Preparing stabilizer states via path-aware reinforcement learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Quantum state preparation forms an essential cornerstone of quantum algorithms. Designing efficient and scalable methods for state preparation on near-term quantum devices remains a significant challenge, with worst-case hardness results compounding this difficulty. We propose a deep reinforcement learning framework for quantum state preparation, using a novel reward function, capable of immediate inference on arbitrary states at a fixed system size post a training phase. This work serves as a starting point for scalable ML-based state preparation algorithms.

1 Introduction

Quantum state preparation is an essential primitive in a vast majority of quantum computing algorithms that require encoding classical data into gate-based quantum hardware. Such algorithms include not only the prototypical methods (e.g., for solving a linear system of equations [1]), but also several common routines for quantum machine learning [2, 3]. Absent robust quantum hardware due to limited coherence times and gate inaccuracies in the current noisy intermediate-scale quantum (NISQ) era [4, 5], efficient state preparation is critical to harboring any hope for *practical* quantum advantage.

Efficient and accurate quantum state preparation is, however, challenging. The key difficulty lies in the fact that almost all states require exponentially large circuits to prepare [6, 7], with the number of circuits of such length blowing up combinatorially thanks to the quadratic number of entangling gates ($\sim n^2$ two-qubit gates) allowed at *each* step. We investigate these results further in Appendix A.3. In essence, one needs to relax the requirements of optimality and ask for an "efficient enough" heuristic instead. There has been much interest in this problem in recent times following the development of larger quantum computers; Appendix B provides an overview of existing methods. In this work, we learn such a heuristic with reinforcement learning (RL), training agents to learn to prepare arbitrary states of a state-space as efficiently as possible. The agent is guided in choosing gates appropriately using a novel reward function that helps remember paths to high-fidelity states seen at intermediate steps in training-time trajectories.

While our approach is broadly applicable, in this work, we focus on an important class of quantum states, namely, the *stabilizer states*. Notably, these states can be physically realized using only gates from the *Clifford* group and find extensive use in quantum error correction [8, 9, 10] and quantum information [11, 12]. Clifford circuits retain much of the expressivity afforded by general quantum circuits, and can generate highly *entangled* or complex states. Furthermore, they can be efficiently classically simulated, leading to faster training and inference times. For stabilizer states, the average circuit size is quadratic in n, which is substantially smaller than the general case; however, the number of possible circuits of this size still grows combinatorially, faster than n^n, making optimal preparation very likely intractable. For all these reasons they serve as an ideal testbed for our method.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

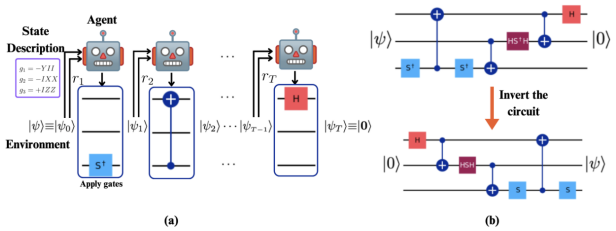
A significant contribution of this work is the analysis of the state preparation heuristics learned, addressing the limitations of existing work in this regard. We prove that our heuristics are guaranteed to succeed on at least 96.5% of all stabilizer states; for 9-qubit states, this corresponds to a guaranteed preparation of more than 4×10^{16} states—despite the agent seeing only 2×10^7 states throughout training. Furthermore, an advantage of heuristics like ours that iterate a next-gate policy is the potential ability to correct gate-noise at intermediate stages of execution of the preparation procedure. Using a standard gate-noise model, we find that the heuristics—especially those restricted to a linear connectivity—are very robust to noise and continue to prepare states successfully with minimal overhead to circuit size. One can view this as a manifestation of the fact that the theoretical guarantee of reverse-preparing $|\psi^*\rangle$ from almost any state $|\psi\rangle$ implies successful preparation whichever noisy state $|\psi'\rangle$ an action took $|\psi\rangle$ to. We also analyze the learned entanglement dynamics of the linear-connectivity agent in preparing highly entangled states: we show that on average, the agent monotonically increases the entanglement content until it reaches the target state.

Finally, to underscore the generality of our approach for RL-based state preparation beyond Clifford circuits, we solve the 3-qubit general problem to 0.99-fidelity. This is a significantly harder problem, with circuits with hundreds of gates required to prepare most states.

2 Methods

We first define the precise problem, before detailing our line of attack: Given $n \in \mathbb{N}$, collection of n-qubit states $\mathcal{S} \subseteq \mathbb{C}^{2^n}$, starting state $|\psi^*\rangle$, a set of gates $\mathcal{A}$ induced by a collection $\mathcal{G}$ of allowed gates and qubit connectivity graph $\mathcal{N}$, and $\epsilon > 0$, find an (efficient) algorithm that upon input any state $|\psi\rangle \in \mathcal{S}$ returns a (short) circuit $\mathcal{C}$ such that $\mathcal{F}(|\psi\rangle, \mathcal{C}|\psi^*\rangle) \geq 1 - \epsilon$.

Naïvely, our goal of starting from a *fixed* state and reaching arbitrary targets of choice is at odds with the RL setup where the start state may be sampled from a distribution but the target is fixed. We employ a standard reverse-preparation trick [13, 14, 15, 16] to overcome this: we train the RL agent instead to learn paths from any given state $|\psi\rangle$ to target state $|\mathbf{0}\rangle$. That is, we learn paths from $|\mathbf{0}\rangle$ to arbitrary states *in reverse*, by sampling states from the state-space and requesting a path from them *to* $|\mathbf{0}\rangle$. At test-time, we efficiently invert the generated circuits to prepare $|\psi\rangle$ from $|\mathbf{0}\rangle$. See the figure for a pictorial depiction and Table 3 for a more detailed setup.



2.1 Memory-Guided Reward (MGR)

Often, the difficult part in an RL design is the reward function: it must be both informative and computationally efficient. In Appendix C, we discuss issues with natural reward functions for state preparation in detail, and here, we describe our reward that successfully helps agents learn short paths in the above RL framework. In what follows, we denote the (variable) length of an episode by T.

To provide useful signal to the learning agent, the reward function must reward actions that led to a high-fidelity state at some step ahead, and should be positive whenever the agent makes new progress ahead in the trajectory, say by visiting a state with better fidelity than seen previously in the trajectory. To achieve this, we must remember when the maximum-fidelity state of the trajectory occurred and assign positive cumulative return to actions that came before, and non-positive cumulative return to those that came after. To achieve these objectives, we choose the reward

$$r(s_i, a_i, s_{i+1}) = \gamma M_{i+1} - M_i, \tag{1}$$

where $M_i := \max_{j \leq i} \mathcal{F}(s_j, |\mathbf{0}\rangle)$ is the maximum fidelity seen until now. Essentially, the reward for the ith action is positive if the $(i+1)$-th state s_{i+1} has fidelity higher than that seen so far [1] and is ≈ 0 if not. The cumulative reward compounds positive rewards after step i; Lemma C.2 proves that

[1] In fact, it will have fidelity at least *twice as much* as seen before, by the well-known fact that stabilizer state fidelities are always an inverse power of 2 [17].

the cumulative reward is exactly $G_i = \gamma^{T-i} M_T - M_i$: indeed, this is positive in proportion to the maximum fidelity across the trajectory M_T, and ≈ 0 after a maximum-fidelity state is reached.

2.2 Theoretical guarantees on performance

Let S be a collection of quantum states. Define a property (such as "the state is prepared correctly with a circuit containing at most 34 gates") by a function $P : \{0,1\}^* \times S \to \{0,1\}$ taking as input a proposed state preparation model T and a state $|\psi\rangle \in S$, and outputting 1 if the property is satisfied when T is run on $|\psi\rangle$ and 0 otherwise. Now, given a trained agent T, we seek to show that some desired properties hold over most states $|\psi\rangle \in S$. It is nearly impossible to examine all of S; we resort to randomized methods (risking a tiny probability of erroneous results), uniformly sampling a large number of states from S and testing the property P on these states, following which we appeal to concentration to render our sampled results almost exact for the whole state space S. This approach (proofs and computation in Appendix E) indicated that in every case,

$$\Pr_{|\psi\rangle \sim \mathbb{U}(S)} [P(T, |\psi\rangle) = 1] \geq 0.965.$$

Since the state space S is finite, the left-hand-side probability is simply the number of states satisfying P over $|S|$, so we get that the number of states prepared successfully by the agent is at least 96.5% of the full stabilizer state space. This is quite remarkable, given that the agent sees a tiny fraction of the state space (around 1 in 1 billion) during the entirety of its training and has generalized to almost the whole space despite this. Using a similar technique, we also bound the number of gates that the agents use to prepare at least 95% of all states; the results are presented in Table 1 and details of the calculation in Appendix E. This almost-universal correctness and efficiency guarantee for our heuristics provides much support for their integration into the quantum compiling and control stacks.

3 Experimental results

Tasks. We trained agents to prepare arbitrary stabilizer states, and assessed the learned agents' preparation capabilities by inference on uniformly sampled stabilizer states (cf. Table 1) and highly entangled brickwork states (cf. Figure 1) for each value of the number of qubits $n = 5, 6, 7, 9$ with both linear and full connectivity. As seen in Section 2.2, the agent already prepares states exactly. We thus focus on the efficiency of the (correctly) generated circuits. We define a circuit's efficiency to be the number of gates used.

