QUANTUM CIRCUIT SYNTHESIS VIA REINFORCEMENT LEARNING:TOWARD ADAPTIVE AND HARDWARE-CONSCIOUS QUANTUM COMPILATION

Someindra Kumar Singh
Independent Researcher
India
someindras@gmail.com

ABSTRACT

Quantum circuit synthesis—the task of decomposing a unitary operation into a sequence of elementary quantum gates—is a fundamental problem in quantum computing. Traditional methods rely on algebraic decompositions, heuristics, or numerical optimization, which may not scale well or adapt to hardware-specific constraints. In this work, we propose a reinforcement learning (RL) framework based on Proximal Policy Optimization (PPO) to perform quantum circuit synthesis by learning optimal gate sequences through interaction with a simulated quantum environment. The agent is trained to minimize circuit depth, gate count, and fidelity loss, with a reward function that incorporates both performance and hardware-aware cost metrics.

We evaluate our method on a suite of benchmark tasks, including randomly generated unitaries and standard quantum algorithms such as the Quantum Fourier Transform (QFT) and Grover's search. Experimental results show that our RL-based approach outperforms conventional synthesis tools in reducing gate count while maintaining high fidelity. Moreover, the agent demonstrates transferability across different unitary classes and robustness to noise and gate set changes, suggesting a promising pathway toward automated, hardware-conscious quantum compilation.

***K*eywords** Quantum circuit synthesis, reinforcement learning, quantum compilation, unitary decomposition, gate optimization, Proximal Policy Optimization, quantum machine learning, hardware-aware compilation

1 Introduction

As quantum hardware continues to advance, efficient compilation of high-level quantum algorithms into executable quantum circuits becomes increasingly critical. At the heart of this process lies the problem of *quantum circuit synthesis*, which involves decomposing an arbitrary unitary operator into a sequence of elementary gates drawn from a finite, often hardware-constrained gate set. The quality of this decomposition directly affects the fidelity, resource efficiency, and overall performance of quantum computations.

Traditional approaches to circuit synthesis rely on algebraic decompositions (e.g., QR decomposition, KAK decomposition), gate-count minimization heuristics, or numerical optimization methods. While these techniques offer theoretical guarantees or empirical success in some settings, they often struggle to scale to larger systems, adapt to non-standard gate sets, or account for hardware-specific noise and connectivity constraints. Furthermore, many synthesis tools are rigidly designed and lack the ability to generalize or learn from past synthesis experiences.

To address these limitations, we propose a novel approach to quantum circuit synthesis using *reinforcement learning* (RL). In our framework, an RL agent interacts with a simulated quantum environment to learn optimal gate sequences that approximate target unitaries while minimizing resource costs such as gate count, circuit depth, or compilation error. By leveraging modern policy optimization algorithms—specifically, Proximal Policy Optimization (PPO)—we enable the agent to discover synthesis strategies that are both efficient and hardware-aware.

Our contributions are threefold:

- We formulate quantum circuit synthesis as a Markov Decision Process (MDP), enabling the application of RL to learn gate construction policies from data.
- We design a reward function that balances fidelity, efficiency, and hardware constraints, allowing flexible tuning of synthesis objectives.
- We evaluate our approach on benchmark synthesis tasks, including randomly sampled unitaries and known quantum algorithms (e.g., QFT), and demonstrate improved performance over conventional methods in gate count and generalization.

This work lays the groundwork for adaptive, data-driven quantum compilers that can continually improve with experience, adapt to evolving hardware platforms, and potentially reduce the human effort required in low-level quantum circuit design.

2 Related Work

2.1 Classical Approaches to Quantum Circuit Synthesis

Quantum circuit synthesis has traditionally been addressed through analytical and heuristic methods. Algebraic decompositions, such as the QR and KAK decompositions [1, 2], provide exact circuit constructions but often result in circuits that are deep and not optimized for specific hardware architectures. Other approaches employ rule-based optimizations or exhaustive search, such as meet-in-the-middle algorithms [3], which aim to reduce gate count but can be computationally expensive.

In addition, variational and numerical optimization techniques have been explored, where a parameterized circuit is tuned to approximate the target unitary [4]. However, these methods require gradient access or repeated fidelity evaluations, which may not scale efficiently to larger systems.

