, B.; White, T.; Yao, Z. J.; Yeh, P.; Zalcman, A.; Neven, H.; and Martinis, J. M. 2019. Quantum Supremacy Using a Programmable Superconducting Processor. *Nature*, 574(7779): 505–510.

Bauer, B.; Bravyi, S.; Motta, M.; and Chan, G. K.-L. 2020. Quantum Algorithms for Quantum Chemistry and Quantum Materials Science. *Chemical Reviews*, 120(22): 12685–12717.

Beer, K.; Bondarenko, D.; Farrelly, T.; Osborne, T. J.; Salzmann, R.; Scheiermann, D.; and Wolf, R. 2020. Training Deep Quantum Neural Networks. *Nature Communications*, 11(1): 808.

Bhattacharjee, D.; and Chattopadhyay, A. 2017. Depth-Optimal Quantum Circuit Placement for Arbitrary Topologies. arxiv:1703.08540.

Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; and Lloyd, S. 2017. Quantum Machine Learning. *Nature*, 549(7671): 195–202.

Boixo, S.; Isakov, S. V.; Smelyanskiy, V. N.; Babbush, R.; Ding, N.; Jiang, Z.; Bremner, M. J.; Martinis, J. M.; and Neven, H. 2018. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6): 595–600.

Burgholzer, L.; Schneider, S.; and Wille, R. 2022. Limiting the Search Space in Optimal Quantum Circuit Mapping. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 466–471. Taipei, Taiwan: IEEE. ISBN 978-1-66542-135-5.

Cao, Y.; Romero, J.; Olson, J. P.; Degroote, M.; Johnson, P. D.; Kieferová, M.; Kivlichan, I. D.; Menke, T.; Peropadre, B.; Sawaya, N. P. D.; Sim, S.; Veis, L.; and Aspuru-Guzik, A. 2019. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119(19): 10856–10915.

Contributors and IBM. 2024. Qiskit: an open-source SDK for working with quantum computers at the level of extended quantum circuits, operators, and primitives. https://github.com/Qiskit/qiskit.

Cross, A. W.; Javadi-Abhari, A.; Alexander, T.; de Beaudrap, N.; Bishop, L. S.; Heidel, S.; Ryan, C. A.; Sivarajah, P.; Smolin, J.; Gambetta, J. M.; and Johnson, B. R. 2022. OpenQASM 3: A Broader and Deeper Quantum Assembly Language. *ACM Transactions on Quantum Computing*, 3(3): 1–50.

de Moura, L.; and Bjørner, N. 2008. Z3: An Efficient SMT Solver. In Ramakrishnan, C. R.; and Rehof, J., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-78800-3.

Deutsch, D.; and Jozsa, R. 1997. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907): 553–558.

Ding, C.; Bao, T.-Y.; and Huang, H.-L. 2022. Quantum-Inspired Support Vector Machine. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12): 7210–7222.

Ding, Y.; Gokhale, P.; Lin, S. F.; Rines, R.; Propson, T.; and Chong, F. T. 2020. Systematic Crosstalk Mitigation for Superconducting Qubits via Frequency-Aware Compilation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 201–214.

François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M. G.; and Pineau, J. 2018.

Fu, X.; Rol, M. A.; Bultink, C. C.; van Someren, J.; Khammassi, N.; Ashraf, I.; Vermeulen, R. F. L.; de Sterke, J. C.; Vlothuizen, W. J.; Schouten, R. N.; Almudever, C. G.; DiCarlo, L.; and Bertels, K. 2017. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, 813–825. New York, NY, USA: Association for Computing Machinery. ISBN 9781450349529.

Huang, H.-L.; Du, Y.; Gong, M.; Zhao, Y.; Wu, Y.; Wang, C.; Li, S.; Liang, F.; Lin, J.; Xu, Y.; Yang, R.; Liu, T.; Hsieh, M.-H.; Deng, H.; Rong, H.; Peng, C.-Z.; Lu, C.-Y.; Chen, Y.-A.; Tao, D.; Zhu, X.; and Pan, J.-W. 2021. Experimental Quantum Generative Adversarial Networks for Image Generation. *Phys. Rev. Appl.*, 16: 024051.

Huang, H.-L.; Wu, D.; Fan, D.; and Zhu, X. 2020. Superconducting Quantum Computing: A Review. *Science China Information Sciences*, 63(8): 180501.

Huang, H.-L.; Xu, X.-Y.; Guo, C.; Tian, G.; Wei, S.-J.; Sun, X.; Bao, W.-S.; and Long, G.-L. 2023. Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation. *Science China Physics, Mechanics & Astronomy*, 66(5): 250302.