Baselines. We benchmark our agents against two popular state preparation algorithms: (a) Aaronson & Gottesman [18], which generates circuits using a canonical decomposition of the Clifford unitary of interest, and (b) the state-of-the-art heuristic proposed by Bravyi et al. [19] that comprises several rounds of template matching and peephole optimizations to compile the final circuit from an initial inefficient circuit. Both methods use the X, Y, Z, H, S, CNOT and SWAP gates with full qubit connectivity, while we use a significantly smaller set (cf. Appendix F.3). Both methods are accessed via the IBM Qiskit API [20]. The full experimental setup may be found in Appendix D, and a few additional experiments in Appendix F.

Performance. The learned policies perform substantially better than other methods despite a more restrictive gate-set. Further, unlike the baselines, our heuristics are *guaranteed* to prepare at least

Table 1: Circuit size (↓) comparison with baselines, averaged across 200 uniformly random stabilizer states of the appropriate size. The agents are guaranteed to prepare at least 95% of all n-qubit stabilizer states using at most T_n gates, where T_n is as described in Appendix E.

Algorithm	5-qubit	6-qubit	7-qubit	9-qubit
Aaronson & Gottesman [18]	26.00 ± 6.37	36.43 ± 7.25	48.13 ± 7.29	76.56 ± 8.23
Bravyi et al. [19]	21.10 ± 4.88	29.77 ± 5.97	38.50 ± 5.96	59.24 ± 7.57
DRL (linear connectivity)	**15.52** ± 3.25	**21.68** ± 3.32	**30.18** ± 4.09	**48.74** ± 4.50
DRL (full connectivity)	**12.83** ± 2.40	**17.86** ± 2.88	**24.36** ± 3.47	**41.92** ± 5.91
Guarantee T_n (full conn.)	22	29	32	56

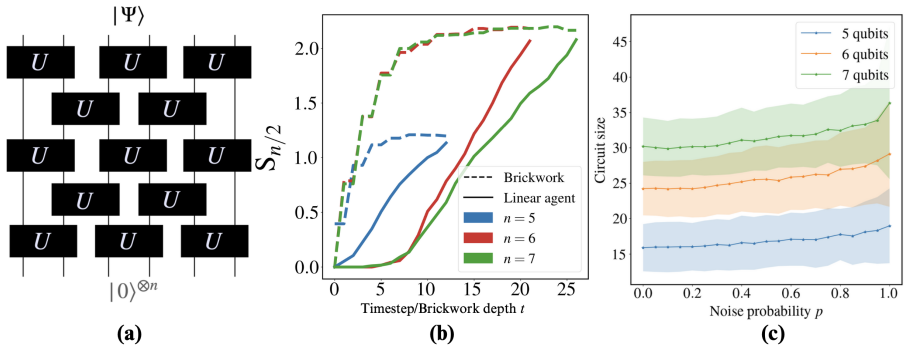


Figure 1: **(a)** A schematic of brickwork Clifford circuits, where each U is sampled independently from the 2-qubit Clifford group. **(b)** The progression of entanglement entropy during the preparation of volume-law entangled $2n$-depth brickwork states (solid) vs the entanglement entropy of n-qubit brickwork states of increasing depth t (dashed). **(c)** Effect of noisy operations on circuit size in the linear-connectivity case. The learned heuristics have minimal overhead in circuit size even at large p.

95% of all n-qubit stabilizer states using at most T_n gates (cf. Table 1) as explained in Appendix E. The results are presented in Table 1.

Robustness. On a quantum computer, gate operations always suffer from errors upon implementation [7]. A standard noise model is the local bit/phase flip channel model, where with probability $p \lessapprox 0.1$, a uniformly random qubit out of those affected by U is either bit-flipped or phase-flipped. We find that the agent continues to succeed in this noisy environment, correcting for the noise with small size overheads (cf. Figure 1(c)).

Learned entanglement dynamics. The linear connectivity agents generate *local* dynamics when preparing circuits: interactions on ≤ 2 qubits at each step. Random brick-work circuits on n qubits are prototypical examples of this, displaying a linear increase in the partial circuits' half-chain entanglement entropy $S_{n/2}(t)$ (cf. Appendix A) followed by saturation at $t \lesssim 2n$. It is thus of interest to contrast the dynamics generated by linear agents versus brickwork layers. The solid lines in Figure 1(b) show the agent's $S_{n/2}(t)$ averaged over the preparation of 200 $2n$-deep brickworks; the dashed lines denote those of the brickworks. We see that the agents also achieve monotonically improving entropy, implying little redundancy in their gate usage.

General state preparation. Our framework and reward function are sufficiently general to prepare richer classes of states beyond stabilizers. As an example, we train an agent to prepare the full 3-qubit state space to a fidelity of $> 99\%$ ($\epsilon = 0.01$). Appendix F.2 details the full experimental setup and result statistics. Almost automatically, our sampling analysis as done in Appendix E yields a *guarantee* of preparing at least 96.5% of the infinite state space successfully. Benchmarking used 1500 Haar-randomly sampled states: the agent succeeded each time, with an average gate count of 286.

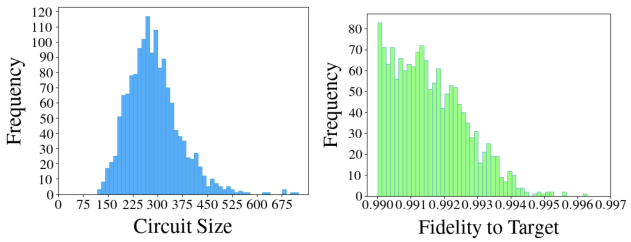


Figure 2: Frequency plots of the 3-qubit general state preparation heuristic. All states were prepared successfully.

4 Conclusions

In this work, we have demonstrated that deep reinforcement learning can facilitate immediate inference on arbitrary stabilizer states without needing re-training. This is achieved utilizing a novel reward function. The framework is sufficiently general as to accommodate various families of quantum systems. Our experiments perform a thorough analysis of the agent's inference capabilities and providing promise for the integration of RL-based methods into real quantum computing environments for transpilation.

References

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), October 2009.

[2] Scott Aaronson. Quantum machine learning algorithms : Read the fine print, 2015.

[3] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, July 2021.

[4] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[5] Xianjing Zhou, Xinhao Li, Qianfan Chen, Gerwin Koolstra, Ge Yang, Brennan Dizdar, Yizhong Huang, Christopher S. Wang, Xu Han, Xufeng Zhang, David I. Schuster, and Dafei Jin. Electron charge qubits with 0.1 millisecond coherence time, 2023.

[6] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995.

[7] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[8] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997.

[9] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, September 2017.

[10] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz. Realization of real-time fault-tolerant quantum error correction. *Phys. Rev. X*, 11:041058, Dec 2021.

[11] Zak Webb. The clifford group forms a unitary 3-design, 2016.

[12] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, June 2020.

[13] David Kremer, Victor Villar, Hanhee Paik, Ivan Duran, Ismael Faro, and Juan Cruz-Benito. Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning, 2024.

[14] Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang, and Dong-Ling Deng. Topological quantum compiling with reinforcement learning, 2020.

[15] Sebastian Rietsch, Abhishek Y. Dubey, Christian Ufrecht, Maniraman Periyasamy, Axel Plinge, Christopher Mutschler, and Daniel D. Scherer. Unitary synthesis of clifford+t circuits with reinforcement learning. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 824–835. IEEE, September 2024.

[16] Pavel Tashev, Stefan Petrov, Friederike Metz, and Marin Bukov. Reinforcement learning to disentangle multiqubit quantum states from partial observations. *arXiv preprint arXiv:2406.07884*, 2024.

[17] Theodore J. Yoder. A generalization of the stabilizer formalism for simulating arbitrary quantum circuits, 2012.

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

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

[20] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with qiskit, 2024.

[21] Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm, 2005.

[22] Sarah Schneider, Lukas Burgholzer, and Robert Wille. A sat encoding for optimal clifford circuit synthesis. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ASPDAC '23, page 190–195, New York, NY, USA, 2023. Association for Computing Machinery.

[23] Sergey Bravyi, Joseph A. Latone, and Dmitri Maslov. 6-qubit optimal clifford circuits. *npj Quantum Information*, 8(1), July 2022.

[24] Yaodong Li, Xiao Chen, and Matthew P. A. Fisher. Measurement-driven entanglement transition in hybrid quantum circuits. *Phys. Rev. B*, 100:134306, Oct 2019.

[25] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[26] Ashlesha Patil and Saikat Guha. Clifford manipulations of stabilizer states: A graphical rule book for clifford unitaries and measurements on cluster states, and application to photonic quantum computing, 2023.

[27] M. H. Cheng, K. E. Khosla, C. N. Self, M. Lin, B. X. Li, A. C. Medina, and M. S. Kim. Clifford circuit initialisation for variational quantum algorithms, 2022.

[28] Gokul Subramanian Ravi, Pranav Gokhale, Yi Ding, William M. Kirby, Kaitlin N. Smith, Jonathan M. Baker, Peter J. Love, Henry Hoffmann, Kenneth R. Brown, and Fred Chong. Cafqa: A classical simulation bootstrap for variational quantum algorithms. *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2022.

