Discovering Quantum Circuits for Logical State Preparation with Deep Reinforcement Learning

Remmy Zen[1] **Jan Olle**[1] **Matteo Puviani**[1] **Florian Marquardt**[1,2]
[1]Max Planck Institute for the Science of Light, Staudtstraße 2, 91058 Erlangen, Germany
[2]Friedrich-Alexander Universität Erlangen-Nürnberg, Staudtstraße 5, 91058 Erlangen, Germany
{remmy.zen,jan.olle,matteo.puviani,florian.marquardt}@mpl.mpg.de

Abstract

Quantum error correction (QEC) is important for the realization of fault-tolerant quantum computers. The first essential step of QEC is to encode the logical state into physical qubits. However, there is no unique recipe for finding a quantum circuit that encodes or prepares the logical state, especially for a given gate set and qubit connectivity. In this work, we use deep reinforcement learning to automatically discover quantum circuits to prepare the logical state of a QEC code given a gate set and qubit connectivity. We show that our method can prepare a logical state of up to 17 physical qubits code in fully connected qubits and up to 15 physical qubits code with IBM quantum devices gate set and connectivity with smaller circuit size than other methods.

1 Introduction

One of the key aspects in realizing large-scale fault-tolerant quantum computers is quantum error correction (QEC) [17]. The basic idea of QEC is to protect quantum information by encoding k logical qubits into $n > k$ noisy physical qubits in such a way that we can detect and correct errors without destroying the logical state $|\psi\rangle_L$. A QEC code determines how to encode and decode $|\psi\rangle_L$ and how to detect and correct errors. Once a code is chosen, the next step is to find a unitary U, represented as a quantum circuit, that prepares $|\psi\rangle_L$ of that code. Finding U is not trivial, especially with a given set of gates and qubit connectivity. The problem is often referred to in the literature as the *compilation* problem [13], and it is of great relevance in current noisy intermediate scale quantum (NISQ) devices where the gate set and qubit connectivity are restricted.

We focus on a special type of QEC code called the stabilizer code [9]. In this case, we can restrict U to a circuit that uses only Clifford gates (e.g., Hadamard H, phase S, and controlled not CNOT). The preparation of a state with a Clifford circuit has been studied extensively [2, 3, 7, 16]. Similar to our work, Xu et al. [23] used a parameterized circuit to compile $|\psi\rangle_L$ of codes with $n = 5$ and 7. However, in a parameterized circuit, one needs to provide a circuit ansatz, and the optimization process generally does not scale well because of the barren plateaus [14]. Therefore, there is still a need for a scalable compilation tool that discovers circuits and works autonomously without an ansatz.

Recently, machine learning, and in particular reinforcement learning (RL) [22], has proven to be a useful tool for solving various problems in quantum technologies [10]. In RL, an agent is trained to discover a policy (optimal set of actions) in an environment by maximizing cumulative rewards. In this paper, we propose to use RL to automatically discover quantum circuits that prepare a logical state $|\psi\rangle_L$ of a given QEC code, gate set, and qubit connectivity, as shown in Figure 1a. In our methods, we use discrete Clifford gates, which means that, compared to [23], our approach does not require an ansatz. RL has also been used for the preparation of random states of only up to 2 qubits [8, 15, 24] but not in the stabilizer domain. We show that RL can prepare $|\psi\rangle_L$ of up to $n = 17$

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

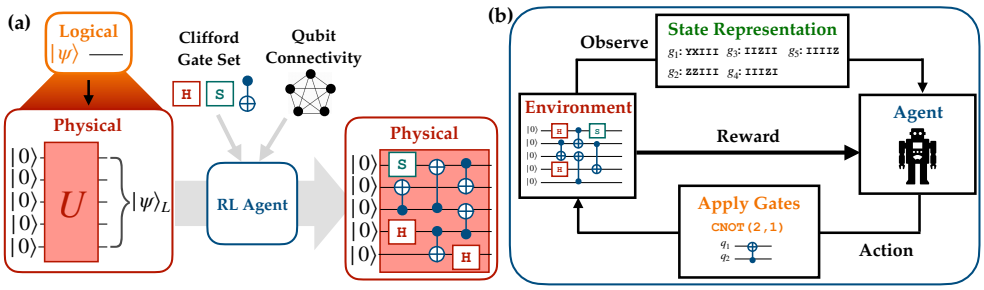

Figure 1: Overview of the preparation of logical states with reinforcement learning (RL). (a) The RL agent takes as input the target logical state $|\psi\rangle_L$ of a $[[n, k, d]]$ code, a Clifford gate set, and the qubit connectivity, and outputs a circuit U that prepares $|\psi\rangle_L$. (b) The RL framework in this paper. The RL environment is the circuit represented by the stabilizer tableau of the circuit's state and is observed by the RL agent. At each step, the agent applies a Clifford gate and receives a reward.