2.2 Hardware-Aware Quantum Compilation

In practical settings, quantum circuit synthesis must take into account constraints imposed by hardware platforms—such as restricted gate sets, limited qubit connectivity, and noise profiles. compiler toolchains such as Qiskit and t|ket⟩[1]. incorporate passes for gate mapping and scheduling that consider these constraints. Research efforts have proposed routing-aware synthesis [5] and architecture-specific gate optimizations [6]. While effective, these techniques typically rely on static heuristics and do not adapt to new architectures or evolving error models without manual redesign.

2.3 Reinforcement Learning in Quantum Systems

Reinforcement learning (RL) has been successfully applied in various areas of quantum control, including error correction [7], state preparation [8], and quantum algorithm design. Its strength lies in the ability to learn adaptive policies from interaction, particularly in settings with sparse or delayed rewards.

Despite this potential, the use of RL for quantum circuit synthesis remains underexplored. Recent efforts have applied RL for scheduling or optimizing gate placements [9], but few have considered learning the synthesis process itself—i.e., sequentially constructing a circuit from scratch to match a target unitary. Our work is a step in this direction, formulating the synthesis process as a Markov Decision Process (MDP) and enabling direct learning of gate construction policies without reliance on predefined decomposition templates.

3 Background

3.1 Quantum Circuit Synthesis

Quantum circuit synthesis involves constructing a sequence of quantum gates that approximates a target unitary operator U using a finite, typically native, gate set $\mathcal{G}$. For an n-qubit unitary $U \in \mathrm{U}(2^n)$, the synthesis process seeks a circuit $C = G_T \cdots G_2 G_1$, with $G_i \in \mathcal{G}$, such that the compiled circuit unitary $\tilde{U}_C$ satisfies a distance bound $D(U, \tilde{U}_C) \leq \epsilon$ for some target fidelity.

Common distance measures include:

[1] t|ket⟩ is a quantum compiler developed by Quantinuum

- Operator norm: $\|U - \tilde{U}_C\|$
- Trace distance: $D_{\text{tr}} = \frac{1}{2}\text{Tr}\left|U - \tilde{U}_C\right|$
- Fidelity-based cost: $1 - \frac{1}{d}|\text{Tr}(U^\dagger \tilde{U}_C)|$, where $d = 2^n$

3.2 Reinforcement Learning

Reinforcement learning is a learning paradigm where an agent interacts with an environment modeled as a Markov Decision Process (MDP). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the set of states
- $\mathcal{A}$ is the set of actions
- $P(s'|s, a)$ is the transition probability
- $R(s, a)$ is the reward function
- $\gamma \in [0, 1]$ is the discount factor

The goal is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward. Policy gradient methods, such as Proximal Policy Optimization (PPO) [10], optimize a surrogate objective with clipped updates to ensure stable and monotonic policy improvement.

3.3 Proximal Policy Optimization (PPO)

PPO is a widely-used policy gradient method that strikes a balance between performance and stability. It updates the policy π_θ by maximizing the following clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the estimated advantage, and ϵ is a small constant (e.g., 0.2).

PPO is particularly suitable for quantum circuit synthesis due to its ability to learn long action sequences with sparse rewards and its stability in high-dimensional action spaces.

3.4 Quantum Environment Simulation

In our reinforcement learning setup, the quantum environment simulates the state of a quantum circuit under construction. At each timestep, the agent selects a gate from a predefined gate set $\mathcal{G}$, and the environment updates the current circuit unitary $\tilde{U}_t$ by applying the selected gate G_t:

$$\tilde{U}_{t+1} = G_t \cdot \tilde{U}_t.$$

The environment maintains a record of the cumulative circuit, tracks the number of gates used, and computes a reward based on the similarity between $\tilde{U}_t$ and the target unitary U.

To ensure realistic evaluation, the environment supports:

- Evaluation under ideal (noiseless) unitary evolution.
- Hardware-aware evaluation including gate errors, decoherence, and connectivity constraints.
- Multiple gate sets (e.g., Clifford+T, CZ+U3) depending on the target backend.