[29] Jiace Sun, Lixue Cheng, and Shi-Xin Zhang. Stabilizer ground states: theory, algorithms and applications, 2024.

[30] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[32] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[33] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2015.

[34] Vivek V. Shende, Igor L. Markov, and Stephen S. Bullock. Minimal universal two-qubit controlled-not-based circuits. *Physical Review A*, 69(6), June 2004.

[35] Adam Bouland and Tudor Giurgica-Tiron. Efficient universal quantum compilation: An inverse-free solovay-kitaev algorithm, 2021.

[36] Xiao-Ming Zhang, Zezhu Wei, Raza Asad, Xu-Chen Yang, and Xin Wang. When does reinforcement learning stand out in quantum control? a comparative study on state preparation. *npj Quantum Information*, 5, 2019.

[37] Run-Hong He, Rui Wang, Shen-Shuang Nie, Jing Wu, Jia-Hui Zhang, and Zhao-Ming Wang. Deep reinforcement learning for universal quantum state preparation via dynamic pulse control. *EPJ Quantum Technology*, 8(1), December 2021.

[38] Thomas Gabor, Maximilian Zorn, and Claudia Linnhoff-Popien. The applicability of reinforcement learning for the automatic generation of state preparation circuits. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 2196–2204, New York, NY, USA, 2022. Association for Computing Machinery.

[39] Michael Kölle, Tom Schubert, Philipp Altmann, Maximilian Zorn, Jonas Stein, and Claudia Linnhoff-Popien. A reinforcement learning environment for directed quantum circuit synthesis, 2024.

[40] Remmy Zen, Jan Olle, Luis Colmenarez, Matteo Puviani, Markus Müller, and Florian Marquardt. Quantum circuit discovery for fault-tolerant logical state preparation with reinforcement learning, 2024.

[41] Hsin-Yuan Huang, Yunchao Liu, Michael Broughton, Isaac Kim, Anurag Anshu, Zeph Landau, and Jarrod R. McClean. Learning shallow quantum circuits. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC '24, page 1343–1351. ACM, June 2024.

[42] Tom Peham, Nina Brandl, Richard Kueng, Robert Wille, and Lukas Burgholzer. Depth-optimal synthesis of clifford circuits with sat solvers, 2023.

[43] A. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

[44] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–4094, May 1995.

[45] Simon J. Evered, Dolev Bluvstein, Marcin Kalinowski, Sepehr Ebadi, Tom Manovitz, Hengyun Zhou, Sophie H. Li, Alexandra A. Geim, Tout T. Wang, Nishad Maskara, Harry Levine, Giulia Semeghini, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature*, 622(7982):268–272, October 2023.

[46] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021.

[47] M. Morales. *Grokking Deep Reinforcement Learning*. Manning, 2020.

[48] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correction code, 1996.

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

[50] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

A Background

A.1 Quantum computation

We briefly introduce the principles of quantum computing, quantum circuits and stabilizer states. For more elaborate discussions of these topics, we recommend [7] for quantum computing and [18, 17] for stabilizer state theory.

Notation. We use $A^\dagger$ to denote the hermitian conjugate $\overline{A^\top}$ of operator A. We shall also make use of Dirac notation: the 'ket' $|\psi\rangle$ represents a column vector and 'bra' $\langle\psi| = |\psi\rangle^\dagger$ the dual row vector.

A.2 Quantum circuits and Qubit connectivity

A quantum gate or operation on a system of qubits is a unitary linear operator U (i.e. $U^{-1} = U^\dagger$) on the corresponding Hilbert space.

The Pauli group consists of the following canonical single-qubit gates (represented as matrices w.r.t. the computational basis $\{|0\rangle, |1\rangle\}$).

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -\iota \\ \iota & 0 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

From definition, it can be inferred that X and Z act as $X|b\rangle = |1-b\rangle$ and $Z|b\rangle = (-1)^b|b\rangle$ on the qubit state. The single-qubit Pauli group generalizes to the n-qubit Pauli group $\mathcal{P}_n$, which consists of tensor products of single-qubit Pauli gates. Other useful quantum gates are the single-qubit Hadamard gate H and single-qubit phase gate S, which act as $H|b\rangle = (|0\rangle + (-1)^b|1\rangle)/\sqrt{2}$ and $S|b\rangle = \iota^b|b\rangle$ respectively. The computational basis matrix representations are given by

$$I = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \text{and} \quad X = \begin{pmatrix} 1 & 0 \\ 0 & \iota \end{pmatrix}.$$

The canonical two-qubit gate is the controlled-NOT (CNOT), which operates on one target qubit conditioned on one control qubit by $|x, y\rangle \mapsto |x, x \oplus y\rangle$. The computational basis (this time, containing the four elements $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$) representation is

$$\text{CNOT}_{1,2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Here, the subscript $1, 2$ represents the fact that the first qubit acts as control and the second qubit is the target.

Finally, for our purposes, a quantum circuit is a visual representation of a sequence of quantum gates $[U_1, U_2, \cdots, U_k]$ applied left-to-right, and is thus associated with the quantum operation $U = U_k U_{k-1} \cdots U_1$.

Connectivity graphs. A connectivity graph is a practical relaxation: in today's quantum computers, not all qubits are adjacent to each other, and hence it is not physically possible to apply entangling gates, e.g. CNOT, to every pair of qubits. A connectivity graph specifies when we may apply an entangling gate: an edge $\{q_i, q_j\}$ between two qubits in the connectivity graph indicates that entangling gates may be applied to the system $q_i q_j$. See Figure 3 for an example.

A connectivity graph together with an allowed set of gates, e.g. $\{H, S, CNOT\}$ naturally induces a set of unitary gates that may be applied to the full system. An example induced set is presented in Figure 3.

A.3 Hardness of the state preparation problem

The state-space graph associated with a quantum system, gate-set and qubit connectivity is constructed as follows. The vertex set V is the state-space itself, e.g. the (infinite) set of n-qubit states. We pick an allowed induced gate U, e.g. H applied to the ith qubit, and add edges $(|\psi\rangle, U|\psi\rangle)$ labeled by U for every $|\psi\rangle \in V$. Doing this across all induced gates yields the complete state-space graph. When

Gate	Denotation
$I\,\lvert x\rangle = \lvert x\rangle$	——
$X\,\lvert x\rangle = \lvert x \oplus 1\rangle$	—[X]—
$Y\,\lvert x\rangle = (-1)^x i\,\lvert x \oplus 1\rangle$	—[Y]—
$Z\,\lvert x\rangle = (-1)^x\,\lvert x\rangle$	—[Z]—
$H\,\lvert x\rangle = \frac{\lvert 0\rangle + (-1)^x \lvert 1\rangle}{\sqrt{2}}$	—[H]—
$S\,\lvert x\rangle = i^x\,\lvert x\rangle$	—[S]—
$T\,\lvert x\rangle = e^{i\pi x/4}\,\lvert x\rangle$	—[T]—
$\mathrm{CNOT}\,\lvert x, y\rangle = \lvert x, y \oplus x\rangle$	
$\mathrm{SWAP}\,\lvert x, y\rangle = \lvert y, x\rangle$	

Table 2: Description and representation of the gates used in this work.

the state-space graph is infinite, a popular notion of size is the ε-covering number, representing the minimum size of a collection of elements of the space—called a cover—that satisfy the covering property: every state of the state space is ε-close (here, in fidelity) to some element of the cover. For finite graphs, the ε-covering number for small enough ε is the size of the graph itself.

The ε-covering number of the n-qubit state space graph is well-known to be *doubly exponential* in the number of qubits:

Claim A.1. The ε-covering number of the n-qubit state space $M(\mathcal{H}^{\otimes n}; \varepsilon)$ is given by

$$M(\mathcal{H}^{\otimes n}; \varepsilon) = \frac{\sqrt{\pi}\Gamma(2^n - 1/2)(2^{n+1} - 1)}{\Gamma(2^n)} \frac{1}{\varepsilon^{2^{n+1}-1}} = \Omega\left(\frac{1}{\varepsilon^{2^{n+1}-1}}\right).$$

Here, Γ denotes the gamma function.

Proof. See Sec. 4.5.4 of [7]. □

Claim A.1 has the following immediate corollary:

Claim A.2. Suppose $\mathcal{G}$ is a finite gateset of allowed gates, e.g. $\{H, S, HSH, Z, \mathrm{CNOT}\}$. Then there are n-qubit states ψ that are exponentially hard to optimally prepare to $(1-\varepsilon)$-fidelity; in fact, the optimal circuit has length at least

$$\Omega\left(\frac{2^n \log 1/\varepsilon}{\log n}\right)$$

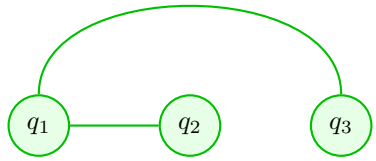
and thus trivially takes exponential time to construct and return.