in fully connected qubits and up to $n = 15$ on IBM quantum device connectivity and gate set with smaller circuit sizes than other methods.

Background. In the following, we give a brief background on the stabilizer formalism. A QEC code that encodes k logical qubits into n physical qubits is commonly denoted as $[[n, k, d]]$, where d is the code distance, meaning that it can detect (correct) errors up to weight $d - 1$ ($\lfloor d/2 \rfloor$). We focus specifically on the stabilizer codes [9]. The main idea of the stabilizer formalism is to represent a state $|\psi\rangle$ as a set of Pauli operators O that stabilize the state $O|\psi\rangle = |\psi\rangle$.

The set of stabilizers S is a subset of the Pauli group of n qubits such that all elements of S commute with each other and $-I \notin S$. S can be written by specifying its set of generators, $S = \langle g_1, \ldots, g_{n-k} \rangle$, meaning that states in the code space are stabilized by any element of S. Within the code space, each code word can be transformed into one another using the logical operators Z_L^i and X_L^i commute with S and where $i = 1, ..., k$. For example, if $k = 1$, then $Z_L^1|0\rangle_L = |0\rangle_L$ and $X_L^1|0\rangle_L = |1\rangle_L$, so the choice of logical operators determines the logical state $|0\rangle_L$ and $|1\rangle_L$.

We are concerned only with Clifford circuits, where the state $|\psi\rangle = U|0 \ldots 0\rangle$ is always determined by the stabilizer canonical tableau [2]. In particular, the tableau of a logical state is fixed to contain the $n - k$ stabilizer generators and the k logical operators. A tableau can be represented as a $n \times (2n + 1)$ matrix of binary variables x_{ij}, z_{ij}, r_i for $i, j \in \{1, \ldots, n\}$. Each row i of the tableau $[x_{i1}, \ldots x_{in}, z_{i1}, \ldots, z_{in}, r_i]$ represents the Pauli string of the generators or logical operators, where $x_{ij}z_{ij}$ bits determine the j-th Pauli matrix, where 00, 01, 10, and 11 denote I, Z, X, and Y Pauli, respectively, and r_i denotes its sign (1 for negative and 0 for positive). The canonical tableau is obtained by applying Gaussian elimination to the tableau [2].

2 Reinforcement learning approach for logical state preparation

The goal of logical state preparation is to find a circuit U that prepares the target stabilizer state (as shown in Figure 1a). The input is the canonical tableau of the target logical state s_{target}, which consists of $n - k$ generators and k logical operators. Additionally, it takes as input the Clifford gate set and the qubit connectivity. Note that although in this work we focus on preparing a logical state $|\psi\rangle_L$ of a stabilizer code, our approach is general enough that it can be used to prepare arbitrary stabilizer states.

The RL framework is shown in Figure 1b. The environment of RL is the quantum circuit. We assume that all qubits in the circuit are initialized in the state $|0\rangle$. To represent the environment, one could use the state vector of size 2^n, as in [15, 24]. However, this representation is sparse and scales exponentially with n. Since we are focusing on stabilizer codes, we will instead use the current canonical stabilizer tableau of the circuit [2] to represent the state of the environment. This representation is denser and scales quadratically with n. This representation serves as input to the neural networks.

Based on the input, the RL agent needs to choose and apply an action. The action that the agent takes is to apply discrete Clifford gates to the circuit, which is determined by the given gate set and

qubit connectivity. Suppose we choose a gate set consisting of G_1 one-qubit gates and G_2 two-qubit gates with fully connected qubits. At each step, the agent must decide over $nG_1 + (n^2 - n)G_2$ possible actions. Assuming that the circuit has L gates, then the space of possible solutions grows exponentially as $L^{nG_1+(n^2-n)G_2}$, making search algorithms infeasible. After applying an action, the agent gets a reward that we will discuss next.