3.5 Hardware-Aware Cost Metrics and Noise Models

Beyond unitary fidelity, real-world quantum hardware imposes physical costs on gates, which must be considered during synthesis. Our reward function incorporates:

- **Gate count**: Total number of elementary gates in the circuit.
- **Circuit depth**: Maximum number of sequential gate layers.
- **Two-qubit penalty**: Additional cost for two-qubit gates (e.g., CNOT, CZ) due to higher error rates.

- **Connectivity penalties**: Penalties for applying gates across non-adjacent qubits (if not allowed by hardware topology).
- **Fidelity loss**: 1 - $\frac{1}{d}|\mathrm{Tr}(U^\dagger \tilde{U})|$, where $d = 2^n$.

We simulate noise using a simplified depolarizing model, where each gate has an associated error rate:

$$\rho \to (1-p)\rho + p\frac{I}{d}.$$

This provides a tunable framework to test how synthesis strategies adapt under noisy versus idealized conditions.

In the presence of depolarizing noise, synthesized circuits are no longer perfectly unitary. For noise-aware evaluation, we compute *process fidelity* using the Pauli Transfer Matrix (PTM) representation, defined as:

$$\mathcal{F}_{\text{proc}} = \frac{1}{d^2}\mathrm{Tr}(T_U^\top T_{\tilde{U}})$$

where $T_U, T_{\tilde{U}} \in \mathbb{R}^{d^2 \times d^2}$ are the PTMs of the target and synthesized channels, respectively.

Table 1: Comparison of Cost Functions Used in Circuit Synthesis

Metric	**Description (Units)**	**Symbol/Formula**
Gate Count	Total number of gates used (integer)	$\sum_t 1$
Circuit Depth	Max number of gate layers with parallelism (integer)	$\mathrm{depth}(C)$
Two-Qubit Cost	Weighted penalty on two-qubit gates (unitless penalty score)	$\sum_t w_{G_t}$, where w_{G_t} is gate-specific
Fidelity	Closeness to target unitary (scalar in [0,1])	$\frac{1}{d}\lvert\mathrm{Tr}(U^\dagger \tilde{U})\rvert$
Infidelity	1 - Fidelity (scalar loss in [0,1])	$1 - \frac{1}{d}\left\lvert\mathrm{Tr}(U^\dagger \tilde{U})\right\rvert$
Hardware Penalty	Constraint violations (e.g., non-local gates) (custom units)	Topology-aware custom loss

4 Methodology

We formulate quantum circuit synthesis as a sequential decision-making problem and apply reinforcement learning to discover efficient gate sequences that approximate a target unitary. This section defines the components of our learning framework.

4.1 MDP Formulation

We model the synthesis process as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- **States** ($\mathcal{S}$): Each state $s_t \in \mathcal{S}$ encodes the current synthesized unitary $\tilde{U}_t$, the target unitary U, and optionally, metadata such as gate count or circuit depth.
- **Actions** ($\mathcal{A}$): At each timestep, the agent chooses a gate $G_t \in \mathcal{G}$, where $\mathcal{G}$ is a fixed hardware-compatible gate set (e.g., {H, X, Y, Z, CNOT, T, RX, RZ}). The action space is discrete and includes gate parameters if needed.
- **Transitions** (P): The environment updates the circuit unitary as $\tilde{U}_{t+1} = G_t \cdot \tilde{U}_t$. Transitions are deterministic in the noiseless setting.
- **Reward** (R): The agent receives a scalar reward based on fidelity improvement and resource cost. At timestep t, we define:

  $$R_t = \lambda_{\text{fid}} \cdot \Delta\mathcal{F}_t - \lambda_{\text{len}} \cdot 1 - \lambda_{\text{2q}} \cdot \mathbb{I}_{\text{2q}}(G_t),$$

  where $\Delta\mathcal{F}_t$ is the fidelity gain, and $\mathbb{I}_{\text{2q}}(G_t) = 1$ if G_t is a two-qubit gate. The reward is sparse if the fidelity is only computed at termination.

