Equivariant Reinforcement Learning for Clifford Quantum Circuit Synthesis

Richie Yeung[1,2] **Aleks Kissinger**[1] **Rob Cornish**[3]
[1]Department of Computer Science, University of Oxford [2]Quantinuum
[3]College of Computing and Data Science, Nanyang Technological University, Singapore

Abstract

We consider the problem of synthesizing Clifford quantum circuits for devices with all-to-all qubit connectivity. We approach this task as a reinforcement learning problem in which an agent learns to discover a sequence of elementary Clifford gates that reduces a given symplectic matrix representation of a Clifford circuit to the identity. This formulation permits a simple learning curriculum based on random walks from the identity. We introduce a novel neural network architecture that is equivariant to qubit relabelings of the symplectic matrix representation, and which is size-agnostic, allowing a single learned policy to be applied across different qubit counts without circuit splicing or network reparameterization. On six-qubit Clifford circuits, the largest regime for which optimal references are available, our agent finds circuits within one two-qubit gate of optimality in milliseconds per instance, and finds optimal circuits in 99.2% of instances within seconds per instance. After continued training on ten-qubit instances, the agent scales to unseen Clifford tableaus with up to thirty qubits, including targets generated from circuits with over a thousand Clifford gates, where it achieves lower average two-qubit gate counts than Qiskit's Aaronson-Gottesman and greedy Clifford synthesizers.

1 Introduction

Quantum programs are executed as *circuits*: sequences of elementary gates acting on qubits. For any given high-level operation of interest, there are usually many different circuits that can implement it. A quantum compiler finds a circuit that does so while optimizing for some cost function, typically related to the error rates of the gates used in the circuit.

A general quantum operation involving n qubits is described by a $2^n \times 2^n$ complex matrix, and so finding an optimal circuit for a given operation is intractable in general. As such, practitioners have focussed on important special cases of quantum circuits that admit more concise representations. In this work, we consider the task of synthesizing *Clifford circuits*, a general and important class of quantum circuits which can be represented tractably by $2n \times 2n$ binary matrices known as *stabilizer tableaus* [1]. Figure 1 shows a four-qubit Clifford circuit together with its corresponding tableau representation. The synthesis task is to implement a target Clifford operation as an explicit circuit using as few two-qubit (or *entangling*) gates as possible, since these are typically much more expensive and error-prone than single-qubit gates [4, 29].

Existing Clifford synthesis methods sit at two ends of a tradeoff between runtime and solution quality. At one end, there exist polynomial-time algorithms such as Aaronson–Gottesman [1], which scale efficiently but often produce many more entangling gates than necessary. At the other end, stronger optimization and exact-synthesis methods produce much shorter circuits, but at heavy computational

[1]Correspondence to `richie.yeung@cs.ox.ac.uk`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

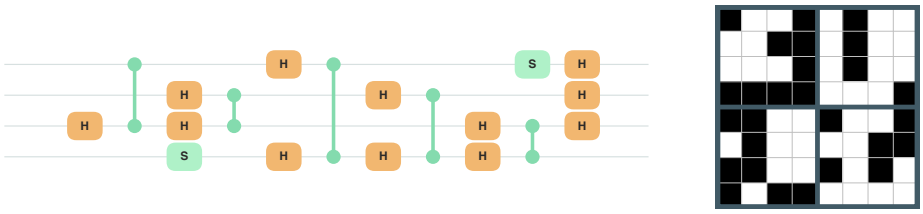


Figure 1: A four-qubit Clifford circuit with 5 CZ gates and its corresponding stabilizer tableau, shown as a $2n \times 2n$ binary matrix with black entries for 1 and white entries for 0. Appendix Fig. A3 shows how successive gate applications locally update this tableau until it reaches the identity.

cost. For example, template-based optimization can take hours per circuit [8], while the optimal six-qubit database of [10] required 2.1 TB of storage and over 300,000 CPU-hours to generate.

In this work, we pursue a middle ground between these two extremes, by learning a reusable neural heuristic for Clifford synthesis that is much lighter than exact or search-heavy optimization, but achieves substantially better entangling-gate counts than polynomial-time baselines. Our approach also generalises to out-of-distribution targets larger and harder than those seen during training.

We approach this task via reinforcement learning, combining curriculum learning together with a novel neural network architecture that respects the symmetries of the problem. For the six-qubit benchmark of Bravyi *et al.* [8], which is currently the largest regime where exact references are available, our policy solves every instance, reaches a maximum gap of one CZ gate across the full suite in 21 seconds, and with extended search matches the exact optimum on $995/1003$ circuits (99.2%) in three hours. In comparison, the prior state-of-the-art method recovers $982/1003$ optima (97.9%) after 217 hours; across the full reported time-limit sweep, the same 21 unrecovered circuits consume 576 hours without reaching optimum [8, 9].

Our neural network architecture is size-agnostic, which allows its reuse across different qubit counts without circuit splicing or network reparameterization. We exploit this to go beyond the exact regime of [8]. In particular, after training on six-qubit instances, we continue training the same policy on ten-qubit instances. The resulting model is able to synthesize Clifford circuits up to 30 qubits while using fewer CZ gates than Qiskit's standard Clifford synthesizers, Aaronson–Gottesman and the Bravyi *et al.* greedy method [1, 24, 23]. To our knowledge, our proposal is the first reinforcement learning method for synthesis of fully connected Clifford circuits, and the first learned policy to produce near-optimal synthesis on six-qubit circuits and moreover transfer to much larger targets.

Our paper aims to be accessible to machine learning researchers with no prior background in quantum computing. In what follows, Section 2 reviews Clifford synthesis from first principles; Section 3 covers related work; Section 4 describes our reinforcement-learning formulation and equivariant architecture; Section 5 covers our experiments, and Section 6 discusses limitations.

2 Background on Clifford Synthesis

In full generality, a quantum computation involving n qubits requires a $2^n \times 2^n$ unitary matrix to represent. This becomes intractably large very quickly as n grows. Accordingly, practitioners have developed more concise representations for important special cases of quantum circuits. A key example is Clifford circuits [21, 1]. By the Gottesman-Knill theorem [21], these correspond to stabilizer tableaus [1], whose essential phase-free content can be described succinctly as a $2n \times 2n$ binary symplectic matrix.[1] A $2n \times 2n$ binary matrix M is *symplectic* if it satisfies

$$M^T \Omega M = \Omega, \qquad \text{where } \Omega = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix}.$$

Here multiplication is meant over the field $\mathbb{F}_2$ (i.e. integers modulo 2), and I_n denotes the $n \times n$ identity matrix. Below we denote the set of symplectic $2n \times 2n$ binary matrices as $\mathrm{Sp}(2n, \mathbb{F}_2)$. At a high level, the (i, j)-component of each of the four $n \times n$ quadrants of M encodes a certain interaction between the i-th and j-th qubits of the corresponding Clifford circuit (see [1] for details).

[1] The precise correspondence is described in Appendix D.

It is standard to show that $\mathrm{Sp}(2n, \mathbb{F}_2)$ forms a group under matrix multiplication. Moreover, its group multiplication structure respects the composition of Clifford circuits. In other words, the Clifford circuit obtained by "plugging in" the output of one Clifford circuit into another corresponds to the symplectic matrix obtained by multiplying the tableaus of those two circuits.

The symplectic group admits a collection of generators $\mathcal{G} \subseteq \mathrm{Sp}(2n, \mathbb{F}_2)$, which we denote below as

$$\mathcal{G} \coloneqq \underbrace{\{H_i : 1 \le i \le n\} \cup \{S_i : 1 \le i \le n\}}_{\text{Single-qubit gates}} \cup \underbrace{\{\mathrm{CZ}_{i,j} : 1 \le i < j \le n\}}_{\text{Two-qubit gates}}. \tag{1}$$

In other words, every Clifford circuit corresponds to a product of these generators, and every product of these generators gives a valid Clifford circuit. Informally, each index i and j refers to a qubit; each H_i is obtained by applying a certain column swap to the identity matrix; and each S_i and $\mathrm{CZ}_{i,j}$ are obtained by applying certain column additions to the identity matrix (see Figure 2 for an illustration). Exact definitions of these generators are given in Appendix D.