Proof. Suppose $\mathcal{G}$ consists of g gates, each acting on at most f qubits. The total number of circuits with at most m gates from $\mathcal{G}$ is upper-bounded by n^{fgm}. However, the number of distinct circuits required to prepare every state to $(1-\varepsilon)$-fidelity is at least the minimum size of a ε-covering set, i.e. the ε-covering number. Thus, letting m denote the maximum number of gates (as a function of n) required to prepare n-qubit states to $(1-\varepsilon)$-fidelity, we must have

$$n^{fgm} \geq M(\mathcal{H}^{\otimes n}; \varepsilon) = \Omega\left(\frac{1}{\varepsilon^{2^{n+1}-1}}\right),$$

which implies $m \geq \Omega\left(\frac{2^n \log 1/\varepsilon}{\log n}\right)$, as needed. □

Connectivity Graph $\mathcal{N}$



Gate set $\mathcal{G}$:

$$H\left|x\right\rangle := \frac{\left|0\right\rangle+(-1)^x\left|1\right\rangle}{\sqrt{2}},\ S\left|x\right\rangle := \iota^x\left|x\right\rangle,\ \text{CNOT}\left|x,y\right\rangle := \left|x, x\oplus y\right\rangle$$

Induced gates $\mathcal{A}$:

$$H\otimes I\otimes I,\ I\otimes H\otimes I,\ I\otimes I\otimes H$$

$$S\otimes I\otimes I,\ I\otimes S\otimes I,\ I\otimes I\otimes S$$

$$\text{CNOT}_{1,2}\otimes I_3,\ \text{CNOT}_{2,1}\otimes I_3$$

$$\text{CNOT}_{1,3}\otimes I_2,\ \text{CNOT}_{3,1}\otimes I_2$$

Figure 3: An example of a connectivity graph, gate set ($x, y \in \{0,1\}$) and induced gates for a 3 qubit quantum system.

Corollary A.3. *Let $0 > \varepsilon < 1$ be fixed and $q(n)$ be a polynomial in qubit size n. The fraction of all states that are realizable to $(1-\varepsilon)$-fidelity with a circuit of size at most $q(n)$ goes to 0 as $n \to \infty$. Or almost every state is exponentially hard to prepare, asymptotically.*

Thus, the problem of state-preparation is already worst-case exponentially hard because the circuits are exponentially long! This is not just a worst-case estimate: nearly *all* states require exponentially long circuits, for large enough n:

Proof. Following the proof of Claim A.2, the fraction η_q corresponding to states prepared using at most $m = q(n)$ gates satisfies

$$\eta_q \leq \frac{n^{fg\cdot q(n)}}{\varepsilon^{2^{n+1}-1}} \to 0,$$

since $n^{fg\cdot q(n)} = \exp\left(fg\cdot q(n)\log n\right) = o\left(\exp\left((2^{n+1}-1)\log\frac{1}{\varepsilon}\right)\right)$. □

The fact that one needs to find a long circuit makes the problem hard. Indeed, even a circuit of length 5 (circuits will typically be at least $2^9 = 512$ gates long) at $n = 9$ qubits can be chosen from around $99^5 \approx 10^{10}$ options, since each gate can be one of 72 CNOTs and 9 each of H, S and T gates (the smallest universal set for general state preparation). To search over circuits of length in the $100s$ of qubits, even to prepare a single state, requires searching over a gigantic number of states in the state-space graph, which is simply not feasible. Thus, algorithms for optimal circuits are likely to be computationally intractable very, very quickly as n increases.

This suggests looking for a relaxed algorithm: a fast algorithm that returns *approximately optimal-size* circuits for *almost all* states of the state-space. In particular, we allow some slack from optimality,

and allow the algorithm a small failure-rate, allowing the algorithm to fail on a small number of "worst-case" states if the slack on more typical states can be reduced instead.

The Solovay-Kitaev algorithm [21] is a classic approximation algorithm for state preparation; it is known to be optimal for circuit size up to a factor polynomial in n. However, this factor is often large in practice, which calls for more sophisticated circuit search methods. In recent times, there has been much work in approximating the state-space graph search for optimal circuits via machine and reinforcement learning (cf. Appendix B).

In the next section, we describe stabilizer states, which are a rich subset of states that find enormous use throughout quantum information science. It turns out that the optimal state preparation problem on stabilizer state is slightly more manageable, with the state-space graph size growing singly-exponentially as 2^{n^2} [18]. The state preparation problem thus remains hard on stabilizer states [22], although the worst-case (and average-case) optimal circuit size is $\Theta\left(\frac{n^2}{\log n}\right)$, which is polynomial in n. Much like its general counterpart, it can be shown by a counting argument like Claim A.2 that asymptotically, almost all stabilizer states require $\Omega\left(\frac{n^2}{\log n}\right)$ gates to prepare. Note again that the degree of the state-space graph is quadratic ($n^2 - n$ CNOTs) and so the number of possible circuits even with a few gates quickly blows up and an exhaustive search will not scale.

The goal then is to find algorithms that bring the worst-case circuit size down, while continuing to return correct preparations for almost all states. As Appendix B details, there have been decomposition-based, rule-based and learning-based algorithms for this problem; however, the problem remains open in terms of the best tradeoff between small circuit size and high success rate.

A.4 Stabilizer states

We say that an element $\pi \in \mathcal{P}_n$ stabilizes state $|\psi\rangle$ if $\pi |\psi\rangle = |\psi\rangle$. The set of stabilizers of a state comprises its stabilizer group (generated by at most n elements). A state is a *stabilizer state* iff its stabilizer group is generated by n elements. Conversely, every $\mathbb{S}$ uniquely determines a corresponding stabilizer state $|\psi\rangle$ as the simultaneous eigenstate with eigenvalue 1, $g|\psi\rangle = |\psi\rangle, \forall g \in \mathbb{S}$. Since a stabilizer group generator $\in \mathcal{P}_n$ can be represented using $2n+1$ bits, a stabilizer state $|\psi\rangle$ can be written using $n(2n+1)$ bits.

Stabilizer states can also be characterized as the states that can be reached from the all-zeros state $|\mathbf{0}\rangle$ using *Clifford* circuits, i.e. unitaries that are a combination of H, S and CNOT gates. Notably, the Pauli gates are Clifford unitaries. The action of each of these gates on a stabilizer state's bit-representation is simple, resulting in the efficient classical simulation of quantum computation exclusively with Clifford unitaries [18]. This alternate interpretation of stabilizer states leads to them also being called *Clifford* states.

As mentioned at the end in the previous section, preparing stabilizer states optimally remains a challenge despite efficient classical simulability, since the number of stabilizer states grows rapidly as $2^{\mathcal{O}(n^2)}$. Known optimal implementations have been limited to 6 qubits [23]. Further, the (anti-)commutation and involutary properties of Clifford gates make it harder to reason about locally greedy search steps. We outline existing work towards stabilizer state preparation in Appendix B.

Due to their simple mathematical structure and their ability to capture volume-law entanglement (where the entanglement content grows with the volume of the qubit lattice, i.e., $S \sim cn$ for a linear $n-$qubit chain) [24], stabilizer states enjoy vast applicability. They have found immense use in the exploration of quantum information [11, 12] and are also crucial for quantum error correction (QEC) [8, 7, 9, 10]. They are also applied beyond to measurement-based quantum computing [25, 26], quantum-classical hybrid algorithms [27, 28] and even ground-state physics [29].

A.5 Reinforcement learning

In a Reinforcement Learning (RL) problem an agent learns through interactions with an environment to maximize its reward [30]. The environment is modeled as a Markov decision process, consisting of (a) a set $\mathcal{S}$ of states of the environment, (b) a set of actions $\mathcal{A}$ of the agent, (c) a transition function $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ where $p(s'|s, a)$ is the probability that the state of the environment will be s' if the environment is in state s and the agent takes action a, (d) a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$

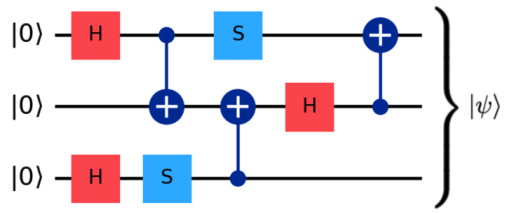


Figure 4: A Clifford circuit preparing stabilizer state $|\psi\rangle$. The $\oplus$-end of a CNOT gate denotes its target qubit.

with $r(s, a, s')$ representing the *reward* that the agent receives from the environment for taking action a from state s and reaching state s', and (e) a set $\mathcal{T} \subset \mathcal{S}$ of terminal states. The interaction between agent and environment stops on reaching a terminal state or exceeding a maximum number T of actions without reaching a terminal state.

A policy of an RL agent is a function, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ with $\pi(a|s)$ being the probability that the agent will take action a when in state s. A trace/trajectory τ of π is a tuple of alternating states and actions, starting and ending in a state: $\tau = (s_0, a_0, s_1, \cdots, a_{T-1}, s_T)$. A policy π along with a distribution μ over possible start states s_0 induces a distribution over traces, with $a_i \sim \pi(\cdot|s_i)$, $s_{i+1} \sim p(\cdot|s_i, a_i)$ for each i.