The training of the RL agent consists of updating the neural network parameters to maximize the cumulative reward. Therefore, designing a good reward function is an important aspect of training the RL agent. One could choose to reward based on state fidelity used in [15, 24] or to use the energy cost function used in [23]. However, we find that these reward functions rarely produce a useful reward signal (sparse reward problem [22]). Instead, we propose a new reward scheme based on the binary distance $d(s_t, s_{target})$ between the current canonical tableau of the circuit s_t and the target state s_{target}. Furthermore, we also use the reward shaping technique [22] by giving small intermediate rewards at each step. Thus, at each time step, we give the difference between the distance of the previous time step to the current time step (since we want to maximize the cumulative reward), which is formally given as $d(s_{t-1}, s_{target}) - d(s_t, s_{target})$.

3 Results

We divide the discussion of the results into two parts. First, we use fully connected qubits with H, S, and CNOT gates, which we will refer to as the *standard gate set*. Then, we show that RL can adapt to the qubit connectivity and gate sets of realistic NISQ hardware platforms. In this work, we choose to focus on IBM quantum devices. As an evaluation metric, we measure the *circuit size*, which corresponds to the number of gates in the circuit. The smaller the circuit size, the better.

We use PUREJAXRL library [12] for the implementation of the RL with Proximal Policy Optimization (PPO) algorithm [19], which is written in PYTHON with JAX [6] library to allow parallel training on Graphics Processing Units (GPU). This library allows us to rapidly train multiple agents in parallel. All experimental results shown below were performed on a single NVIDIA Quadro RTX 6000 GPU.

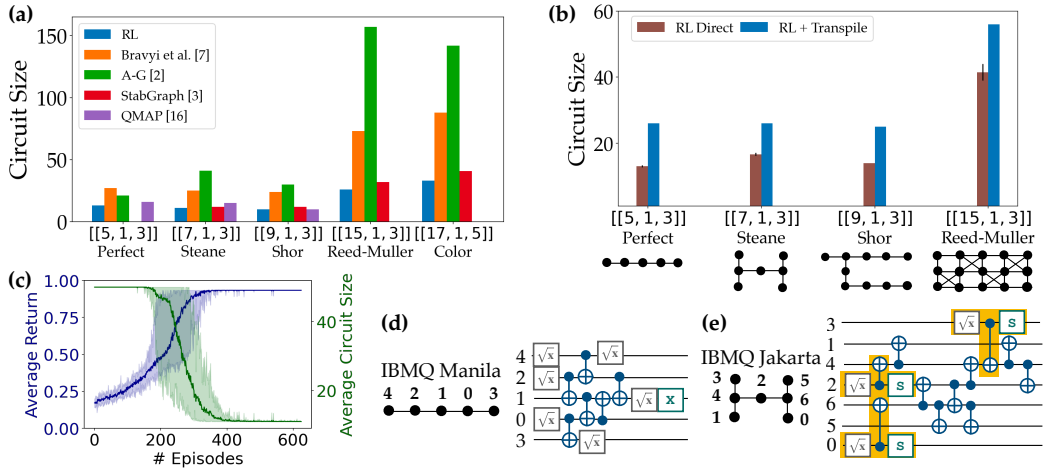

Figure 2: Results of RL-prepared logical state. (a) Circuit size of different logical state preparation methods for different QEC codes with fully connected qubits and H, S, and CNOT gates. (b) Comparison of circuit size between RL agent that incorporates the connectivity and gate set during training (RL Direct) compared to a RL-prepared circuit for fully connected qubits followed by QISKIT transpilation [1] (RL + Transpile) of different logical state preparation methods on IBM quantum devices qubit connectivity and gate set. (c) shows an example of the RL training progress for preparing $|0\rangle_L$ state of $[[7, 1, 3]]$ code. (d) and (e) show two examples of RL-prepared circuits for preparing $|0\rangle_L$ state of $[[5, 1, 3]]$ code in IBMQ Manila connectivity and $[[7, 1, 3]]$ code in IBMQ Jakarta connectivity, respectively. Since H is not in the gate set, the agent learns a new gate sequence $\sqrt{X}$, CNOT, S (in yellow) that is equivalent to H gate followed by CNOT.