In physical terms, the generators H_i, S_i, and $\mathrm{CZ}_{i,j}$ correspond to the native instruction set of a quantum computer. In order to execute a Clifford circuit on hardware, it must therefore be decomposed into a sequence of these generators. By using the symplectic matrix representation of Clifford circuits, the basic task of Clifford synthesis therefore becomes as follows:

$$\text{Given } M_{\text{target}} \in \mathrm{Sp}(2n, \mathbb{F}_2), \text{ find } G_1, \ldots, G_k \in \mathcal{G} \text{ such that } M_{\text{target}} = G_1 \cdots G_k. \tag{2}$$

In general, there are many different sequences $G_1 \cdots G_k$ of different lengths that produce the same overall tableau M_{target}. As such, for practical purposes, it is desirable to solve (2) in a way that is in some sense efficient. In particular, it is especially useful to find circuits with fewer two-qubit gates $\mathrm{CZ}_{i,j}$, as these are typically far more error-prone than the single-qubit gates H_i and S_i. In this way, Clifford synthesis becomes a natural search problem, and therefore amenable to techniques from reinforcement learning, as we explore in this paper.

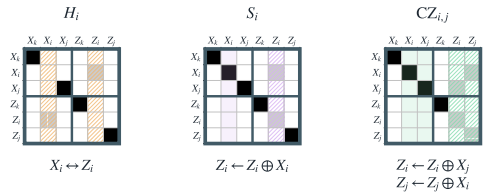


Figure 2: Symplectic generator matrices H_i, S_i, and $\mathrm{CZ}_{i,j}$. Right-multiplying any symplectic matrix by one of these generators applies the highlighted column operations, leaving the other columns untouched.

3 Related Work

Why Clifford synthesis? Clifford operations are a core part of almost every quantum algorithm. A classical result shows that every quantum computation can be expressed up to arbitrary precision using only Clifford gates (H, S and CZ) and T *gates* [6, 37], an additional kind of gate for which two consecutive applications are equivalent to an S gate. In the context of *fault-tolerant quantum computation*, the implementation of the T gate requires a costly procedure known as *magic state distillation* [7, 28]. Recent work on *magic state cultivation* [20] reduces this overhead to roughly match the overhead of a fault-tolerant two-qubit gate such as the CZ. It is therefore practically relevant to reduce CZ gate counts in Clifford circuits as we do here.

Non-neural synthesis. Most prior work on Clifford synthesis has focused on traditional, non-neural methods. These typically involve a sharp trade-off between solution quality and runtime. At one end of the spectrum, a variety of efficient polynomial-time synthesis procedures have been proposed [1, 8], which run quickly but often produce circuits far from the optimum. As an example, Bravyi *et al.* [8] demonstrate that the widely-used Aaronson–Gottesman algorithm produces Clifford circuits with up to 8 times more CZ gates when compared to the output of their peephole optimization procedure, although this procedure can use up to 36 hours per circuit. On the other hand, various search procedures have also been considered, which can achieve near-optimal synthesis but at a much higher computational cost, which limits their applicability to smaller circuits. For example, while SAT-based methods can produce certifiably optimal solutions, they have only been applied successfully on 5 qubit Clifford circuits [33, 38] and with partial success on 6–7 qubit instances [38] using a 3-hour timeout. A* search has also been used to achieve optimal two-qubit counts for random 5 qubit Clifford tableaus but is outperformed by a greedy heuristic beyond 16 qubits [46], both in terms of runtime and two-qubit gate count.

Our work targets the regime between these two extremes: we learn a reusable heuristic that is much lighter than exact or search-heavy optimization, but achieves substantially better entangling-gate counts than polynomial-time baselines.

Neural methods. Reinforcement learning has been applied to various related problems in quantum compilation. AlphaTensor [17] was adapted to T-count optimization, improving state-of-the-art results on quantum arithmetic circuits [35]. Most closely related to our setting, Kremer *et al.* [27] apply reinforcement learning to the problem of *constrained* Clifford synthesis, a setting where entangling gates can only be applied to nearest-neighbor qubits. In contrast, we consider the fully unconstrained setting with all-to-all qubit connectivity. This gives an $O(n^2)$ action space compared to the constrained setting of [27], whose action space is only $O(n)$. The general Clifford synthesis task also admits stronger polynomial-time, heuristic, and exact baselines than the constrained setting, making it a more challenging and informative benchmark for learning-based methods.

Beyond these examples, reinforcement learning has also been applied to various tableau synthesis problems, including CNOT synthesis [42, 27, 12], stabilizer state preparation [47, 14], and Pauli-network synthesis [15], all of which can be viewed as special cases of our setting with additional constraints on the target Clifford tableau. Closely related work also applies learning and reinforcement learning to rewriting graphical representations such as ZX diagrams, which encode Clifford and non-Clifford circuit structure [11, 32, 34, 30]. However, current graph-based ZX simplification techniques [16, 26] mostly rewrite Clifford structure around fixed non-Clifford gates [40, 43]; reducing T count requires separate optimization techniques [3, 22, 13, 35]. Compared to these related tasks, our work is the first to demonstrate robust generalization across unseen circuits of different sizes. In contrast, Kremer *et al.* [27] train 31 models on 31 device architectures of up to 11 qubits for their task. Since our method improves synthesis directly in the unrouted Clifford setting, the same symmetry-aware policy ideas may also be useful for these neighboring stabilizer-structured compilation problems.

4 Method

We approach the problem (2) via reinforcement learning rather than supervised learning. This is due to the combinatorial nature of the problem, which means exact optimal supervision is only available up to the six-qubit regime (i.e. the 1003 exact references of [8, 10]), whereas we would like to scale beyond this to larger circuit sizes.

4.1 Reverse-reduction formulation

We begin by reformulating (2) in a way that is better suited to reinforcement learning. By a standard result, the generators (1) are involutive, so we have $G^{-1} = G$ for each $G \in \mathcal{G}$. This allows us to write (2) equivalently as follows:

$$\text{Given } M_{\text{target}} \in \mathrm{Sp}(2n, \mathbb{F}_2), \text{ find } G_1, \ldots, G_k \in \mathcal{G} \text{ such that } M_{\text{target}} G_1 \cdots G_k = I_{2n}.$$

Given a solution to this latter problem, involutiveness then implies that $M_{\text{target}} = G_k \cdots G_1$ (i.e. the reversed sequence) solves the original problem. Figure 1 gives an example for a four-qubit tableau, and Appendix Fig. A3 shows the full reduction of that tableau to the identity.

This "reverse-reduction" viewpoint is well suited to reinforcement learning because it allows us to define each episode with the same goal state I_{2n}, rather than having this be a general tableau that changes across episodes. As we discuss in Section 4.5, it also provides a natural curriculum, where the initial state of each episode is obtained via a random walk away from the identity, and the walk length controls the overall difficulty of that episode.

4.2 Reinforcement learning setup

We formulate Clifford synthesis as a deterministic Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. To simplify notation, we formulate everything with respect to a fixed qubit count that we denote by n, although we emphasize that the policy we will train is size-agnostic and can be applied across different qubit counts without reparameterization.

We take the state space to be the set of binary symplectic matrices of size $2n \times 2n$, i.e. $\mathcal{S} \coloneqq \mathrm{Sp}(2n, \mathbb{F}_2)$. The action set is then defined as $\mathcal{A} \coloneqq \mathcal{G}$, where $\mathcal{G}$ is the set of generator matrices from (1). There are $\binom{n}{2}$ two-qubit actions and hence $|\mathcal{A}| = \frac{n(n+3)}{2}$ actions in total. The transition function is

$$\mathcal{T}(M, G) \coloneqq MG.$$

so each step applies one Clifford generator to the current tableau by right multiplication. In our implementation, an episode terminates whenever the identity matrix I_{2n} is reached, which corresponds to a successful synthesis, or when a step cap is reached, which corresponds to a failed rollout.