The cumulative reward, or return, at step i of a T-step trace is defined by

$$G_i(\tau) \equiv \sum_{j=i}^{T} \gamma^{j-i} r(s_j, a_j, s_{j+1}),$$

where $\gamma \in (0, 1)$ is the discount factor, describing the value of future actions in the present. In our work, $\gamma = 0.99 \equiv 1$, so the cumulative return is

The return of trace τ is defined by $G(\tau) := G_0(\tau)$, the full cumulative return of the episode. The goal in RL is to find a policy π^* that maximizes the expected return

$$J_\pi := \mathbb{E}_{\tau \sim (\mu, \pi)} \left[G(\tau) \right]. \tag{2}$$

Two key objects of interest in the search for such a policy are the value function $V^\pi(s) := \mathbb{E}_{\tau \sim \pi | s_0 = s} \left[G(\tau) \right]$ and the Q-function $Q^\pi(s, a) := \mathbb{E}_{\tau \sim \pi | s_0 = s, a_0 = a} \left[G(\tau) \right]$. An associated function is the advantage function, denoting how much better a particular action is w.r.t the average:

$$A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s).$$

We use Proximal Policy Optimization (PPO) throughout our experiments. PPO [31] is a reinforcement learning algorithm from the class of actor-critic algorithms designed to improve stability and efficiency in policy optimization. A running policy function π (parameterized by θ) and value function V (parameterized by ϕ) are maintained, typically as neural networks. Multiple agents gather experience by taking actions in the environment, according to current policy π. Concurrently, advantages $A^\pi(s, a)$ are estimated, approximating $Q^\pi(s, a)$ and $V^\pi(s)$ using sample averages over the experiences collected. In practice, one replaces advantages by generalized advantages, exponentially-weighted linear combinations of the advantages along a trace, which yield more robust estimates [32].

Once sufficiently many steps are collected, we perform several optimization steps. Each optimization step starts by sampling a minibatch $\mathcal{D} = \{(s_i, a_i, \widehat{A}_i)\}_i$, where $\widehat{A}_i$ is the advantage estimate, from the experience pool $\mathcal{E}$. We next compute the policy objective of PPO, which can be viewed as a simplified alternative to the objective in Trust Region Policy Optimization [33]:

$$\mathcal{J}^{\text{CLIP}}(\theta) = \mathbb{E}_{(s_i, a_i, \widehat{A}_i) \sim \mathcal{E}} \left[\min \left(r(\theta) \widehat{A}_i, \text{clip}\left(r(\theta), 1 - \epsilon, 1 + \epsilon\right) \widehat{A}_i \right) \right] \tag{3}$$

where $r_i(\theta) = \pi_\theta(a_i|s_i) / \pi_{\theta_{\text{old}}}(a_i|s_i)$ denotes the probability ratio between new and old policies with respective parameters θ and θ_{old}. The clip function is defined for $a < b$ by $\text{clip}(x, a, b) = \max(a, \min(x, b))$. ϵ is the clipping hyperparameter; clipping ensures that the new policy does not

deviate significantly from the old policy, thereby providing more stable learning. The gradient of the objective is computed and the parameters θ updated by gradient ascent. This completes one policy optimization step, and the process is now repeated, starting with sampling a new minibatch. Along with the policy objective, the value function is trained via the (clipped) value loss defined by

$$\mathcal{L}^{\text{value}}(\phi) = \mathbb{E}_{(s_i, a_i, \widehat{G}_i)\sim\mathcal{E}}\left[\max\left(\left(\widehat{G}_i - V_\phi(s_i)\right)^2, \left(\widehat{G}_i - V_\phi^{\text{clip}}(s_i)\right)^2\right)\right]. \tag{4}$$

Here, $V_\phi^{\text{clip}}(s_i) = V_{\phi_{\text{old}}}(s_i) + \text{clip}\left(V_\phi(s_i) - V_{\phi_{\text{old}}}(s_i), -\epsilon, \epsilon\right)$ stabilizes the update $\phi_{\text{old}} \to \phi$. $\widehat{G}_i$ refers to the cumulative reward obtained starting from (s_i, a_i), estimated from the trace containing the step (s_i, a_i). Finally, to encourage exploration, an entropy term is also included as part of the policy objective,

$$\mathcal{J}^{\mathcal{H}}(\theta) = -\mathbb{E}_{s_i\sim\mathcal{E}}\left[\mathcal{H}(\pi_\theta(\cdot \mid s_i))\right], \tag{5}$$

here $\mathcal{H}(\pi_\theta(\cdot \mid s_i))$ represents the *entropy* of the policy in state s_i.

B Related work

There is a rich body of literature devoted to preparing quantum states. Algorithmic methods include the quantum Shannon decomposition [34], Solovay-Kitaev construction [21] and recently an inverse-free Solovay-Kitaev type construction [35]. These algorithms succeed on all states, but are often far from optimal in absolute terms. Complexity-theoretically, they are a polynomial factor of n from optimal.

Algorithmic heuristics for the preparation of stabilizer states have been studied in great detail as well [18, 19]. Exhaustive search has been performed to find optimal circuits for up to 6-qubit stabilizer states [23].

In recent times, machine learning and deep reinforcement learning approaches have been examined to learn preparation heuristics for small systems [36, 37, 38, 39]. [37, 39] have explored learning a heuristic for general state preparation up to 2 qubits. Tashev et al. [16] go significantly further, leveraging the existence of an analytical best-disentangling-gate once the qubits to disentangle have been chosen, and are able to scale their approach to preparing 5-qubit general states. Zen et al. [40] use deep RL to learn to prepare specific logical code states. Huang et al. [41] used a method using local circuit inversions to learn shallow unitaries. Finally, approaches based on representing stabilizer state preparation as an optimization problem and using a SAT solver have also been studied [42, 22].

We would like to expand discussion w.r.t two papers with similar goals, Bravyi et al. (2021) [19] and Kremer et al. (2024) [13]. The former is the current state-of-the-art algorithmic heuristic method for stabilizer state preparation, and the latter a top method for reinforcement-learning based state preparation. Both methods, however, focus on the related Clifford circuit optimization problem (which lends itself to state preparation). Despite this, they remained the best state-preparation methods as well.

Bravyi et al. (2021) initially prepare an un-optimized circuit and optimize it with rule-based replacement and symbolic peephole optimization till convergence. They benchmark their method against an optimal database [23], showing close to optimal CNOT (not total gate) usage, and also against the previous baseline of Aaronson-Gottesman (2004) [18].

Kremer et al. (2024), like our work, also advocate an approach based on RL. However, they slowly increase the task difficulty to solve the problem, while we directly use our maximum-based reward to achieve successful training. In terms of experiments, Kremer et al. (2024)'s focus is on different architectures and CNOT counts, showing a uniform improvement over Bravyi et al. They also remark that their agent succeeds on virtually all tested states, which we believe can be turned into a theoretical guarantee as well; however, they do not provide one.

In contrast to these works, we directly attack the state preparation algorithm, seeking to optimize circuit size further. Note that our goal is to optimize the total circuit size (not just CNOTs), with key focus being to provide worst-case performance guarantees and analyze properties of the learnt algorithm empirically. In addition to performance benchmarks, we analyze the agents' resistance to noise and entanglement dynamics, which we believe are important considerations in the analysis of state preparation agents.

C Reward Analysis

We shall describe and analyze different natural reward functions that do not succeed, leading finally to the derivation of the MGR reward that we use. In what follows, we denote the (variable) length of an episode by T (with maximum allowed value T^*, i.e. T^* is the maximum allowed length of an episode, after which the episode is truncated).

A simple choice of reward is the constant penalty, $r(s_i, a_i, s_{i+1}) := -\alpha$; or, as used in [39], $r(s_i, a_i, s_{i+1}) = T^* - i$ when $s_{i+1} = |\mathbf{0}\rangle$ to reward the agent for succeeding. This generates reward dynamics akin to learning to escape a maze: exploration of a majority of the state space is virtually necessary for success, since all visited states are given roughly the same value estimate until the goal state is reached. Given the large ($2^{\mathcal{O}(n^2)}$-sized) state space, learning by exploring the space is simply infeasible, and the agent learns to improve fidelity only slightly, even at $n = 6$ (cf. Figure 5).

Another natural choice of reward function is the fidelity to the target state: $r(s_i, a_i, s_{i+1}) = \Phi(s_{i+1}) := \mathcal{F}(s_{i+1}, |\mathbf{0}\rangle)$ [37, 38]. However, this reward fails to learn—the cumulative reward obtained from this reward function does not reflect maximum *final* fidelity, which is our true goal. The agent might, for example, learn to stay close to the target without actually terminating the episode. Concretely, an agent in state $(|00\rangle + |11\rangle)/\sqrt{2}$ can apply a CNOT gate repeatedly, always staying at a fidelity of $1/2$ to target state $|00\rangle$; this is optimal for the agent. Indeed, we observe precisely this style of behavior when training with this reward (cf. Figure 5). To force the return and final fidelity to have the same maximizing policy, we can use reward-shaping [43] which yields the incremental fidelity $r(s_i, a_i, s_{i+1}) = \gamma\Phi(s_{i+1}) - \Phi(s_i)$ (γ is the discount parameter). However, we find that this also does not learn (cf. Figure 5). Roughly, the key bottleneck here is that the cumulative reward fails to capture the visiting of states s_j with large $\Phi(s_j)$ at intermediate points the trajectory since it only involves the start and end states. Since the value and policy functions are updated according to these sample returns, the agent fails to capitalize on partial paths taken to high-fidelity intermediate states. These high-fidelity intermediate states are rare and crucial for the policy to improve.