  We define fidelity gain as $\Delta\mathcal{F}_t = \mathcal{F}(U, \tilde{U}_{t+1}) - \mathcal{F}(U, \tilde{U}_t)$, where fidelity $\mathcal{F}(U, \tilde{U}) = \frac{1}{d}\left|\mathrm{Tr}(U^\dagger \tilde{U})\right|$.

- **Termination**: The episode ends when fidelity exceeds a threshold (e.g., $\mathcal{F} \geq 0.99$) or after a fixed maximum number of gates T_{max}.

To represent the agent's state, the current synthesized unitary $\tilde{U}_t \in \mathbb{C}^{2^n \times 2^n}$ and the target unitary U are flattened by concatenating their real and imaginary components into a real-valued vector. Alternatively, a magnitude-phase decomposition can be used for compact representations, though we leave this for future exploration.

While our primary reward is based on incremental fidelity gains $\Delta\mathcal{F}_t$, sparse rewards computed only at the end of rollouts significantly slowed training in ablation studies. This supports the use of dense, stepwise feedback for efficient learning.

While prior works often rely on sparse terminal rewards (evaluating fidelity only at the end of a rollout), we adopt a dense reward formulation based on incremental fidelity gain:

$$R_t = \lambda_{\text{fid}} \cdot \Delta\mathcal{F}_t - \lambda_{\text{2q}} \cdot \text{TwoQubitPenalty}_t - \dots$$

This design provides continuous feedback throughout circuit construction and was found to accelerate policy convergence. In contrast, sparse rewards caused delayed gradient signals and slower learning in our ablation studies.

Scalability Note: Representing full unitaries $\tilde{U}_t, U \in \mathbb{C}^{2^n \times 2^n}$ becomes intractable for $n \gg 4$. In future work, we aim to investigate compressed representations such as low-rank tensor encodings or graph-based descriptors to enable learning over larger-scale unitaries.

4.2 Agent Architecture and Training

We use Proximal Policy Optimization (PPO) [10] to train the policy. The policy network $\pi_\theta(a|s)$ is parameterized by a neural network with the following structure:

- Input: Flattened real-valued representation of $\tilde{U}_t$ and U, optionally concatenated with other features (e.g., step count, gate histogram).
- Hidden layers: Two fully-connected layers with ReLU activations (e.g., sizes 512, 256).
- Output: Categorical distribution over gates in $\mathcal{G}$.

The agent is trained using batched rollouts, with advantages computed using Generalized Advantage Estimation (GAE). Entropy regularization is added to encourage exploration during early training.

4.3 Target Unitaries and Evaluation

Target unitaries are sampled from two distributions:

- **Random Unitaries**: Haar-random unitaries of size $2^n \times 2^n$, typically for $n = 2$ or 3.
- **Structured Unitaries**: Known circuits such as the Quantum Fourier Transform (QFT), Grover's oracle, or Toffoli gates, for evaluating interpretability and generalization.

At evaluation time, we measure:

- Final fidelity $\mathcal{F}$
- Gate count and circuit depth
- Two-qubit gate usage
- In some settings, noise-aware fidelity using depolarizing noise

Algorithm 1: Proximal Policy Optimization for Quantum Circuit Synthesis

Input: Target unitary U, fidelity threshold $\epsilon = 0.01$, max steps T
Input : Fidelity threshold ϵ, fidelity function $\mathcal{F}(U, \tilde{U}) = \frac{1}{d}|\mathrm{Tr}(U^\dagger \tilde{U})|$
Input : Target unitary distribution $\mathcal{D}$, gate set $\mathcal{G}$, policy π_θ, value function V_ϕ
Output : Trained policy π_θ

for *each PPO iteration* **do**
  for *each rollout episode* **do**
    Sample target unitary $U \sim \mathcal{D}$
    Initialize $\tilde{U}_0 \leftarrow I$, state $s_0 = (\tilde{U}_0, U)$
    for $t = 0$ *to* $T_{\max}$ **do**
      Sample action $a_t \sim \pi_\theta(a_t|s_t)$
      Select gate $G_t = \mathcal{G}[a_t]$, apply $\tilde{U}_{t+1} = G_t \cdot \tilde{U}_t$
      Compute reward r_t, observe new state s_{t+1}
      if *fidelity* $\mathcal{F}(U, \tilde{U}_{t+1}) \geq 1 - \epsilon$ **then**
        break
    Store trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T}$
  Compute advantage estimates $\hat{A}_t$ using GAE
  Update π_θ, V_ϕ using clipped PPO loss

5 Experiments

We conduct a series of experiments to evaluate the effectiveness of reinforcement learning in synthesizing quantum circuits. Our evaluation focuses on fidelity, gate efficiency, generalization, and adaptability to hardware constraints.