State preparation on fully-connected qubits with standard gate set. We prepare the zero logical $|0\rangle_L$ state of the $[[5,1,3]]$ perfect code [11], $|0\rangle_L$ of $[[7,1,3]]$ Steane code [21], plus logical $|+\rangle_L$ of $[[9,1,3]]$ Shor code [20], $|0\rangle_L$ of $[[15,1,3]]$ quantum Reed-Muller code or 3D color code [4], and $|0\rangle_L$ of $[[17,1,5]]$ 2D color code [5]. We choose the logical operators to be $Z^{\otimes n}$, where n is the number of physical qubits.

We compare the RL method with four different Clifford circuit synthesis methods. Two of them are available in the QISKIT [18] library, which is based on the algorithm provided by Bravyi et al. [7] and Aaronson-Gottesman (A-G) [2]. We also compare with StabGraph [3] which works only for a specific type of codes called Calderbank-Shor-Steane (CSS) codes by converting it into a graph state and QMAP [16], which converts the problem into a boolean satisfiability problem (SAT) and solves it with SAT solver.

Figure 2a shows the comparison of the circuit size for the logical state preparation between different methods. We see that proposed RL method prepares a smaller circuit size compared to other methods. StabGraph [3] is specialized in preparing the logical state of a CSS code, therefore it does not work for preparing the $|0\rangle_L$ state of the $[[5,1,3]]$ code. QMAP [16] could not prepare the $|0\rangle_L$ state of the $[[15,1,3]]$ and $[[17,1,3]]$ code within the specified time of 24 hours. In terms of efficiency, training the RL agent takes about 1.8 minutes, 1.6 minutes, 2 minutes, 50 minutes, and 1.4 hours for the state preparation of $[[5,1,3]]$, $[[7,1,3]]$, $[[9,1,3]]$, $[[15,1,3]]$, and $[[17,1,5]]$ code, respectively, while Bravyi et al. [7] and A-G [2] can prepare the circuits in under 5 seconds. Even though RL is slower than other algorithms, we argue that the discovery of a state preparation circuit only needs to be done once so the time scales are completely acceptable and the resulting circuit size is lower.

Figure 2c shows how the average return and the average circuit size evolve during training for the $|0\rangle_L$ state preparation of $[[7,1,3]]$ code. The agent already converges after seeing roughly 300 episodes and just improves the average circuit size afterward. The whole training for this plot took 100 seconds and output 10 circuits in parallel.

State preparation on a specific hardware. In the previous result, state preparation was done on fully connected qubits with the standard gate set. However, this connectivity and gate set may not be realistic for current NISQ hardware. The usual procedure is to use the transpiler routine [1] after the state preparation to respect the connectivity and gate set. Here we show the robustness of RL, which can take as input the connectivity and native gate set of a specific NISQ hardware. Thus, RL streamlines the state preparation and transpilation routine into a single process.

In the following, we will focus on IBM quantum devices. The native gate set for IBM quantum devices includes the gates CNOT, X, $\sqrt{X}$, and R_Z, which is a parameterized rotation gate along the z axis. We choose a subset of CNOT, X ,$\sqrt{X}$, and $S = R_Z(\pi/2)$ gate as input to the RL agent.

We prepare the $|0\rangle_L$ state of the $[[5,1,3]]$ perfect code [11] with connectivity from IBMQ Manila, $|0\rangle_L$ of $[[7,1,3]]$ Steane code [21] with connectivity from IBMQ Jakarta, $|+\rangle_L$ of $[[9,1,3]]$ Shor code [20] with connectivity from some part of IBMQ Guadalupe, $|0\rangle_L$ of $[[15,1,3]]$ quantum Reed-Muller code or 3D color code [4] with connectivity from some part of IBMQ Tokyo. We choose the logical operators to be $Z^{\otimes n}$, where n is the number of physical qubits. The placement of the qubits is randomized.

We refer to the RL method that directly incorporates the gate set and connectivity constraint into the training as *RL Direct*. We compare this with taking the RL prepared circuit for fully connected qubits with the lowest circuit size and apply the QISKIT transpilation routine with the maximum optimization level, which we refer to as *RL + Transpile*.