Why are they rare? A gate (H, S, HSH or CNOT) applied to a state $|\psi_i\rangle$ is very likely not to increase the fidelity to $|\mathbf{0}\rangle$. Indeed, we picked 1000 uniformly sampled 6-qubit states, applying every one of 48 gates induced from our gateset to each state. We found that 83.4% of the actions yielded no change in fidelity, 10.0% reduced fidelity and only 6.6% improved it. Thus, it is very unlikely during the (critical) early stages of training, that states with large fidelity are reached since it would involve multiple increases along the path.

Further, the reward is often negative: during the early stages of training, the final fidelity upon truncation of the episode concentrates around the expected change in fidelity, which is ≈ 0 as the start and end states are essentially distributed uniformly (the step limit T^* is typically sufficient for mixing). Hence, the cumulative reward G_i from the ith state, given by $G_i = \gamma^{T^*}\Phi(s_{T^*}) - \Phi(s_i)$ (via telescoping), is likely negative (empirically, 94.2% of the time).

We conclude that to provide useful signal to the learning agent, the reward must reward actions that led to a high-fidelity state at some step ahead, and should be positive whenever progress is made (say, the fidelity increased beyond the best-seen fidelity so far at least once across the course of the trajectory). The reward that we define is motivated by and addresses these two crucial requirements.

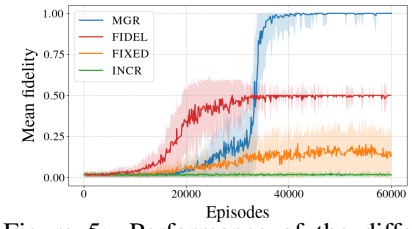


Figure 5: Performance of the different reward functions discussed towards training a 6-qubit full-connectivity agent. FIXED corresponds to the fixed-penalty reward; FIDEL corresponds to using the fidelity as reward; INCR is the incremental fidelity $\gamma\mathcal{F}(s_{i+1}, |\mathbf{0}\rangle) - \mathcal{F}(s_i, |\mathbf{0}\rangle)$; MGR is the reward introduced in this work, and is the only one that succeeds.

To provide useful signal to the learning agent, the reward function must reward actions that led to a high-fidelity state at some step ahead, and should be positive whenever the agent makes new progress ahead in the trajectory, say by visiting a state with better fidelity than seen previously in the trajectory. To achieve this, we must remember when the maximum-fidelity state of the trajectory occurred and assign positive cumulative return to actions that came before, and non-positive cumulative return to those that came after. We say cumulative return and not instantaneous reward because the probabilities of choosing actions are determined by the sign and magnitude of the cumulative

returns from that step onward. In our case, $\gamma \approx 1$ so future rewards are almost equally valued as the current.

Let $G_i = \sum_{j=i}^{T-1} \gamma^{j-i} r(s_j, a_j, s_{j+1}$ denote the cumulative return from step i using which the utility of action a_i is updated. We choose the reward r such that G_i is positive for each action until the maximum-fidelity state of the trajectory is reached, with value proportional to the increase in fidelity with respect to states seen before i. This leads, by noting that $r(s_i, a_i, s_{i+1}) = G_i - \gamma G_{i+1}$, that we may choose

$$r(s_i, a_i, s_{i+1}) = \gamma M_{i+1} - M_i, \tag{6}$$

where $M_i := \max_{j \leq i} \mathcal{F}(s_j, |\mathbf{0}\rangle)$ is the maximum fidelity seen until now. As described in the main paper, this is indeed the reward we use. We define it formally below for completeness.

Definition C.1. [MGR reward] The MGR reward at step i of a trajectory is given by

$$r(s_i, a_i, s_{i+1}) := \gamma M_{i+1} - M_i,$$

where $M_i := \max_{0 \leq k \leq i} \Phi(s_k)$ with $\Phi(\cdot)$ the fidelity, and γ is the discount parameter ≈ 1.

The reward can be equivalently expressed as

$$r(s_i, a_i, s_{i+1}) = \begin{cases} \gamma \Phi(s_{i+1}) - M_i & \Phi(s_{i+1}) > \Phi(s_k) \, \forall k \leq i \\ (\gamma - 1) M_i & \text{otherwise} \end{cases}$$

which expresses the fact that rewards are proportional to new progress made, as motivated.

The cumulative reward from step i to the terminal step T then takes a particularly neat form, consistent with our hope that it is positive for each state reached until the maximum-fidelity state on the path with magnitude proportional to the difference of fidelities to $|\mathbf{0}\rangle$ between $|\psi_i\rangle$ and the maximum-fidelity state:

Lemma C.2. *Let r denote the MGR reward as defined by Definition C.1. Then, the cumulative reward from step i to the terminal step T*

$$G_i \equiv \sum_{j=i}^{T} \gamma^{j-i} r(s_j, a_j, s_{j+1})$$

can be expressed as

$$G_i = \gamma^{T-i} M_T - M_i,$$

where γ and M_i are as defined in Definition C.1.

Proof. The key idea is the finite-difference-style formulation of the reward, which lends itself to a telescoping sum for the cumulative reward. We have

$$\begin{aligned} G_i &= \sum_{j=i}^{T} \gamma^{j-i} r(s_j, a_j, s_{j+1}) \\ &= \sum_{j=i}^{T} \gamma^{j-i} (\gamma M_{j+1} - M_j) \\ &= \sum_{j=i+1}^{T} \gamma^{j-i} M_j - \sum_{j=i}^{T} \gamma^{j-i} M_j \\ &= \gamma^{T-i} M_T - M_i, \end{aligned}$$

as needed. □

Essentially, the reward for the ith action is positive if the $(i+1)$-th state s_{i+1} has fidelity higher than that seen so far [2] and is slightly negative if not. The cumulative reward compounds positive rewards after step i; Lemma C.2 proves that the cumulative reward is exactly $G_i = \gamma^{T-i} M_T - M_i$

[2] In fact, it will have fidelity at least *twice as much* as seen before, by the well-known fact that stabilizer state fidelities are always an inverse power of 2 [17].

– positive in proportion to the maximum fidelity across the trajectory M_T, and non-positive after a maximum-fidelity state is reached since $M_i = M_T$ from thereon.

The episodic return that is the agent's goal to maximize is then $J \equiv G_0 = \gamma^T M_T - M_0$. Since $M_0 = \mathcal{F}(s_0, |\mathbf{0}\rangle)$ is a stochastic quantity independent of the policy, the policy essentially seeks to maximize M_T or achieve $M_T = 1$ corresponding to successful preparation (at which point the episode ends). The return is also maximized when γ^T is maximum, or when the episode length T is as small as possible. Thus, the agent is incentivized to prepare correct circuits, and then do them quickly. In the beginning, when no episode is able to succeed, we artificially terminate episodes at a large maximum-timestep T^* to facilitate learning. Once $M_T = 1$ is often reached from randomly sampled initial states, the agent begins to optimize gate usage along with M_T and achieves convergence with $M_T = 1$ almost always and an average episode size T much smaller than T^*.

D Experimental Setup and Hyper-parameters

Table 3 details the exact RL framework used for stabilizer state preparation. The state space $\mathcal{S}$ is the set of all n-qubit states. The agent picks actions from the *inverted* action set $\mathcal{A}^\dagger = \{a^{-1} : a \in \mathcal{A}\}$. The start state for each episode is drawn uniformly from $\mathcal{S}$ and target states are those which are ϵ-close to the fiducial $|\mathbf{0}\rangle$. A successful trajectory leads to a circuit C with $\mathcal{F}(|\mathbf{0}\rangle, C|\psi\rangle) = |\langle|\mathbf{0}\rangle |C|\psi\rangle|^2 \geq 1 - \epsilon$. Further notice that

Table 3: The proposed RL framework for stabilizer state preparation.

Component	Description
State Space $\mathcal{S}$	The set of n-qubit quantum states to be prepared.
Action Space $\mathcal{A}$	The *inverse* of all gates in the induced collection of n-qubit gates.
Transition Function $p(s, u, s')$	Deterministic: $s' = u \cdot s$ if action $u \in \mathcal{A}$ is applied to state $s \in \mathcal{S}$.
Starting Distribution μ	Uniform over $\mathcal{S}$.
Terminal state	$|\psi^*\rangle \equiv |\mathbf{0}\rangle$.

Setup used in the experiments. In this work, we primarily demonstrate the performance and scaling behavior of our algorithm by preparing stabilizer states, i.e. $\mathcal{S} = \mathbb{S}_n$, the set of n-qubit stabilizer states. Each stabilizer state is represented in flattened canonical tableau form [18], so that each state is represented by a $(2n^2 + n)$-dimensional bit-vector of stabilizers. We set $\epsilon = 0$, i.e. we target *exact* stabilizer state preparation.

We work with two qubit connectivities at opposite ends of the connectedness spectrum: (a) the fully-connected connectivity with coupling map being the complete graph on the set of qubits, and (b) the linear/local connectivity with the edge set of the coupling map being the path $\{\{i, i+1\} : 1 \leq i \leq n-1\}$.