Recall that our goal is to synthesize Clifford circuits with as few gates as possible (and in particular, as few two-qubit gates as possible). Accordingly, we define our reward function as follows:

$$\mathcal{R}(M, G) \coloneqq -\underbrace{0.01\,\mathbb{I}(G = H_i \text{ or } S_i)}_{\text{single-qubit gate penalty}} - \underbrace{\mathbb{I}(G = \mathrm{CZ}_{i,j})}_{\text{two-qubit gate penalty}} + \underbrace{25\,\mathbb{I}(MG = I_{2n})}_{\text{success bonus}} - \underbrace{\frac{\|MG - I_{2n}\|_0}{8n^2}}_{\text{progress shaping}},$$

where $\mathbb{I}$ denotes an indicator function and $\|\cdot\|_0$ counts the number of nonzero entries in a matrix. The single- and two-qubit gate penalty terms of the reward function correspond to the actual cost of the gates on an actual quantum computer. The success bonus term encourages the agent to avoid the undesirable local optimum of only ever applying single-qubit gates. Finally, the last term computes the Hamming distance of the current matrix from the identity, which provides a dense notion of progress throughout an episode. Other reward weights and shapes may be possible here, but we found empirically that this particular choice is robust and works well in practice.

Overall, our goal is to find a policy π to maximise the usual expected sum of discounted rewards over the episode, i.e. $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R_t]$, where here R_t denotes the reward received at time t; the variable T denotes the length of the episode (capped at a maximum number of steps); and $\gamma \in [0, 1)$ is a discount factor. We allow π to be stochastic, so that formally $\pi : \mathcal{S} \to \mathcal{PA}$, where $\mathcal{PA}$ denotes the set of all probability distributions over $\mathcal{A}$. In our experiments, we train π via PPO [36], and at evaluation time either greedily decode a fixed policy or run policy-guided rollouts (see Appendix E).

4.3 Clifford synthesis symmetries

The Clifford synthesis problem admits a natural notion of symmetry. At a high level, this is because the underlying physical qubits are interchangeable, and so relabeling the qubits in a tableau should correspondingly relabel the qubits in the optimal action sequence. In turn, these symmetries also translate into symmetries of the reinforcement learning problem itself, as we explain now.

Denote by $\mathrm{Sym}(n)$ the group of permutations of n elements, i.e. its elements are bijections of the form $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$. This group acts on the state space $\mathcal{S}$ as follows:

$$\sigma \cdot M \coloneqq \Pi_\sigma M \Pi_\sigma^T, \qquad \text{where } \Pi_\sigma \coloneqq \begin{bmatrix} P_\sigma & 0 \\ 0 & P_\sigma \end{bmatrix},$$

and P_σ is the permutation matrix associated with σ. Intuitively, this says that relabeling qubits simultaneously permutes the corresponding rows and columns of the block tableau, while preserving the 2×2 block structure within each qubit. $\mathrm{Sym}(n)$ also acts on the action space $\mathcal{A}$ by

$$\sigma \cdot H_i \coloneqq H_{\sigma(i)}, \qquad \sigma \cdot S_i \coloneqq S_{\sigma(i)}, \qquad \sigma \cdot \mathrm{CZ}_{i,j} \coloneqq \mathrm{CZ}_{\sigma(i),\sigma(j)}.$$

Intuitively, this says that after relabeling the qubits, the same physical gate is represented by the correspondingly relabeled qubit indices. In turn, this leads to an action of $\mathrm{Sym}(n)$ on the space of distributions $\mathcal{PA}$ via the pushforward (i.e. $(\sigma \cdot P)(a) \coloneqq P(\sigma^{-1} \cdot a)$ for $P \in \mathcal{PA}$).

It is now straightforward to show that the transition function $\mathcal{T}$ and reward function $\mathcal{R}$ are respectively equivariant and invariant to these actions, in the sense that

$$\mathcal{T}(\sigma \cdot M, \sigma \cdot G) = \sigma \cdot \mathcal{T}(M, G), \qquad \mathcal{R}(\sigma \cdot M, \sigma \cdot G) = \mathcal{R}(M, G)$$

always holds. By a standard symmetry result for finite MDPs [48], this leads to the following result. (See also Appendix A for a proof in our notation.)

Proposition 4.1. *There exists an optimal policy $\pi^\star : \mathcal{S} \to \mathcal{PA}$ whose corresponding value function $V^\star : \mathcal{S} \to \mathbb{R}$ has the following equivariance and invariance properties respectively:*

$$\pi^\star(\sigma \cdot M) = \sigma \cdot \pi^\star(M), \qquad V^\star(\sigma \cdot M) = V^\star(M).$$

4.4 Network Architecture

We solve our reinforcement learning problem using PPO [36], which requires us to parameterize a neural network $\mathcal{S} \to \mathcal{PA} \times \mathbb{R}$ that maps states to pairs of action distributions and value estimates. In light of Proposition 4.1, we would like these outputs to be respectively equivariant and invariant to the symmetries of the problem. To achieve this, we design a novel neural network architecture that respects these symmetries by construction. At a high level, the architecture we use is as follows:

$$\mathcal{S} \xrightarrow{\text{Embed}} \underbrace{\mathbb{R}^{n\times n\times h}}_{\text{Edge features}} \xrightarrow{\text{Aggregate}} \underbrace{\mathbb{R}^{n\times h}}_{\text{Node features}} \times \underbrace{\mathbb{R}^{n\times n\times h}}_{\substack{\text{Edge features}\\ \text{(unchanged)}}} \xrightarrow[\text{passing}]{\text{Message}} \underbrace{\mathbb{R}^{n\times h}}_{\substack{\text{Node features}\\ \text{(updated)}}} \times \underbrace{\mathbb{R}^{n\times n\times h}}_{\substack{\text{Edge features}\\ \text{(unchanged)}}} \xrightarrow{\text{Readout}} \mathcal{PA}\times\mathbb{R}$$

We explain each of the layers in turn now.

Embedding. Recall that tableaus are $2n \times 2n$ binary matrices. Given an input tableau $M_{\text{in}} \in \mathcal{S}$, we begin by reshaping it to an $n \times n \times 4$ binary tensor M according to its four $n \times n$ quadrants (see Figure 3). From a physical perspective, each entry $M_{ij} \in \{0,1\}^4$ then encodes the directed interactions from the i-th to the j-th qubit in the corresponding Clifford circuit. We then map each M_{ij} to an embedding in $\mathbb{R}^h$ using an embedding dictionary with $2^5 = 32$ entries, where the dictionary key is given by the four bits in M_{ij} together with a binary indicator $\mathbb{I}(i = j)$ for whether the entry encodes a self-interaction. We denote the resulting embeddings by $e_{ij} \in \mathbb{R}^h$.

Aggregation. To initialize our message passing, we compute an aggregate embedding $q_i^{(0)} \in \mathbb{R}^h$ for each qubit by mean and max pooling the embeddings e_{ij} over the rows and columns of the tableau. We give precise details of this pooling in Appendix C. It is also possible to initialise message passing more simply with $q_i^{(0)} = 0$, although we found that aggregation performed better empirically.

