

Fast stabilizer state preparation via AI-optimized graph decimation

Michael Doherty^{1,†}, Matteo Puviani^{2,†}, Jasmine Brewer^{2,†}, Gabriel Matos²,
David Amaro², Ben Criger^{2,3}, David T. Stephen²

¹UCL; ²Quantinuum; ³IGDORE; [†]Equal contribution

Abstract

Efficient stabilizer state preparation is essential in quantum algorithms, measurement-based quantum computation, and quantum error correction. We introduce graph decimation, a scalable formulation of stabilizer-state circuit synthesis as a sequential decision-making problem on graph states and a compact classical learning environment for quantum circuit synthesis. The method uses exact graph transformation rules to remove graph edges using CX, CY, and CZ gates, then reverses this sequence to obtain a preparation circuit. We compare greedy search, beam search, model-free reinforcement learning (RL), and an RL-guided Monte Carlo tree search (MCTS) method, *QuSynth*. Across logical code states from 9 to 144 physical qubits, the best found circuits match or improve published best circuits where available. For larger codes in this range, for which we are not aware of published results, the best circuits we find reduce two-qubit gate count by up to a factor of 2.5 relative to LC-optimized CZ-only baselines while retaining low circuit depth.

1 Motivation

Stabilizer state preparation is a common procedure in quantum computation and communication. These states are especially important as logical states and ancilla states in quantum error correction [1]. Preparing them with low two-qubit gate count and low depth is a practical compilation problem: any reduction in entangling gates lowers exposure to gate faults, decoherence, and scheduling overhead.

Existing exact and optimal synthesis approaches have produced strong results for small stabilizer states, but scale poorly beyond tens of qubits [2, 3]. Recent machine-learning approaches have shown promise for discovering more efficient preparation circuits [4–6], but have generally been restricted to substantially smaller instances than the stabilizer states of the larger quantum error-correcting codes of current interest. A central challenge is to choose a representation in which useful local decisions expose global circuit savings.

Contributions. We formulate stabilizer-state preparation as graph decimation: a finite-horizon search problem whose states are graphs and whose actions are two-qubit Clifford gates. This is a machine-learning-for-quantum-systems problem in which policies are learned over exact stabilizer graph transformations, and the same environment supports fast heuristic search, RL, and RL-guided MCTS. We benchmark QEC logical states up to 144 qubits, extending AI-assisted stabilizer synthesis beyond the size regime of prior methods. Across 9-144 physical-qubit logical code states, our best circuits match or improve published best circuits where available. For larger codes, where we are not aware of published results, they reduce two-qubit gate count by up to $2.5\times$ versus LC-optimized CZ-only baselines while keeping low depth.

2 Graph Decimation

Every stabilizer state is local-Clifford equivalent to a graph state [7]. A naive graph-state preparation applies one CZ gate per graph edge. Here, we synthesize the graph using CX and CY Clifford gates in addition to CZ gates, in an all-to-all logical cost model where each counts as one two-qubit Clifford gate. A $CX_{i,j}$ gate with control i and target j has the same graph-level effect as complementing all edges between i and the neighborhood $n(j)$. Similarly, $CY_{i,j}$ complements those edges together with the edge (i, j) . A single two-qubit gate can therefore remove several graph edges and replace a collection of commuting CZ gates by one Clifford operation.

Graph decimation applies these rules in reverse. Starting from the target graph, an algorithm repeatedly chooses a CZ, CX, or CY action that transforms the graph, aiming to reach the empty graph in as few actions as possible. Reversing the resulting action sequence gives a preparation circuit for the target state. This reverse formulation is useful because the initial graph exposes high-degree neighborhoods and the largest potential gate savings. The state is represented by its adjacency matrix, actions are Clifford gates, the terminal state is the edge-free graph, and the main objective is final two-qubit gate count, with depth secondary. Exhaustive search is infeasible because the branching factor is quadratic in the number of qubits and the circuit search space grows exponentially in the number of gates. For the CSS logical states reported here, bipartite graph representatives allow the mixed graph-decimation circuits to be rewritten, up to single-qubit Cliffords, as CX-only circuits using $|0\rangle/|+\rangle$ preparations.