Figure 2b shows the comparison of the state preparation on IBM quantum devices with RL Direct and with RL + Transpile. We see that RL direct produces circuits with smaller circuit sizes compared to RL + Transpile.

Examples of a circuit prepared by the RL agent for $[[5,1,3]]$ in the IBMQ Manila device and $[[7,1,3]]$ in the IBMQ Jakarta device are shown in Figure 2d and e, respectively. We observe that the RL agent qualitatively learns an interesting strategy. For example, in Figure 2e, the agent learns and reuses a new gate sequence ($\sqrt{X}$, CNOT, S, shown with yellow background). This sequence of gate is equivalent to H gate followed by CNOT since H gate is not a part of the IBM quantum device gate set. In other cases, the agent also invents or learns gates such as the $SWAP$ gate to reroute the qubit or the H gate without being explicitly programmed.

4 Conclusion

In this work, we have shown how RL can be used to automatically discover quantum circuits to prepare logical states of a given QEC code with a given gate set and qubit connectivity. Our experimental results show that the RL method can prepare logical states of up to 17 qubits with smaller circuit sizes compared to other methods. We also showed the capability of the RL method to prepare logical states on IBM quantum devices gate set and qubit connectivity of up to 15 qubits.

We have not considered the errors that may occur during the preparation of the logical state. The ongoing extension of this work is to discover a quantum circuit that prepares a logical state in a fault-tolerant manner.

Acknowledgments and Disclosure of Funding

We thank Luis Colmenarez, Markus Müller, Sangkha Borah, Josias Old, Julio Carlos Magdalena de la Fuente, and Manuel Rispler for the fruitful discussions. The research is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

References

[1] Qiskit transpiler. `https://qiskit.org/documentation/apidoc/transpiler.html`. Accessed: 2023-11-16.

[2] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[3] David Amaro, Markus Müller, and Amit Kumar Pal. Scalable characterization of localizable entanglement in noisy topological quantum codes. *New journal of physics*, 22(5):053038, 2020.

[4] Jonas T Anderson, Guillaume Duclos-Cianci, and David Poulin. Fault-tolerant conversion between the steane and reed-muller quantum codes. *Physical review letters*, 113(8):080501, 2014.

[5] Héctor Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New Journal of Physics*, 17(8):083002, 2015.

[6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[7] Sergey Bravyi, Ruslan Shaydulin, Shaohan Hu, and Dmitri Maslov. Clifford circuit optimization with templates and symbolic pauli gates. *Quantum*, 5:580, 2021.

[8] Qiuhao Chen, Yuxuan Du, Qi Zhao, Yuling Jiao, Xiliang Lu, and Xingyao Wu. Efficient and practical quantum compiler towards multi-qubit systems with deep reinforcement learning. *arXiv preprint arXiv:2204.06904*, 2022.

[9] Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[10] Mario Krenn, Jonas Landgraf, Thomas Foesel, and Florian Marquardt. Artificial intelligence and machine learning for quantum technologies. *Physical Review A*, 107(1):010101, 2023.

[11] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Physical Review Letters*, 77(1):198, 1996.

[12] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

[13] Marco Maronese, Lorenzo Moro, Lorenzo Rocutto, and Enrico Prati. Quantum compiling. In *Quantum Computing Environments*, pages 39–74. Springer, 2022.

[14] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[15] Lorenzo Moro, Matteo GA Paris, Marcello Restelli, and Enrico Prati. Quantum compiling by deep reinforcement learning. *Communications Physics*, 4(1):178, 2021.

[16] Tom Peham, Nina Brandl, Richard Kueng, Robert Wille, and Lukas Burgholzer. Depth-optimal synthesis of clifford circuits with sat solvers. *arXiv preprint arXiv:2305.01674*, 2023.

[17] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[18] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.

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

[20] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.

[21] Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Xiaosi Xu, Simon C Benjamin, and Xiao Yuan. Variational circuit compiler for quantum error correction. *Physical Review Applied*, 15(3):034068, 2021.

[24] Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang, and Dong-Ling Deng. Topological quantum compiling with reinforcement learning. *Physical Review Letters*, 125(17):170501, 2020.