Message passing. After aggregation, we perform L rounds of message passing, where in each round we compute ordered-pair messages $m_{i\leftarrow j}^{(k)}$ from qubit j to the updated qubit i as follows:

$$m_{i\leftarrow j}^{(k)} \coloneqq \phi_{\text{msg}}(q_i^{(k)}, q_j^{(k)}, e_{ij}, e_{ji}, \mathbb{I}(i \neq j)) \in \mathbb{R}^h,$$

where ϕ_{msg} is a shared MLP. Here the indicator $\mathbb{I}(i \neq j)$ allows the message function to treat diagonal entries differently from off-diagonal ones, which has physical significance since the former encode self-interactions of qubits rather than interactions between distinct qubits. Given these messages, we then update the qubit embeddings as follows:

$$q_i^{(k+1)} \coloneqq \text{LayerNorm}\left(q_i^{(k)} + \phi_{\text{upd}}\left(q_i^{(k)}, \text{mean}_j\, m_{i\leftarrow j}^{(k)}, \max_j m_{i\leftarrow j}^{(k)}, \eta_{i,1}\right)\right),$$

where $\eta_{i,1}$ denotes additional *rank-based count features* (see Appendix C for a definition). The latter are motivated by work by [46], who show high correlation between these features and the optimal 2-qubit gate count of the related problem of *state preparation*.

Readout. After L rounds of message passing, we get final qubit embeddings $q_i \coloneqq q_i^{(L)}$ for each qubit i. We then form a global summary $g \in \mathbb{R}^{h'}$ by computing permutation invariant summary statistics of these final embeddings and their associated count features (see Appendix C for details). To obtain the final action distribution, we then compute

$$x_i \coloneqq \phi_{\text{H}}(q_i, e_{ii}, g) \in \mathbb{R}, \qquad y_i \coloneqq \phi_{\text{S}}(q_i, e_{ii}, g) \in \mathbb{R},$$
$$z_{ij} \coloneqq \phi_{\text{CZ}}(p_{ij}, g) \in \mathbb{R},$$

where ϕ_{H}, ϕ_{S}, and ϕ_{CZ} are MLPs. Here $p_{ij} = p_{ji}$ collects symmetric combinations of the qubit and edge embeddings, defined explicitly in Appendix C. We then take the collection of every x_i, y_i, and z_{ij} to be the logits for the policy over $\mathcal{A}$. This guarantees that the same logit is produced for $\text{CZ}_{i,j}$ and $\text{CZ}_{j,i}$, as required by the symmetry of these actions. For the value function estimate, a separate MLP acts directly on the permutation-invariant summary statistics defined in Appendix C.

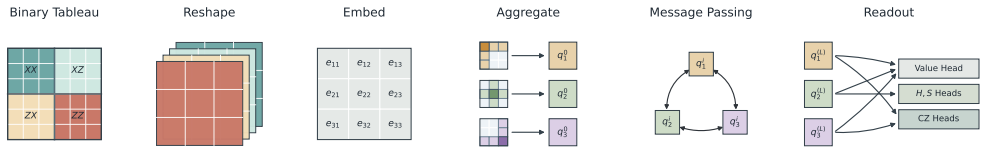


Figure 3: Architecture of the permutation-equivariant policy used for Clifford synthesis. The input tableau is reshaped and embedded into an $n \times n$ grid of qubit-pair edge features, aggregated into n qubit tokens, updated by edge-conditioned message passing on the complete qubit graph, and passed to equivariant policy heads for H, S, and CZ actions, alongside an invariant value head.

Equivariance and invariance. Apart from the value head of the readout layer, each of the individual layers above is equivariant with respect to permutations of the qubits. For example, it is straightforward to check that relabelling the qubits in the input tableau results in a corresponding relabeling of the embeddings e_{ij}, which is then preserved by the aggregation and message-passing layers, and finally results in a corresponding relabeling of the action logits. Since the value head depends only on permutation-invariant summary statistics, it is itself invariant to qubit relabelling.

Size-agnosticism. The same design is also size-agnostic: the only learned weights are in the embedding dictionary and the MLPs, and nothing in the way these are applied depends on the number of qubits n. Accordingly, the same learned weights can be reused across different qubit counts.

4.5 Curriculum

We have not yet defined how the initial tableau M_{target} for each episode is generated. The number of valid tableaus grows as $|\mathrm{Sp}(2n, \mathbb{F}_2)| = 2^{n^2} \prod_{i=1}^{n} (4^i - 1) = 2^{\Theta(n^2)}$ [10]. As such, if we obtained M_{target} by sampling uniformly, we would obtain very difficult targets with high probability, so successful episodes would be rare early in training. We therefore use curriculum learning [5, 31] inspired by reverse curriculum generation for sparse-reward goal-reaching problems [18] and DeepCubeA's reverse-from-goal training for combinatorial puzzles [2]. Kremer *et al.* also use curriculum learning for constrained Clifford synthesis [27].

In our approach, each episode target is generated by a random walk from the identity of some fixed length. The walk length bounds the distance from the goal and provides a way to control target difficulty. Training starts with short walks and advances to longer walks once rollout success at the current level reaches 100%. Precisely, for $n \in \mathbb{N}$ and $d \geq 0$, we define $\mathcal{P}_{n,d}$ to be the distribution over $\mathrm{Sp}(2n, \mathbb{F}_2)$ generated by a random walk from I_{2n} of length

$$L = \lfloor d \rfloor + B, \qquad \text{where } B \sim \mathrm{Bernoulli}(d - \lfloor d \rfloor).$$

This means that $\mathbb{E}[L] = d$, and moreover when d is an integer, it reduces to a d-step random walk. This interpolation gives more granular control over the curriculum, especially for small d, and shifts the distribution smoothly. For $n \geq 3$, this random walk converges to uniform sampling from $\mathrm{Sp}(2n, \mathbb{F}_2)$ as $d \to \infty$ (see Theorem B.1 in Appendix B for a proof).

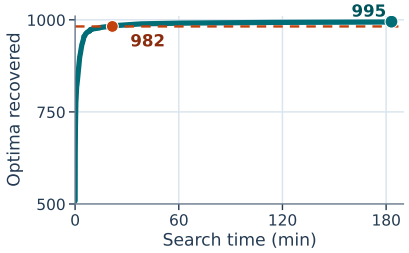
In addition to d, our size-agnostic agent also allows us to build a curriculum across different qubit counts n. In our experiments, we exploit this by training initially on six-qubit circuits until convergence, before continuing to train on ten-qubit circuits.

5 Experimental Results

Using the setup described above, we optimized the parameters of our model via PPO [36] using PufferLib [41]. Full training hyperparameters are listed in Appendix Table A4. At evaluation time, we then use our policy as a heuristic for Clifford synthesis, either by greedily following the action with highest probability at each step, or by sampling actions according to the policy until a solution is found or a step cap is reached (see Appendix E for details).

5.1 Recovering Optimal Six-Qubit Clifford Circuits

We applied our approach to the benchmark of 1003 optimal six-qubit tableaus considered by Bravyi *et al.* [8]. (These reference tableaus were themselves taken from the exhaustive database of optimal six-qubit Clifford circuits generated by Bravyi–Latone–Maslov [10].) However, we emphasize that our agent is trained according to the methodology described in Section 4, without any access to those 1003 tableaus or their optimal solutions. Even the easiest benchmark targets are extremely unlikely to be seen during training, since there are about 1.3×10^{14} optimal six-qubit Cliffords with five CZ gates [10]. As a baseline, we compared against the symbolic peephole optimizer of [8], which matches the optimal entangling-gate count for 982/1003 circuits after 217 hours, with the 21 remaining circuits consuming 576 hours without reaching the optimum [9].



On this task, our approach solves each of the 1003 instances and matches the optimum on 995/1003 circuits (99.2%). The remaining eight circuits are each off by one CZ gate. Greedily following the trained policy takes 0.83 seconds for the whole suite and already recovers 507/1003 optimal solutions. A brief policy-guided search pass brings all the circuits to within one CZ gate from optimal in 21 seconds total, with 610/1003 of those being optimal. The longer policy-guided search reaches the previous state-of-the-art $982/1003$ in 22 minutes, reaches $990/1003$ after 52 minutes, and reaches $995/1003$ after 183 minutes, outperforming the previous state-of-the-art of Bravyi et al. [8] by a significant margin.