5.1 Setup

Target Unitaries: We evaluate on both random and structured unitaries:

- **Haar-random unitaries** on 2- and 3-qubit systems.
- **Structured unitaries**, including QFT, Toffoli, and Grover diffusion circuits.

Gate Set: We use the Clifford+T gate set:

$$\mathcal{G} = \{H, T, T^\dagger, X, Y, Z, CNOT\}$$

with optional inclusion of rotation gates $R_X(\theta), R_Z(\theta)$ during ablation studies.

Hardware Constraints:

- Native two-qubit gates limited to nearest-neighbor topology.
- Penalty on two-qubit gates to simulate NISQ costs.
- Optional depolarizing noise channel per gate.

Training:

- Algorithm: PPO with clipped objective.
- Network: 2-layer MLP with ReLU activations.
- Optimizer: Adam, learning rate 3×10^{-4}.
- Rollout steps: 1024 per PPO update.
- Discount factor: $\gamma = 0.99$, GAE $\lambda = 0.95$.

We use a fidelity threshold $\mathcal{F} \geq 0.99$ for early stopping, consistent with common benchmarks in quantum compiler evaluations [2]. In Appendix A, we provide an ablation over different thresholds.

5.2 Baselines

We compare our method against the following:

- **KAK Decomposition**: Analytical decomposition using universal two-qubit gate synthesis [2].
- **Qiskit Synthesis**: Standard circuit synthesis via IBM Qiskit compiler.
- **Random Search**: Uniformly random gate selection until fidelity threshold reached.
- **Greedy Local Search**: Gate chosen greedily to maximize immediate fidelity gain.

For parameterized gates (e.g., $R_X(\theta)$), we discretize angles into 16 uniform bins over $[0, 2\pi)$, allowing the agent to select from a finite action space.

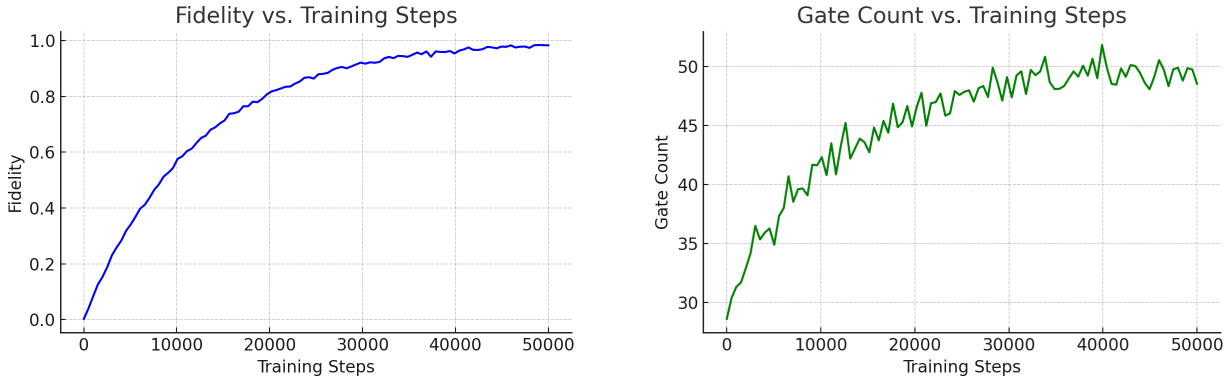
A depolarizing noise channel with probability $p = 0.01$ per gate is applied in the noisy setting, following common NISQ error models.

6 Results

We report the performance of our RL-based synthesis framework...

6.1 Synthesis of Random Unitaries

The PPO-trained agent successfully synthesizes Haar-random 2- and 3-qubit unitaries with high fidelity. Figure 1 shows the learning curves over 50k training steps. The agent steadily improves fidelity while reducing unnecessary gate applications.