We use different allowed gatesets for each connectivity. For the fully-connected case, our gateset G comprises the H (Hadamard), S (Phase), CNOT and HSH (conjugated phase) gates. The inclusion of the conjugated phase gate HSH ensures symmetry within the gate set because it provides an operation for the X component that mirrors the effect of S on the Z component. While S modifies the Z component of the tableau, the HSH gate equivalently modifies the X component. This symmetry is useful since the tableau is also symmetric in X and Z. For the linear connectivity, we use the H, S, CNOT and X, Y, Z gates. Both gate-sets are realistic; for example, they can be easily implemented on trapped-ion-based quantum computers [44], a promising candidate for quantum computation. It is also not unfair to treat HSH as a single gate; it is as easy as S to apply [45] – note that HSH is simply a $\pi/2$-rotation about the x-axis just as S is a $\pi/2$-rotation about the z-axis.

Random stabilizer circuits are sampled using the Stim API [46]. Each sampled brickwork circuit is constructed by choosing independently sampled random 2-qubit stabilizer circuits as "bricks" which are then layered as in Figure 1(a) to the required depth. The uniform density on the space of general 3-qubit states is the Haar measure; Haar-random general 3-qubit states for our experiments are sampled using the IBM Qiskit API [20].

We implement a version of PPO based on [47] in `PyTorch` and simulate stabilizer states using Stim [46]. The environment is vectorized for parallel training on a single GPU.

To facilitate quick learning, especially at the start of training, we restrict episodes to a maximum step-count T^*. The pairs (n, T^*) used in our experiments are $(5, 50)$, $(6, 80)$, $(7, 80)$ and $(9, 127)$. This is especially helpful considering that each gate in our gate-set has an order of at most 4 - four consecutive repetitions yield the identity. In the early phases of training, such repetitions by the agent are common; artificially stopping episodes and increasing PPO's entropy loss parameter to force more uniform action selection helped us counter this effect.

Also, for the linear-connectivity agents, we used an interpolation of the MGR reward and the difference between the Jaccard distances (to $|\mathbf{0}\rangle$'s stabilizer tableau) of the stabilizer tableaus of adjacent states (the latter of which is used in [40]). We found that training is faster with this term.

Additional hyper-parameters used that are part of the PPO algorithm [31] are presented in Table 4.

Table 4: PPO hyper-parameters used in training.

Hyper-parameter	Value
Learning rate (policy)	0.0003
Learning rate (value)	0.0005
Num. optimization epochs	8
Minibatch size	256
GAE parameter (λ)	0.95
policy_optimization_epochs	8
policy_clip_range	0.2
value_optimization_epochs	8
value_clip_range	∞
entropy_loss_weight	0.01

All policy and value networks used had two hidden layers of 512 nodes each. The discount parameter was set to $\gamma = 0.99$.

Computational costs of training. Training for $40,000$ episodes with $n = 5$ qubits took 20 minutes on a single Tesla V100 GPU. At $n = 7$, training took 2.5 hours (130k episodes) and at $n = 9$ took 8 hours (160k episodes) on the same GPU.

E Proofs

Let S be a collection of quantum states. Define a property (such as "the state is prepared correctly with a circuit containing at most 34 gates") by a function $P : \{0,1\}^* \times S \to \{0,1\}$ taking as input a proposed state preparation model T and a state $|\psi\rangle \in S$, and outputting 1 if the property is satisfied when T is run on $|\psi\rangle$ and 0 otherwise.

Now, given a trained agent T, we seek to show that some desired properties hold over most states $|\psi\rangle \in S$. Of course, given the large state space S, it is nearly impossible to examine all states, even for a small number of qubits. Instead, we shall resort to randomized methods (risking a tiny probability of erroneous results), uniformly sampling a large number of states from S and testing the property P on these states, following which we appeal to concentration results to render our sampled results almost exact for the whole state space S. This is formalized in the following theorem.

Theorem E.1 (Randomized Property Testing)**.** *Let $P : \{0,1\}^* \times S \to \{0,1\}$ be a property. Let ε, δ be small error parameters and let the number of samples N satisfy $N \geq \frac{1}{2\varepsilon^2} \log\left(\frac{1}{\delta}\right)$. Denote the uniform measure on S by $\mathbb{U}(S)$ and let $|\psi_1\rangle, |\psi_2\rangle, \cdots, |\psi_N\rangle \overset{i.i.d}{\sim} \mathbb{U}(S)$ be sampled i.i.d from the uniform measure. Then with probability at least $1 - \delta$ over the choice of samples $\{|\psi_i\rangle\}_{i=1}^N$,*

$$\Pr_{|\psi\rangle \sim \mathbb{U}(S)} \left[P(T, |\psi\rangle) = 1\right] \geq \frac{\sum_{i=1}^N P(T, |\psi_i\rangle)}{N} - \varepsilon.$$

Proof. Define the Bernoulli random variable $X : S \to \{0,1\}$ by $X(|\psi\rangle) := P(T, |\psi\rangle)$ with $|\psi\rangle \sim \mathbb{U}(S)$. Notice that $X_i := P(T, |\psi_i\rangle)$ for $i = 1$ to N are thus sampled i.i.d according to the

distribution of X. Since $0 \leq X \leq 1$, Hoeffding's inequality gives

$$\Pr_{(|\psi_1\rangle,\cdots,|\psi_N\rangle)\sim\mathbb{U}(S)^{\otimes N}}\left[\sum_{i=1}^{N} X_i \geq N\mathbb{E}X + N\varepsilon\right] \leq \exp\left(-\frac{2(N\varepsilon)^2}{\sum_{i=1}^{N} 1}\right) = \exp\left(-2N\varepsilon^2\right).$$

Simplifying yields

$$\Pr\left[\mathbb{E}X \geq \frac{\sum_{i=1}^{N} X_i}{N} - \varepsilon\right] \geq 1 - \exp\left(-2N\varepsilon^2\right) \geq 1 - \delta,$$

where the last inequality follows from $N \geq \frac{1}{2\varepsilon^2}\log\left(\frac{1}{\delta}\right)$. Finally, since $X \in \{0,1\}$, note that $\mathbb{E}X = \Pr\left[X = 1\right] = \Pr_{|\psi\rangle\sim\mathbb{U}(S)}\left[P(T,|\psi\rangle) = 1\right]$. Setting $X_i = P(T,|\psi_i\rangle)$, it follows that with probability at least $1-\delta$ over the choice of samples $\{|\psi_i\rangle\}_{i=1}^{N}$,

$$\Pr_{|\psi\rangle\sim\mathbb{U}(S)}\left[P(T,|\psi\rangle) = 1\right] \geq \frac{\sum_{i=1}^{N} P(T,|\psi_i\rangle)}{N} - \varepsilon.$$

□

With the theorem in hand, we were able to prove some rather remarkable properties about our trained agents T (fully-connected architecture, $n = 5, 6, 7, 9$). In every case, S is the set of n-qubit stabilizer states. We choose $\delta = e^{-10} \leq 0.00005, \varepsilon = 10^{-3/2} \approx 0.0316$ and $N = 5000$. Initially, we set

$$P(T,|\psi\rangle) \equiv 1 \text{ iff } T(|\psi\rangle) \text{ returned a circuit preparing } |\psi\rangle \text{ correctly.}$$

We sampled N states and computed the quantity $\overline{P} \equiv \sum P(T,|\psi_i\rangle)/N$. The outcome and corresponding lower bounds on $\Pr\left[P(T,|\psi\rangle) = 1\right]$ guaranteed by the Theorem are displayed below.

Number of qubits	$n = 5$	$n = 6$	$n = 7$	$n = 9$
$\overline{P}$	1.0	1.0	1.0	0.9976
$\Pr_{\|\psi\rangle\sim\mathbb{U}(S)}\left[P(T,\|\psi\rangle) = 1\right] \geq$	0.9684	0.9684	0.9684	0.9660

The results indicate that in every case,

$$\Pr_{|\psi\rangle\sim\mathbb{U}(S)}\left[P(T,|\psi\rangle) = 1\right] \geq 0.965.$$

When the state space S is finite, the left-hand-side probability is simply the number of states satisfying P over $|S|$, so we get that the number of states prepared successfully by the agent is at least 96.5% of the full stabilizer state space. This is quite remarkable, given that the agent sees a tiny fraction of the state space (around 1 in 1 billion) during the entirety of its training and has generalized to almost the whole space despite this.

For the agent trained to prepare arbitrary 3-qubit general states to a fidelity of at least 0.99, we sampled $N = 5000$ states again and used the same ε and δ. We found that every run was successful, so $\overline{P} = 1.0$. This means, by the theorem, that

$$\Pr_{|\psi\rangle\sim\mathbb{U}(S)}\left[P(T,|\psi\rangle) = 1\right] \geq 1 - \varepsilon \geq 0.9684,$$

or at least 96.5% of the states in the 3-qubit state space will be prepared correctly by the agent. With a covering number of around 10^{31}, this means that the agent will prepare $> 9.6 \times 10^{30}$ states after training on just 1.2×10^7 states, which implies a huge generalization on the part of the agent, on the order of correctly preparing 10^{23} states per training state seen!