5.2 Generalizing Beyond the Training Size

We also evaluated our approach on larger circuits beyond six qubits. Since there are no optimal references available in this context, we compare our agent's performance against existing synthesis algorithms that run in polynomial time. Specifically, we compare against the two main polynomial-time Clifford synthesizers from Qiskit [1, 24, 23], a quantum computing library with over a million monthly downloads at the time of writing. We do not compare against the neural methods discussed in Section 3, as they do not address our specific task, and moreover are not size-agnostic and hence would require training separate models for each qubit count in our sweep.

To test size generalization and the cross-size curriculum, we benchmarked two checkpoints: one trained only on six-qubit circuits, and one for which we subsequently continued training on ten-qubit circuits also. For each qubit size n and number of initial gates d, we generated 100 held-out targets by applying d uniformly sampled Clifford gates from $\mathcal{G}$ to the identity tableau. We swept across $n = 7, \ldots, 30$ and $d \in \{16, 32, 64, 128, 256, 512, 1024, \infty\}$, using the same fixed target tableaus for the learned agent and both Qiskit baselines. The ∞ circuits are actually sampled uniformly [25] from $\mathrm{Sp}(2n, \mathbb{F}_2)$ as that is the limit distribution of the random walk (see Theorem B.1).

The results of this experiment are given in Figure 4 (see also Appendix A4 and A7 for the full sweeps). The model trained on ten qubits produces much shorter circuits when it succeeds. Across the finite-difficulty settings, the model returns lower average CZ counts than the Bravyi *et al.* greedy and Aaronson–Gottesman algorithms. At 30 qubits and 1024 initial Clifford gates, where the learned synthesizer still solves all targets, it uses 323.3 CZ gates on average, 124.2 fewer than Qiskit's Bravyi *et al.* greedy synthesizer and 460.1 fewer than Aaronson–Gottesman.

At the ∞ endpoint, the ten-qubit checkpoint starts to lose reliability beyond about 24 qubits: solve rate falls from 99% at 24 qubits to 59% at 30 qubits. The learned model still beats both Qiskit baselines in CZ count after restricting all methods to its solved targets. The appendix gives the full sweeps (Figures A5 and A6).

Interestingly, these scale-transfer experiments expose a reliability–quality tradeoff between the two checkpoints. The six-qubit trained model is more reliable: it solves every larger-circuit target in our sweep, including the ∞ endpoint. Its circuits are worse, however, especially on tableaus corresponding to larger and deeper circuits. See Appendix D.1 for additional discussion.

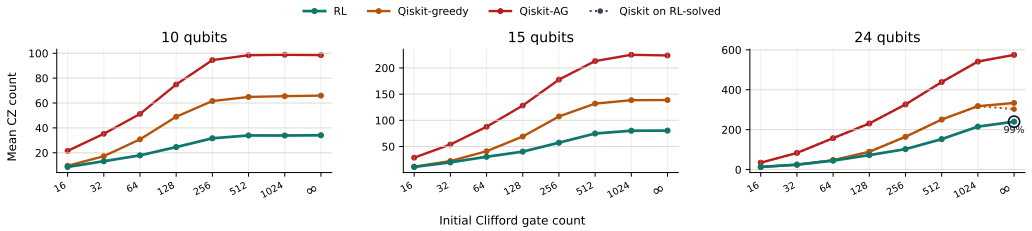


Figure 4: CZ-count comparison for the ten-qubit-trained model at 10, 15, and 24 evaluation qubits. Each panel fixes the qubit count and sweeps target difficulty from 16–1024 initial Clifford gates through the ∞ endpoint defined in Section 5.2. Solid Qiskit curves show all-target means; dotted same-color markers and shaded gaps show the corresponding Qiskit means restricted to targets solved by the agent when these differ. The percentage annotation gives the agent solve rate at the ∞ endpoint. Appendix Figure A4 shows the full evaluation on 7–30 qubits.

Family	CZ Count
Ours	$\mathbf{13.49 \pm 0.01}$
RelTransformer [39]	$\mathbf{13.47 \pm 0.06}$
Transformer [45]	14.12 ± 0.28
MLP	15.88 ± 0.05
FlatMLP	15.19 ± 0.16

Table 1: Architecture ablation on 100 uniformly sampled six-qubit Clifford targets. Entries are mean CZ counts over solved targets under rollout decoding, reported as mean $\pm$ standard deviation over three seeds. All models solve every target. The full sweep is in Appendix Table A2.

5.3 Architecture Ablation

We ablate the policy representation by progressively removing qubit-level structure. Ours uses the message-passing architecture from Section 4; RelTransformer replaces message passing with relation-aware attention [39]; Transformer uses ordinary self-attention [45]; MLP keeps equivariant per-qubit processing but removes communication between qubits; and FlatMLP uses an unconstrained MLP over the flattened tableau.

Table 1 shows model performance on uniformly random Cliffords. Since all models solve all targets, the comparison is entirely in circuit quality. The communication-based equivariant models perform best: Ours and RelTransformer are essentially tied, ordinary attention is weaker, removing communication is worse, and the flat unstructured baseline is also substantially behind. The best communication-based models use more than two fewer CZ gates than MLP and about two fewer than FlatMLP. This suggests that the important ingredient is not the exact message-passing update, but the combination of tableau-block features, qubit equivariance, and information flow between qubit tokens.

6 Discussion, Limitations, and Future Work

We presented a neural synthesis algorithm for Clifford circuits based on reinforcement learning over stabilizer-tableau reduction. The central architectural idea is to build permutation symmetry directly into the policy so that the model can scale across qubit sizes. We have shown that this setup is robust, giving optimal circuits at six qubits and improving over existing polynomial-time algorithms for larger qubit systems.

The symmetries and two-dimensional learning curricula we study here exist in related problems, and it is an open question whether these techniques developed in this work would yield significant improvements in those areas as well. Future work can also explore the optimal way to perform this two-dimensional curriculum learning.

The learned policy is small and relatively fast, and can be used on consumer devices without a GPU. In the future, this can be packaged into a lightweight library for neural Clifford synthesis.

A limitation of our current formulation is that one-step action selection becomes expensive for large n: the number of possible $\mathrm{CZ}_{i,j}$ actions grows as $O(n^2)$, and the resulting rollouts can require $O(n^2)$ gates. This makes naive inference scale poorly even when the network itself is size-agnostic. Future work could reduce this cost with a factorized action space, hierarchical actions that choose structured multi-gate moves, or policy-guided planning methods such as Monte Carlo tree search during training and inference.

See `y-richie-y.github.io/clifford/` for a live browser demo and recovered Bravyi *et al.* benchmark circuits, including new circuits for previously unrecovered optima.

Acknowledgements and Disclosures

We thank Christopher Mingard, Jasmine Brewer, and Ciarán Ryan-Anderson for useful feedback on the manuscript. We also thank the anonymous NeurIPS reviewers for their helpful comments and suggestions. We thank Alexander Koziell-Pipe for interesting discussions and for exploring Monte Carlo tree search for circuit synthesis. RY thanks Dr Simon Harrison for generous support through the Wolfson Harrison UKRI Quantum Foundation Scholarship. This work is supported by the Engineering and Physical Sciences Research Council grant number EP/Z002230/1, *(De)constructing quantum software (DeQS)*. The research of RC is supported by the National Research Foundation, Singapore, under its National Research Foundation Fellowship in Artificial Intelligence (Award No.: NRFFIAI1-2024-0014).

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5), 2004. doi: 10.1103/PhysRevA.70.052328. URL `https://doi.org/10.1103/physreva.70.052328`.

[2] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1 (8):356–363, 2019. doi: 10.1038/s42256-019-0070-z. URL `https://doi.org/10.1038/s42256-019-0070-z`.

[3] Matthew Amy and Michele Mosca. T-count optimization and reed–muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, 2019. doi: 10.1109/TIT.2019.2906374. URL `https://doi.org/10.1109/tit.2019.2906374`.