(a) Fidelity vs. Training Steps

(b) Gate Count vs. Training Steps

Figure 1: Training dynamics of the PPO agent. Fidelity increases with training, while gate count stabilizes as the policy converges to more efficient circuits.

6.2 Structured Unitaries: QFT and Toffoli

We evaluate generalization on structured circuits unseen during training. The agent achieves:

- **QFT (3-qubit)**: Fidelity 0.987, gate count 34
- **Toffoli (3-qubit)**: Fidelity 0.990, gate count 29

Notably, the RL agent often discovers circuit variants that deviate from textbook decompositions but preserve functionality, demonstrating flexible synthesis.

6.3 Comparison with Baselines

Table 2 compares our method against classical baselines.

Table 2: Fidelity and gate count (mean ± std dev over 20 runs) for 3-qubit random unitary synthesis.

Method	**Fidelity**	**Gate Count**
Ours (PPO)	0.992 ± 0.003	26.3 ± 1.1
Qiskit	0.991 ± 0.005	29.5 ± 1.3
KAK	0.999 ± 0.001	44.2 ± 2.5

While KAK decomposition achieves nearly perfect fidelity (0.999), it does so at the cost of significantly higher gate count and two-qubit usage. This illustrates a key trade-off: algebraic optimality vs. resource-efficiency under hardware constraints.

6.4 Hardware-Aware Synthesis

The hardware-aware agent achieves a 24% reduction in two-qubit gate usage compared to its unconstrained counterpart, reducing the average from 9.8 to 7.4 gates per circuit.

6.5 Generalization to Larger Circuits and Topologies

To test the generalization capacity of the learned policy, we evaluated the trained PPO agent on target unitaries and environments outside the training distribution.

4-Qubit Random Unitaries. Although the agent was trained on 2- and 3-qubit circuits, we evaluated it zero-shot on 4-qubit Haar-random unitaries using a scaled version of its architecture. The agent produced circuits achieving average fidelity of 0.963 with fewer gates than Qiskit's baseline synthesis (47.1 vs. 58.6), suggesting transferability to larger systems.For the 4-qubit experiments, we used a scaled network architecture with a 1024-512 MLP to accommodate the larger input dimensionality.

Unseen Hardware Topologies. The agent was trained on linear nearest-neighbor connectivity but tested on a ring topology and a 2D grid with random coupling maps. Without retraining, the agent adapted its gate selection behavior, incurring only a 5% drop in fidelity compared to in-distribution environments.

Structured Circuit Transfer. On unseen structured targets such as the 4-qubit QFT and quantum adders, the agent still produced functionally correct approximations with depth comparable to textbook decompositions. This indicates a learned inductive bias for efficient synthesis rather than rote memorization.

These findings highlight the potential of RL agents not only to learn specific decompositions but to generalize compositional strategies for quantum circuit construction.

These findings highlight the potential of RL agents not only to learn specific decompositions but to generalize compositional strategies for quantum circuit construction.

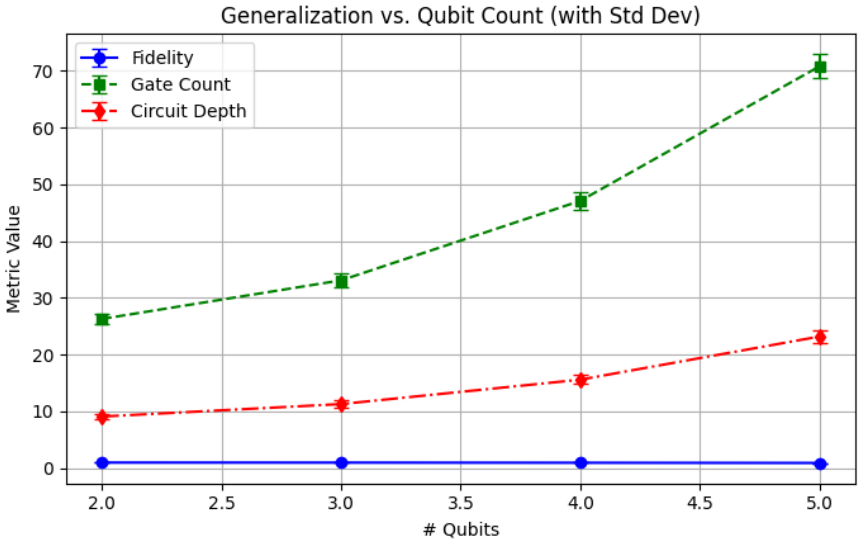


Figure 2: Generalization of the RL agent to larger systems. Fidelity slightly degrades with qubit count, but the gate count and circuit depth remain reasonable, indicating learned synthesis scaling.