IBM. 2022. CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer.

Kobourov, S. G. 2012. Spring Embedders and Force Directed Graph Drawing Algorithms. *CoRR*, abs/1201.3011.

Li, G.; Ding, Y.; and Xie, Y. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. ASPLOS '19, 1001–1014. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362405.

Li, S.; Nguyen, K. D.; Clare, Z.; and Feng, Y. 2023. Single-Qubit Gates Matter for Optimising Quantum Circuit Depth in Qubit Mapping. *arXiv e-prints*, arXiv:2308.00876.

Li, S.; Zhou, X.; and Feng, Y. 2021. Qubit Mapping Based on Subgraph Isomorphism and Filtered Depth-Limited Search. *IEEE Transactions on Computers*, 70(11): 1777–1788.

Liang, E.; Liaw, R.; Moritz, P.; Nishihara, R.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. arxiv:1712.09381.

Liu, J.; Li, P.; and Zhou, H. 2022. Not All SWAPs Have the Same Cost A Case for Optimization-Aware Qubit Routing. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 709–725. IEEE. ISBN 978-1-66542-027-3.

Liu, J.; Lim, K. H.; Wood, K. L.; Huang, W.; Guo, C.; and Huang, H.-L. 2021. Hybrid Quantum-Classical Convolutional Neural Networks. *Science China Physics, Mechanics & Astronomy*, 64(9): 290311.

Lye, A.; Wille, R.; and Drechsler, R. 2015. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *The 20th Asia and South Pacific Design Automation Conference*, 178–183.

Moura, L.; and Bjørner, N. 2009. *Satisfiability Modulo Theories: An Appetizer*, 23–36. Berlin, Heidelberg: Springer-Verlag. ISBN 9783642104510.

MQT Bench. 2023. Brief Description of Benchmarks. https://www.cda.cit.tum.de/mqtbench/benchmark_description.

Murali, P.; Baker, J. M.; Javadi-Abhari, A.; Chong, F. T.; and Martonosi, M. 2019. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 1015–1029. Providence RI USA: ACM. ISBN 978-1-4503-6240-5.

Nannicini, G.; Bishop, L. S.; Günlük, O.; and Jurcevic, P. 2022. Optimal Qubit Assignment and Routing via Integer Programming. *ACM Transactions on Quantum Computing*, 4(1).

Nation, P. D.; and Treinish, M. 2023. Suppressing Quantum Circuit Errors Due to System Variability. *PRX Quantum*, 4(1): 010327.

Nielsen, M. A.; and Chuang, I. L. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

Nishio, S.; Pan, Y.; Satoh, T.; Amano, H.; and Van Meter, R. 2020. Extracting Success from IBM's 20-Qubit Machines Using Error-Aware Compilation. *ACM Journal on Emerging Technologies in Computing Systems*, 16(3): 1–25.

Oddi, A.; and Rasconi, R. 2018. Greedy Randomized Search for Scalable Compilation of Quantum Circuits. In van Hoeve, W.-J., ed., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 446–461.

Cham: Springer International Publishing. ISBN 978-3-319-93031-2.

Quetschlich, N.; Burgholzer, L.; and Wille, R. 2023. MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing. *Quantum*.

Sarovar, M.; Proctor, T.; Rudinger, K.; Young, K.; Nielsen, E.; and Blume-Kohout, R. 2020. Detecting Crosstalk Errors in Quantum Information Processors. *Quantum*, 4: 321.

Schlosshauer, M. 2019. Quantum decoherence. *Physics Reports*, 831: 1–57.

Schrijver, A. 1986. Theory of linear and integer programming. In *Wiley-Interscience series in discrete mathematics and optimization*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arxiv:1707.06347.

Shor, P. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134. IEEE Comput. Soc. Press. ISBN 978-0-8186-6580-6.

Shor, P. W. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5): 1484–1509.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the Game of Go without Human Knowledge. *Nature*, 550(7676): 354–359.

Siraichi, M. Y.; Santos, V. F. D.; Collange, C.; and Pereira, F. M. Q. 2018. Qubit Allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, 113–125. ACM. ISBN 978-1-4503-5617-6.

Siraichi, M. Y.; Santos, V. F. D.; Collange, C.; and Pereira, F. M. Q. 2019. Qubit Allocation as a Combination of Subgraph Isomorphism and Token Swapping. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA): 1–29.