[4] Unai Aseguinolaza, Nahual Sobrino, Gabriel Sobrino, Joaquim Jornet-Somoza, and Juan Borge. Error estimation in current noisy quantum computers. *Quantum Information Processing*, 23(5), 2024. doi: 10.1007/s11128-024-04384-z. URL `https://doi.org/10.1007/s11128-024-04384-z`.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009. doi: 10.1145/1553374.1553380. URL `https://doi.org/10.1145/1553374.1553380`.

[6] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. A new universal and fault-tolerant quantum basis. *Information Processing Letters*, 75(3):101–107, 2000. doi: 10.1016/S0020-0190(00)00084-3. URL `https://doi.org/10.1016/s0020-0190(00)00084-3`.

[7] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, 71(2), 2005. doi: 10.1103/PhysRevA.71.022316. URL `https://doi.org/10.1103/physreva.71.022316`.

[8] Sergey Bravyi, Ruslan Shaydulin, Shaohan Hu, and Dmitri Maslov. Clifford circuit optimization with templates and symbolic Pauli gates. *Quantum*, 5:580, 2021. doi: 10.22331/q-2021-11-16-580. URL `https://doi.org/10.22331/q-2021-11-16-580`.

[9] Sergey Bravyi, Ruslan Shaydulin, Shaohan Hu, and Dmitri Maslov. Data to accompany Clifford Circuit Optimization with Templates and Symbolic Pauli Gates, 2021. URL https://github.com/rsln-s/Clifford_Circuit_Optimization_with_Templates_and_Symbolic_Pauli_Gates. GitHub repository, accessed 2026-05-03.

[10] Sergey Bravyi, Joseph A. Latone, and Dmitri Maslov. 6-qubit optimal Clifford circuits. *npj Quantum Information*, 8(1), 2022. doi: 10.1038/s41534-022-00583-7. URL https://doi.org/10.1038/s41534-022-00583-7.

[11] Francois Charton, Alexandre Krajenbrink, Konstantinos Meichanetzidis, and Richie Yeung. Teaching small transformers to rewrite ZX diagrams. In *3rd MATH-AI Workshop at NeurIPS'23*, 2023. URL https://mathai2023.github.io/papers/34.pdf.

[12] Jacopo Cossio, Daniele Lizzio Bosco, Riccardo Romanello, Giuseppe Serra, and Carla Piazza. AlphaCNOT: Learning CNOT minimization with model-based planning, 2026. URL https://arxiv.org/abs/2604.13812.

[13] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and effective techniques for T-Count reduction via spider nest identities. In *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158, pages 11:1–11:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.TQC.2020.11. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2020.11.

[14] Michael Doherty, Matteo Puviani, Jasmine Brewer, Gabriel Matos, David Amaro, Ben Criger, and David T. Stephen. Fast stabilizer state preparation via AI-optimized graph decimation, 2026. URL https://arxiv.org/abs/2603.17743v1.

[15] Ayushi Dubal, David Kremer, Simon Martiel, Victor Villar, Derek Wang, and Juan Cruz-Benito. Pauli network circuit synthesis with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.14448v1.

[16] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum*, 4:279, 2020. doi: 10.22331/q-2020-06-04-279. URL https://doi.org/10.22331/q-2020-06-04-279.

[17] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://doi.org/10.1038/s41586-022-05172-4.

[18] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495, 2017. URL http://proceedings.mlr.press/v78/florensa17a/florensa17a.pdf.

[19] Craig Gidney. Inverting clifford tableaus, 2020. URL https://algassert.com/post/2002. Blog post, accessed 2026-04-07.

[20] Craig Gidney, Noah Shutty, and Cody Jones. Magic state cultivation: growing t states as cheap as CNOT gates, 2024. URL https://arxiv.org/abs/2409.17595v1.

[21] Daniel Gottesman. The heisenberg representation of quantum computers, 1998. URL https://arxiv.org/abs/quant-ph/9807006.

[22] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces T count. *Quantum Science and Technology*, 4(1):015004, 2018. doi: 10.1088/2058-9565/aad604. URL https://doi.org/10.1088/2058-9565/aad604.

[23] IBM Quantum and Qiskit contributors. GreedySynthesisClifford, 2026. URL https://quantum.cloud.ibm.com/docs/en/api/qiskit/2.2/qiskit.transpiler.passes.synthesis.hls_plugins.GreedySynthesisClifford. Qiskit 2.2 API documentation, accessed 2026-05-07.

[24] IBM Quantum and Qiskit contributors. DefaultSynthesisClifford, 2026. URL https://quantum.cloud.ibm.com/docs/en/api/qiskit/2.2/qiskit.transpiler.passes.synthesis.hls_plugins.DefaultSynthesisClifford. Qiskit 2.2 API documentation, accessed 2026-05-07.

[25] IBM Quantum and Qiskit contributors. random_clifford, 2026. URL https://quantum.cloud.ibm.com/docs/en/api/qiskit/2.2/quantum_info. Qiskit 2.2 API documentation, accessed 2026-05-07.

[26] Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Physical Review A*, 102(2), 2020. doi: 10.1103/PhysRevA.102.022406. URL https://doi.org/10.1103/physreva.102.022406.

[27] 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. URL https://arxiv.org/abs/2405.13196v2.

[28] Daniel Litinski. Magic state distillation: Not as costly as you think. *Quantum*, 3:205, 2019. doi: 10.22331/q-2019-12-02-205. URL https://doi.org/10.22331/q-2019-12-02-205.

[29] Daniel Litinski and Felix von Oppen. Lattice surgery with a twist: Simplifying Clifford gates of surface codes. *Quantum*, 2:62, 2018. doi: 10.22331/q-2018-05-04-62. URL https://doi.org/10.22331/q-2018-05-04-62.

[30] Alexander Mattick, Maniraman Periyasamy, Christian Ufrecht, Abhishek Y. Dubey, Christopher Mutschler, Axel Plinge, and Daniel D. Scherer. Optimizing quantum circuits via ZX diagrams using reinforcement learning and graph neural networks, 2025. URL https://arxiv.org/abs/2504.03429v1.

[31] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. URL http://jmlr.org/papers/volume21/20-212/20-212.pdf.

[32] Maximilian Nägele and Florian Marquardt. Optimizing ZX-diagrams with deep reinforcement learning. *Machine Learning: Science and Technology*, 5(3):035077, 2024. doi: 10.1088/2632-2153/ad76f7. URL https://doi.org/10.1088/2632-2153/ad76f7.

[33] Tom Peham, Nina Brandl, Richard Kueng, Robert Wille, and Lukas Burgholzer. Depth-optimal synthesis of Clifford circuits with SAT solvers. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 802–813, 2023. doi: 10.1109/QCE57702.2023.00095. URL https://doi.org/10.1109/qce57702.2023.00095.

[34] Jordi Riu, Jan Nogué, Gerard Vilaplana, Artur Garcia-Saez, and Marta P. Estarellas. Reinforcement learning based quantum circuit optimization via ZX-calculus. *Quantum*, 9:1758, 2025. doi: 10.22331/Q-2025-05-28-1758. URL https://doi.org/10.22331/q-2025-05-28-1758.

[35] Francisco J. R. Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco J. H. Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, Alhussein Fawzi, Konstantinos Meichanetzidis, and Pushmeet Kohli. Quantum circuit optimization with AlphaTensor. *Nature Machine Intelligence*, 7(3):374–385, 2025. doi: 10.1038/s42256-025-01001-1. URL https://doi.org/10.1038/s42256-025-01001-1.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347v2.

[37] Peter Selinger. Efficient Clifford+T approximation of single-qubit operators. *Quantum Information and Computation*, 15(1&2):159–180, 2015. doi: 10.26421/qic15.1-2-10. URL https://doi.org/10.26421/qic15.1-2-10.

[38] Irfansha Shaik and Jaco van de Pol. CNOT-optimal Clifford synthesis as SAT. In *28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025)*, volume 341, pages 28:1–28:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi: 10.4230/LIPIcs.SAT.2025.28. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2025.28.

[39] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074/.