Remark. It is important to realize why one should expect such a magnitude of generalization. We propose an explanation for this. Consider the 9-qubit Clifford state space as a graph with edges between states that can be prepared from each other using a single gate. Since there are $n^2 + 3n$ gates, this is around 10^2 at $n = 9$ – a large number. This means that one can reach an order of 100^k states from a single state using k gates. Intuitively, this means that with a "slack" of being k gates from optimal, the agent can basically learn to prepare only $|S|/100^k$ states, and still be able to prepare

the rest of the states in the space with at most k gates more. So a 10^9-factor of generalization is not unreasonable, and corresponds to a slack of $4-6$ gates (not too bad when the optimal is around 45-50, only 10% from optimal, worst-case). Similarly, for the 3-qubit general case (here the gate-set has size 18, so the slack is around $23\log_{18}(10)) \approx 20$ (also pretty good with the optimal being around 250). The main contribution of the agent is to apply the correct slack gates at the appropriate places in a way that keeps it along a path of states that it knows how to prepare from training.

Remark. We believe we can improve the guarantees by one or two percent by sampling more states; however, we believe that the current guarantees serve sufficiently well to indicate the power of the agents in our state preparation framework.

We now turn towards a worst-case guarantee on the size of circuits returned by the agents. On n-qubits, we work with properties of the form

$$P(T, |\psi\rangle) \equiv 1 \text{ iff } T(|\psi\rangle) \text{ returned a circuit preparing } |\psi\rangle \text{ correctly using } \textit{at most } T_n \text{ gates.}$$

Notice that a bound of the form $\Pr_{|\psi\rangle\sim\mathbb{U}(S)}[P(T, |\psi\rangle) = 1] \geq p$ means that at least a p-fraction of the state space *will* be prepared correctly and with a circuit of size at most T_n. This is a much stronger guarantee than the previous one! (Of course, such a guarantee with p close to 1 is only possible if the distribution of the circuit sizes across the state space is small-tailed, which we observed it to be). The numbers T_n are chosen to be such that the tail beyond T_n occupies at most 1.5% of the weight, yielding $p > 0.95$ for each n. The details are shown in the following table.

Number of qubits	$n=5$	$n=6$	$n=7$	$n=9$
T_n	22	29	32	56
$\overline{P}$	0.9884	0.9842	0.9840	0.9832
$\Pr_{\|\psi\rangle\sim\mathbb{U}(S)}[P(T, \|\psi\rangle) = 1] \geq$	0.9567	0.9525	0.9523	0.9515

The table shows that in each case,

$$\Pr_{|\psi\rangle\sim\mathbb{U}(S)}[P(T, |\psi\rangle) = 1] \geq 0.95,$$

or in other words, at least 95% of the stabilizer state space *will* be prepared using circuits of size at most T_n gates. We believe that this is the first such worst-case guarantee for state preparation agents trained using learning techniques.

F Additional experiments

F.1 Two-qubit Gate Count

We find that our heuristics use expensive entangling gates efficiently despite never being biased to do so.

CNOT gate count. Our metric for circuit size is the total number of gates, with both one and two-qubit gates counted as one unit each. However, since two-qubit gates are often noisier and more expensive to apply than single-qubit gates, we check our agents to examine the CNOT count, to see if we receive an additional benefit of smaller CNOT counts for free.

We benchmark our trained agents using the CNOT gate count as the metric of performance. Note that our agents are never explicitly trained to minimize two-qubit gates, and are trained with one and two-qubit gates placed on an equal footing. However, as our experiments on entanglement dynamics show (cf. Figure 1(b)), the agent's actions do not display much redundancy and monotonically increase entanglement—so one might expect good usage of the entangling CNOT gate, especially from the linear-connectivity agent. Indeed, that is so: the results are shown in Table 5. Note that the two baseline methods use the two-qubit entangling SWAP gate in addition to CNOT.

Table 5 shows that we perform well, sometimes better than the optimized [19] algorithm with respect to CNOT gates, despite having given no bias towards minimizing the number of two-qubit gates. This further emphasizes our efficiency in zero-shot state preparation.

Table 5: Average number of CNOT gates (↓) used by different algorithms across 200 randomly sampled uniform stabilizer states.

Two-qubit gate count →)	5-qubit	6-qubit	7-qubit	9-qubit
[18]	9.12 ± 3.29	14.92 ± 3.85	21.34 ± 4.16	38.22 ± 5.46
[19]	7.56 ± 2.46	11.89 ± 2.81	$\mathbf{16.30} \pm 2.86$	$\mathbf{26.46} \pm 3.17$
RL (linear connectivity)	10.16 ± 4.16	14.50 ± 7.34	18.44 ± 4.10	33.51 ± 4.59
RL (full connectivity)	$\mathbf{6.08} \pm 2.45$	$\mathbf{9.13} \pm 2.28$	19.52 ± 7.71	34.20 ± 13.48

F.2 Attacking the general problem: general 3-qubit state preparation

Parameter	**Value**
Gateset	H, S, Z, T single-qubit gates; CNOT gates (full connectivity)
Episodes	30,000
Steps per episode (T^*)	350
Policy and Value network sizes	3 hidden layers of size 512 each
Discount parameter γ	0.95

Table 6: training settings for general 3-qubit state preparation.

Table 6 presents the experimental setup used in the PPO algorithm to learn a heuristic for 3-qubit general state preparation. The Solovay-Kitaev theorem [21] guarantees that for each $\epsilon > 0$ and arbitrary n-qubit state $|\psi\rangle$, a finite (exponential) number of gates from our gate-set suffice to prepare a circuit, which when applied to the default state $|\psi^*\rangle$ yields a state ϵ-close to $|\psi\rangle$.

In our experiments, we set the tolerance to $\epsilon = 0.01$, indicating that the heuristic succeeds (or equivalently, the RL episode reaches a terminal state) iff the fidelity to the target state exceeds 0.99. Notice that general states can require substantially more gates than stabilizer states to prepare: the upper bound for general states is $\mathcal{O}(4^n)$ gates [6], while it is $\Theta(n^2/\log n$ for stabilizer states [18]. One can think of this result as a consequence of the fact that general states require $\approx 2^{n+1}$ independent real numbers $\in [0, 1]$ to describe, while stabilizer states only require $2n^2 + n$ independent bits to describe.

The algorithm and reward function used in our experiment is identical to that used in all stabilizer experiments. We tested the learned heuristic after $30,000$ episodes using $1,500$ fresh Haar-randomly sampled 3-qubit states $\in \mathbb{C}^9$.

Each state was prepared successfully with the desired fidelity of $> 99\%$ on the agent's first attempt. The mean gate count was 286.51 ± 78.87 and the median gate count 278. Mean fidelity was 0.9916 ± 0.011. The full statistics are displayed in Figure 2.

F.3 Preparation of code states

We use the fully-connected and linear agents to prepare the logical code states $|0\rangle_L$ and $|1\rangle_L$ of some typical quantum error-correcting codes (QECs) [48, 49, 50]. Figure 6, Figure 7 and Figure 8 show the circuits prepared by full-connectivity agents for three very popular codes.

Note that in each case, the agent was never explicitly trained to prepare any of these states.

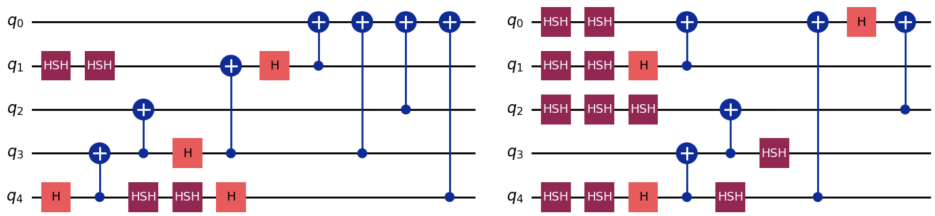


Figure 6: Preparing logical states $|0\rangle_L$ (left) and $|1\rangle_L$ (right) of the $[5, 1, 3]$ perfect code [48], full connectivity.

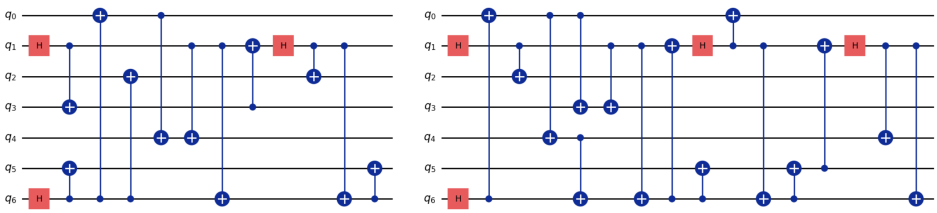


Figure 7: Preparing logical states $|0\rangle_L$ (left) and $|1\rangle_L$ (right) of the $[7, 1, 3]$ CSS code [49], full connectivity.

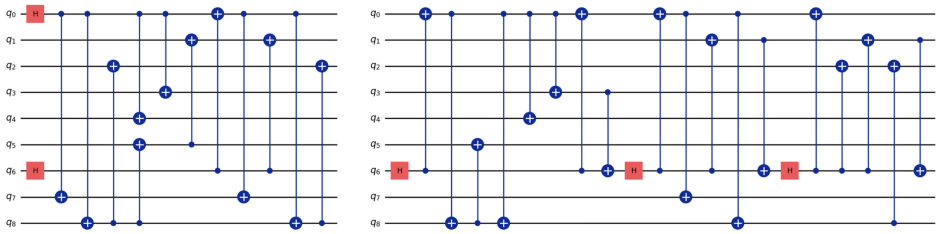


Figure 8: Preparing logical states $|0\rangle_L$ (left) and $|1\rangle_L$ (right) of the $[9, 1, 3]$ Shor code [50], full connectivity.