Sivarajah, S.; Dilkes, S.; Cowtan, A.; Simmons, W.; Edgington, A.; and Duncan, R. 2021. Tket : A Retargetable Compiler for NISQ Devices. 6(1): 014003.

Tannu, S. S.; and Qureshi, M. K. 2019. Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 987–999. Providence RI USA: ACM. ISBN 978-1-4503-6240-5.

Venturelli, D.; Do, M.; Rieffel, E.; and Frank, J. 2017. Temporal Planning for Compilation of Quantum Approximate Optimization Circuits. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4440–4446.

Venturelli, D.; Do, M.; Rieffel, E.; and Frank, J. 2018. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2): 025004.

Wille, R.; Burgholzer, L.; and Zulehner, A. 2019. Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*, 1–6. ACM. ISBN 978-1-4503-6725-7.

Wu, Y.; Bao, W.; Cao, S.; Chen, F.; Chen, M.-C.; Chen, X.; Chung, T. H.; Deng, H.; Du, Y.; Fan, D.; Gong, M.; Guo, C.; Guo, C.; Guo, S.; Han, L.-F.; Hong, L.; Huang, H.; Huo, Y.; Li, L.; Li, N.; Li, S.; Li, Y. Y.; Liang, F.; Lin, C.; Lin, J.; Qian, H.; Qiao, D.; Rong, H.; Su, H.-Y.; Sun, L.; Wang, L.; Wang, S.; Wu, D.; Xu, Y.; Yan, K.; Yang, W.; Yang, Y.; Ye, Y.; Yin, J. H.; Ying, C.; Yu, J.; Zha, C.; Zhang, C.; Zhang, H.; Zhang, K.; Zhang, Y.; Zhao, H.; Zhao, Y.-W.; Zhou, L.; Zhu, Q.; Lu, C.; Peng, C.-Z.; Zhu, X.; and Pan, J.-W. 2021. Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Physical review letters*, 127 18: 180501.

Zeng, W.; and Church, R. L. 2009. Finding shortest paths on real road networks: the case for A*. *Int. J. Geogr. Inf. Sci.*, 23(4): 531–543.

Zhang, C.; Chen, Y.; Jin, Y.; Ahn, W.; Zhang, Y.; and Zhang, E. Z. 2020. A Depth-Aware Swap Insertion Scheme for the Qubit Mapping Problem. arxiv:2002.07289.

Zhang, C.; Hayes, A. B.; Qiu, L.; Jin, Y.; Chen, Y.; and Zhang, E. Z. 2021. Time-optimal Qubit mapping. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, 360–374. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383172.

Zhang, H.; Wu, A.; Wang, Y.; Li, G.; Shapourian, H.; Shabani, A.; and Ding, Y. 2023. OneQ: A Compilation Framework for Photonic One-Way Quantum Computation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, 1–14. New York, NY, USA: Association for Computing Machinery. ISBN 9798400700958.

Zhou, X.; Feng, Y.; and Li, S. 2022. Quantum Circuit Transformation: A Monte Carlo Tree Search Framework. *ACM Trans. Des. Autom. Electron. Syst.*, 27(6).

Zhou, X.; Li, S.; and Feng, Y. 2020. Quantum Circuit Transformation Based on Simulated Annealing and Heuristic Search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12): 4683–4694.

Zhu, Q.; Cao, S.; Chen, F.; Chen, M.-C.; Chen, X.; Chung, T.-H.; Deng, H.; Du, Y.; Fan, D.; Gong, M.; Guo, C.; Guo, C.; Guo, S.; Han, L.; Hong, L.; Huang, H.-L.; Huo, Y.-H.; Li, L.; Li, N.; Li, S.; Li, Y.; Liang, F.; Lin, C.; Lin, J.; Qian, H.; Qiao, D.; Rong, H.; Su, H.; Sun, L.; Wang, L.; Wang, S.; Wu, D.; Wu, Y.; Xu, Y.; Yan, K.; Yang, W.; Yang, Y.; Ye, Y.; Yin, J.; Ying, C.; Yu, J.; Zha, C.; Zhang, C.; Zhang, H.; Zhang, K.; Zhang, Y.; Zhao, H.; Zhao, Y.; Zhou, L.; Lu, C.-Y.; Peng, C.-Z.; Zhu, X.; and Pan, J.-W. 2022. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Science Bulletin*, 67(3): 240–245.

Zulehner, A.; Paler, A.; and Wille, R. 2018. Efficient mapping of quantum circuits to the IBM QX architectures. In

2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), 1135–1138.