[40] Will Simmons. Relating measurement patterns to circuits via Pauli flow. *Electronic Proceedings in Theoretical Computer Science*, 343:50–101, 2021. doi: 10.4204/EPTCS.343.4. URL https://doi.org/10.4204/eptcs.343.4.

[41] Joseph Suarez. PufferLib: Making reinforcement learning libraries and environments play nice, 2024. URL https://arxiv.org/abs/2406.12905v1.

[42] Arianne van de Griend. Constrained quantum CNOT circuit re-synthesis using deep reinforcement learning. Master's thesis, Radboud University, 2019. URL https://theses.ubn.ru.nl/handle/123456789/10713.

[43] John van de Wetering, Richie Yeung, Tuomas Laakkonen, and Aleks Kissinger. Optimal compilation of parametrised quantum circuits. *Quantum*, 9:1828, 2025. doi: 10.22331/q-2025-08-27-1828. URL https://doi.org/10.22331/q-2025-08-27-1828.

[44] Elise van der Pol, Daniel E. Worrall, Herke van Hoof, Frans A. Oliehoek, and Max Welling. MDP homomorphic networks: Group symmetries in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 4199–4210, 2020. URL https://proceedings.neurips.cc/paper/2020/file/2be5f9c2e3620eb73c2972d7552b6cb5-Paper.pdf.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[46] Mark Webster, Stergios Koutsioumpas, and Dan E Browne. Heuristic and optimal synthesis of CNOT and Clifford circuits. *Quantum*, 10:2212, 2026. doi: 10.22331/q-2026-09-21-2212. URL https://doi.org/10.22331/q-2026-09-21-2212.

[47] 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. *Physical Review X*, 15(4):041012, 2025. doi: 10.1103/gqpr-dgz7. URL https://doi.org/10.1103/gqpr-dgz7.

[48] Martin Zinkevich and Tucker R. Balch. Symmetry in Markov decision processes and its implications for single agent and multiagent learning. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 632–640. Morgan Kaufmann, 2001. URL https://www.cs.cmu.edu/~maz/publications/symmetry7.pdf.

A Equivariant Optimal Policies

This appendix explains why we can use an equivariant policy class in the Clifford synthesis MDP, following the standard symmetry theory of finite MDPs [44, 48]. The argument applies more generally to stationary discounted MDPs whose dynamics and rewards commute with a group action.

Proposition. Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ be a finite deterministic MDP with stationary transition and reward functions and discount factor $\gamma \in [0, 1)$. Let a group Γ act on both $\mathcal{S}$ and $\mathcal{A}$. Suppose that, for every $\sigma \in \Gamma$, $M \in \mathcal{S}$, and $G \in \mathcal{A}$,

$$\mathcal{T}(\sigma \cdot M, \sigma \cdot G) = \sigma \cdot \mathcal{T}(M, G), \qquad \mathcal{R}(\sigma \cdot M, \sigma \cdot G) = \mathcal{R}(M, G).$$

Then there exists an optimal stationary stochastic Markov policy $\pi^\star$ that is equivariant:

$$\pi^\star(\sigma \cdot G \mid \sigma \cdot M) = \pi^\star(G \mid M)$$

for all states M, actions G, and group elements σ.

Proof. Let $V^\star$ be the unique solution to the discounted Bellman optimality equation

$$V^\star(M) = \max_{G \in \mathcal{A}} \left[\mathcal{R}(M, G) + \gamma V^\star(\mathcal{T}(M, G))\right].$$

For any fixed $\sigma \in \Gamma$, define $W_\sigma(M) = V^\star(\sigma \cdot M)$. Then

$$\begin{aligned} W_\sigma(M) &= V^\star(\sigma \cdot M) \\ &= \max_{G' \in \mathcal{A}} \left[\mathcal{R}(\sigma \cdot M, G') + \gamma V^\star(\mathcal{T}(\sigma \cdot M, G'))\right] \\ &= \max_{G \in \mathcal{A}} \left[\mathcal{R}(\sigma \cdot M, \sigma \cdot G) + \gamma V^\star(\mathcal{T}(\sigma \cdot M, \sigma \cdot G))\right] \\ &= \max_{G \in \mathcal{A}} \left[\mathcal{R}(M, G) + \gamma V^\star(\sigma \cdot \mathcal{T}(M, G))\right] \\ &= \max_{G \in \mathcal{A}} \left[\mathcal{R}(M, G) + \gamma W_\sigma(\mathcal{T}(M, G))\right]. \end{aligned}$$

Here we reindexed the maximisation using the bijection $G \mapsto \sigma \cdot G$. Thus W_σ also satisfies the Bellman optimality equation. By uniqueness, $W_\sigma = V^\star$, so

$$V^\star(\sigma \cdot M) = V^\star(M).$$

Define the optimal action-value function

$$Q^\star(M, G) = \mathcal{R}(M, G) + \gamma V^\star(\mathcal{T}(M, G)).$$

The invariance of $V^\star$, together with the equivariance of $\mathcal{T}$ and invariance of $\mathcal{R}$, gives

$$Q^\star(\sigma \cdot M, \sigma \cdot G) = Q^\star(M, G).$$

Therefore the optimal action set

$$\mathcal{A}^\star(M) = \arg\max_{G \in \mathcal{A}} Q^\star(M, G)$$

satisfies

$$\mathcal{A}^\star(\sigma \cdot M) = \sigma \cdot \mathcal{A}^\star(M).$$

Let $\pi^\star(\cdot \mid M)$ be the uniform distribution over $\mathcal{A}^\star(M)$. This policy is optimal because it assigns probability only to optimal actions. It is also equivariant, since the optimal action set at $\sigma \cdot M$ is exactly the relabeling of the optimal action set at M. Hence

$$\pi^\star(\sigma \cdot G \mid \sigma \cdot M) = \pi^\star(G \mid M).$$

□

Application to Clifford synthesis. In our Clifford synthesis MDP, the symmetry group is the qubit permutation group S_n. A permutation $\sigma \in S_n$ acts on tableaus by simultaneously relabeling the row and column qubit blocks, and acts on generators by

$$H_i \mapsto H_{\sigma(i)}, \qquad S_i \mapsto S_{\sigma(i)}, \qquad \mathrm{CZ}_{i,j} \mapsto \mathrm{CZ}_{\sigma(i),\sigma(j)}.$$

Right multiplication by generators commutes with this relabeling, so

$$\mathcal{T}(\sigma \cdot M, \sigma \cdot G) = \sigma \cdot \mathcal{T}(M, G).$$

The reward is also invariant: the single-qubit and two-qubit costs depend only on the gate type, the identity tableau is fixed by every qubit relabeling, and the Hamming distance to the identity is unchanged by simultaneously permuting rows and columns. Therefore the proposition applies, and there exists an optimal stationary stochastic Markov policy for the discounted Clifford synthesis objective that is equivariant under qubit relabeling. If a hard rollout cap is included as part of the objective, we can augment the state with the remaining step budget and let S_n act trivially on that extra coordinate. The same argument then gives an optimal equivariant policy on the augmented state space.

B Random-Walk Limit Distribution

This appendix justifies the use of uniformly sampled Clifford targets as the ∞ endpoint of the random-walk curriculum.

Theorem B.1. *Fix $n \geq 3$, write $\Gamma = \mathrm{Sp}(2n, \mathbb{F}_2)$, and let K be the transition matrix for the random walk on Γ that right-multiplies by a uniformly sampled generator from $\mathcal{G}$ in* (1)*. Then the d-step distribution from the identity converges to the uniform distribution on Γ:*

$$K^d(I_{2n}, M) \longrightarrow \frac{1}{|\Gamma|} \qquad \textit{for every } M \in \Gamma.$$

For $n = 1$ and $n = 2$, the corresponding walk has period 2 and therefore does not converge to the uniform distribution at fixed walk length.