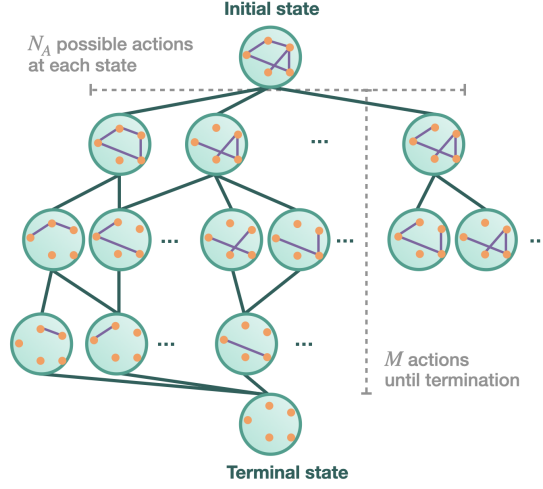


Figure 1: Graph decimation as a sequential decision problem. The root is the target graph state, and each branch corresponds to a two-qubit Clifford action that changes the graph. Search and learning methods differ in how they choose which branch to explore, so the same environment supports greedy search, beam search, PPO, and the RL-guided MCTS method *QuSynth*.

3 AI and Search Methods

We compare four solvers for the same graph decimation problem:

1. Greedy best-first search (BeFS) selects the highest-scoring action according to edge-removal and circuit-layer penalties, using tie-breaking heuristics for equal-score actions.
2. Beam search retains multiple candidate graph states per step, which lets it accept locally worse moves that can lead to shorter global solutions. This matters because optimal decimation trajectories sometimes temporarily preserve or even increase edge count before enabling larger future removals. Both search methods are fast and benefit from vectorized JAX implementations.
3. A model-free reinforcement learning PPO agent [8] that maps graph observations to actions and learns from self-play, with invalid or non-improving actions masked where appropriate to keep episodes terminating.
4. *QuSynth*, which combines reinforcement learning with Monte Carlo tree search [9] in the style of agent-guided planning [10]: the network predicts action priors and state values, while a shallow tree search selects the next move. This planning step can improve credit assignment over longer horizons, at the cost of larger per-move compute.

4 Results

We benchmark on logical stabilizer states of CSS and non-CSS quantum error-correcting codes. Initial graph representatives are obtained using the *Stim* stabilizer toolkit [11]; for larger codes, we also test graph decimation after local-Clifford preprocessing. Here, LC-optimized or LC-preprocessed means that we first search the local-complementation orbit of the graph state to find an equivalent graph with fewer edges and lower CZ-only preparation cost. For small codes we compare with published best or optimal circuits; for larger codes, where such external baselines are unavailable, we compare with the best LC-optimized CZ-only preparation we found. Because the rules are exact graph-state identities, any decimation path ending at the empty graph certifies a preparation circuit for the original stabilizer state, up to tracked local Cliffords.

Table 1 summarizes representative best-found results under finite compute budgets. For the smallest codes, graph decimation matches the best known SAT-based circuits and sometimes lowers depth. For the Golay code, *QuSynth* gives the lowest two-qubit gate count, while beam search and model-free RL recover a 45-gate, depth-7 circuit. For larger codes, RL and *QuSynth* usually outperform greedy search, although the best method depends on the instance. The largest example is the 144-qubit Gross code [12], where model-free RL finds a 357-gate preparation after LC preprocessing, a roughly $2.5\times$ reduction relative to the LC-optimized CZ-only baseline and roughly $5\times$ relative to the unoptimized graph representative.

Target logical state	Baseline (G_2, D_2)	Best found (G_2, D_2)	Method
[[9, 1, 3]] surface code	(8, 3) ¹	(8, 3)	BeFS
[[23, 1, 7]] Golay code	(56, 17) ²	(44, 8)	<i>QuSynth</i>
[[36, 8, 6]] code	(124, 11+1) ³	(77, 11)	<i>QuSynth</i>
[[40, 20, 6]] non-CSS code	(284, 21+1) ³	(130, 24)	<i>QuSynth</i>
[[42, 12, 5]] code	(131, 13+1) ³	(90, 9)	<i>QuSynth</i>
[[64, 12, 8]] code, LC-preprocessed	(298, 19+1) ³	(161, 13)	<i>QuSynth</i>
[[144, 12, 12]] Gross code, LC-preprocessed	(876, 37+1) ³	(357, 18)	RL

Table 1: Representative two-qubit gate counts G_2 and depths D_2 . Baselines are prior best circuits for small codes and LC-optimized CZ-only preparations for larger codes. Baseline superscripts denote ¹ Ref. [2], ² Ref. [3], and ³ LC-optimized CZ-only graph-state preparation. For D_2 entries that include a +1, the reported D_2 is the graph maximum degree; the exact depth is D_2 or $D_2 + 1$ by Vizing’s theorem.