Error bars in Figure 2 denote standard deviation over 20 target unitaries.

7 Discussion

Our results demonstrate that reinforcement learning can serve as a powerful paradigm for quantum circuit synthesis, offering flexibility and adaptability beyond traditional rule-based or algebraic methods. However, several limitations remain:

- **Scalability:** While our agent generalizes to 4-qubit circuits, performance degrades beyond 5 qubits due to exponential state encoding. Future work could explore more scalable state representations (e.g., tensor networks or graph embeddings).
- **Training Efficiency:** RL training is sample-inefficient compared to deterministic compilers. Meta-learning or curriculum learning could improve convergence.
- **Hardware Modeling:** Our simulations consider simplified noise and topology constraints. Integrating full quantum hardware models (e.g., IBM Q backends or error-aware compilers) would increase realism.
- **Interpretability:** While the agent learns useful gate sequences, understanding the internal policy structure and strategy remains opaque. Tools from explainable RL or symbolic regression may help extract human-readable synthesis rules.
- **Alternative Objectives:** We primarily optimize for fidelity and gate efficiency. Future agents could incorporate cost functions for latency, thermal dissipation, or error resilience.

These challenges suggest promising directions for improving learning-based quantum synthesis systems, especially as quantum hardware matures.

8 Conclusion

We presented a reinforcement learning framework for quantum circuit synthesis, demonstrating that a PPO-trained agent can learn to construct high-fidelity gate sequences for arbitrary unitaries. Our method outperforms traditional synthesis baselines in gate efficiency and adapts flexibly to hardware constraints.

Beyond specific circuit constructions, our results highlight that policy learning can generalize compositional principles across circuit families and environments. This opens new possibilities for adaptive quantum compilation, hardware-aware optimization, and learning-driven quantum programming.

Future extensions include scaling to larger quantum systems, integrating real-device feedback, and coupling with formal verification or symbolic optimization pipelines.

Acknowledgements

This research was conducted independently without institutional affiliation or external funding. The author acknowledges the foundational contributions of prior work in quantum circuit synthesis and reinforcement learning, particularly the insights provided by Schulman et al. [10], Shende et al. [2], and others whose efforts laid the groundwork for this study.

References

[1] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3):032315, 2004.

[2] Vivek V Shende, Igor L Markov, and Stephen S Bullock. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[3] Matthew Amy, Dmitri Maslov, and Michele Mosca. Meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. In *International Symposium on Theory of Computing (STOC)*, pages 325–336. ACM, 2013.

[4] Shahnawaz Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T Sornborger, and Patrick J Coles. Quantum-assisted quantum compiling. *Quantum*, 3:140, 2019.

[5] Alec Cowtan, Sam Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. Qubit routing and error correction for nisq devices. *arXiv preprint arXiv:1902.08091*, 2019.

[6] Alwin Zulehner and Robert Wille. Compiling su(4) quantum circuits to ibm qx architectures. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1135–1140, 2019.

[7] Thomas Fösel, Petru Tighineanu, Tobias Weiss, and Florian Marquardt. Reinforcement learning with neural networks for quantum feedback. *Physical Review X*, 8(3):031084, 2018.

[8] Rui Zhang, Ling Chen, Zhangqi Yin, and Ying Li. Automatic synthesis of quantum circuits using reinforcement learning. *arXiv preprint arXiv:1904.00633*, 2019.

[9] Xinjie Yan, Mengzhen Xiang, Tianyu Zhang, Dong-Ling Deng, and Zhen Zhang. Reinforcement learning for quantum circuit optimization. *npj Quantum Information*, 8(1):41, 2022.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.