The methods occupy different points on a compute-quality tradeoff. Greedy search and beam search often find strong circuits in seconds to minutes. As a timing reference, the accompanying data for Ref. [3] reports 8025.7 seconds for its 56-gate Golay A* run [13], whereas our BeFS implementation finds a 46-gate Golay circuit in under one second on an NVIDIA A100 GPU. RL and *QuSynth* require more compute because they currently train a separate agent for each code, but they give the best gate counts in most non-small instances; the Gross-code RL run, for instance, takes hours rather than seconds. This cost is acceptable for one-time offline synthesis of a reusable preparation circuit.

5 Impact

Graph decimation scales beyond the range of current optimal stabilizer synthesis, gives large reductions in two-qubit gates for QEC logical states, and can support constraints such as depth, connectivity, or fault-tolerance-oriented verification. The circuits reported here are not by themselves fault-tolerant preparation protocols; rather, they provide low-entangling-gate seeds for verification, flagging, or purification. More broadly, the same graph-based formulation can be adapted to measurement-based resource states, Clifford isometries, and simultaneous diagonalization subroutines used in Pauli-rotation synthesis.

References

- [1] Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.
- [2] Tom Peham, Ludwig Schmid, Lucas Berent, Markus Müller, and Robert Wille. Automated synthesis of fault-tolerant state preparation circuits for quantum error-correction codes. *PRX Quantum*, 6: 020330, May 2025. doi: 10.1103/PRXQuantum.6.020330. URL <https://link.aps.org/doi/10.1103/PRXQuantum.6.020330>.
- [3] Mark Webster, Stergios Koutsoumpas, and Dan E Browne. Heuristic and optimal synthesis of cnot and clifford circuits, 2025. URL <https://arxiv.org/abs/2503.14660>.
- [4] Jan Olle, Remmy Zen, Matteo Puviani, and Florian Marquardt. Simultaneous discovery of quantum error correction codes and encoders with a noise-aware reinforcement learning agent. *npj Quantum Information*, 10(1), December 2024. ISSN 2056-6387. doi: 10.1038/s41534-024-00920-y. URL <http://dx.doi.org/10.1038/s41534-024-00920-y>.
- [5] 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. *Phys. Rev. X*, 15:041012, Oct 2025. doi: 10.1103/gqpr-dgz7. URL <https://link.aps.org/doi/10.1103/gqpr-dgz7>.
- [6] Sara Giordano, Kornikar Sen, and Miguel A. Martin-Delgado. Hybrid reward-driven reinforcement learning for efficient quantum circuit synthesis. *Quantum Machine Intelligence*, 8(1), February 2026. ISSN 2524-4914. doi: 10.1007/s42484-026-00359-8. URL <http://dx.doi.org/10.1007/s42484-026-00359-8>.
- [7] Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. Graphical description of the action of local clifford transformations on graph states. *Phys. Rev. A*, 69:022316, Feb 2004. doi: 10.1103/PhysRevA.69.022316. URL <https://link.aps.org/doi/10.1103/PhysRevA.69.022316>.
- [8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- [9] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.
- [10] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL <https://www.science.org/doi/abs/10.1126/science.aar6404>.
- [11] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. ISSN 2521-327X. doi: 10.22331/q-2021-07-06-497. URL <https://doi.org/10.22331/q-2021-07-06-497>.
- [12] Sergey Bravyi, Andrew W. Cross, Jay M. Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, March 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07107-7. URL <http://dx.doi.org/10.1038/s41586-024-07107-7>.

- [13] Mark Webster. Cliffordopt, 2025. URL https://github.com/m-webster/CliffordOpt/blob/4663db0d38eb1a8bb40d8e91308a75c5a5a9d40c/paper_results/RL_SAT_encoding.xlsx. Sheet as-tar, cell F65.