Proof. We use the standard ergodic theorem for finite Markov chains: an irreducible and aperiodic finite chain converges to its unique stationary distribution. Because $\mathcal{G}$ generates Γ, the associated Cayley graph is connected, so the chain is irreducible. Figure A1 shows a local ball in the corresponding generator graph for the two-qubit case, where random-walk steps move along generator-labeled edges from the identity. The uniform distribution is stationary: if M is uniform on Γ and $G \in \mathcal{G}$ is fixed, then MG is still uniform, because right multiplication by G only permutes the elements of Γ.

For convergence to this stationary distribution we also need aperiodicity. Each generator is an involution in the binary symplectic representation, so $G^2 = I_{2n}$ gives a closed walk of length 2. For $n \geq 3$, there is also an odd closed walk:

$$H_1\,\mathrm{CZ}_{1,2}\,H_1\,\mathrm{CZ}_{1,3}\,H_1\,\mathrm{CZ}_{2,3}\,\mathrm{CZ}_{1,2}\,H_1\,\mathrm{CZ}_{1,3} = I_{2n}.$$

Figure A2 draws this nine-step identity circuit. Equivalently, the identity is verified by multiplying the generator matrices defined in Appendix D; it acts nontrivially only on qubits $1, 2, 3$, so the same relation embeds in every larger n. Thus the period of the identity divides both 2 and 9, and is therefore 1. Irreducibility implies every state has the same period, so the whole chain is finite, irreducible, and aperiodic. The standard convergence theorem for finite Markov chains then gives

$$K^d(I_{2n}, M) \longrightarrow \frac{1}{|\Gamma|} \qquad \text{for every } M \in \Gamma,$$

which is exactly uniform sampling from $\mathrm{Sp}(2n, \mathbb{F}_2)$.

For $n = 1$ and $n = 2$, the walk generated by $\{H_i, S_i, \mathrm{CZ}_{i,j}\}$ has period 2, so the fixed-length distributions oscillate between the two parity classes of the Cayley graph rather than converging. Concretely, for $n = 1$ the generators are only H_1 and S_1, and the two parity classes are

$$\{I, H_1S_1, S_1H_1\} \qquad \text{and} \qquad \{H_1, S_1, H_1S_1H_1\},$$

the elements reachable at even and odd times, respectively. For $n = 2$, $Sp(4, \mathbb{F}_2)$ has 720 elements, split into two parity classes of size 360 under this generator walk. □

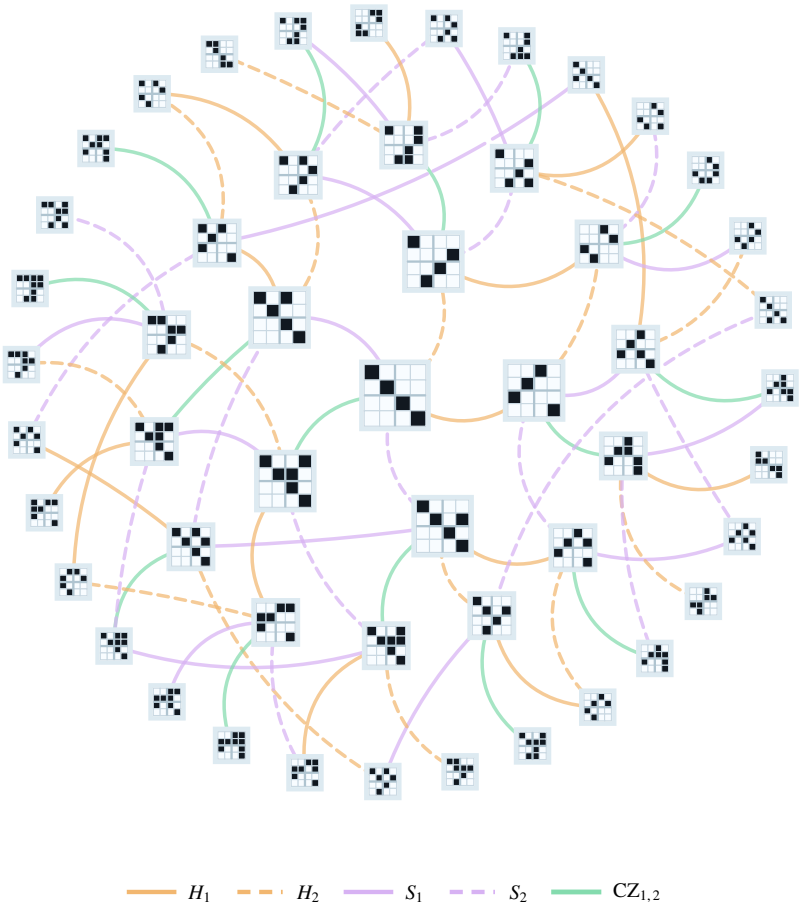


Figure A1: Local view of the Cayley graph induced by the two-qubit Clifford generators. Vertices are tableaus reached from the identity by walks of length at most three; the legend identifies the generator for each edge.

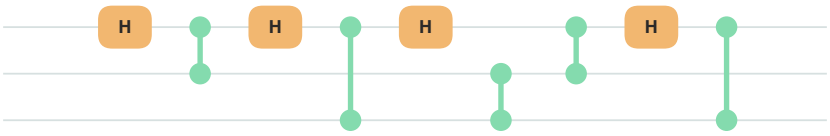


Figure A2: Nine-gate identity circuit used in Theorem B.1.

C Architecture Implementation Details

This appendix gives the implementation details behind the embedding, aggregation, message-passing, and readout stages described in Section 4. All additional features below are permutation-equivariant count-based features or permutation-invariant global summaries, so including them does not change the symmetry argument in the main text.

Embedding. For each 2×2 block M_{ij} of the reshaped tableau, the embedding lookup uses the four binary entries of M_{ij} together with a diagonal indicator $\mathbb{I}(i = j)$, giving 32 possible block states. This produces the edge embedding $e_{ij} \in \mathbb{R}^h$ used in the main text.

Aggregation and count features. The initial qubit token $q_i^{(0)}$ is built from row-pooled and column-pooled edge embeddings, the diagonal edge embedding e_{ii}, and a count-feature vector η_i. Specifically, with a shared MLP ϕ_{node}, we set

$$q_i^{(0)} \coloneqq \phi_{\text{node}}(\underbrace{[\text{mean}_j\, e_{ij}, \max_j e_{ij}]}_{\text{Row features}}, \underbrace{[\text{mean}_j\, e_{ji}, \max_j e_{ji}]}_{\text{Column features}}, \underbrace{e_{ii}}_{\text{Diagonal embedding}}, \eta_i). \tag{3}$$

Here $\eta_i \coloneqq [\eta_{i,1}, \eta_{i,2}] \in \mathbb{R}^{11}$. The choice of these features is motivated by Webster *et al.* [46], who show a strong correlation between related rank-count features and the optimal two-qubit gate count for stabilizer state preparation. The first component $\eta_{i,1} \in \mathbb{R}^9$ is

$$\eta_{i,1} = \big[d_i,\ \rho_i^{\text{row}}, \rho_i^{\text{col}},\ \rho_i^{\text{row},1}, \rho_i^{\text{row},2},\ \rho_i^{\text{col},1}, \rho_i^{\text{col},2},\ \rho_i^{\text{offrow}}, \rho_i^{\text{offcol}}\big],$$

where d_i indicates whether the diagonal block M_{ii} is the 2×2 identity; ρ_i^{row} and ρ_i^{col} are the fractions of nonzero blocks in block row i and block column i; $\rho_i^{\text{row},1}, \rho_i^{\text{row},2}$ and $\rho_i^{\text{col},1}, \rho_i^{\text{col},2}$ are the corresponding fractions of rank-one and rank-two blocks; and $\rho_i^{\text{offrow}}, \rho_i^{\text{offcol}}$ are the fractions of nonzero off-diagonal blocks in row i and column i, normalized by $n-1$. The vector $\eta_{i,2} \in \mathbb{R}^2$ records the rank of the diagonal block M_{ii} as two indicators for rank one and rank two.

Message passing. The message function ϕ_{msg} and update function ϕ_{upd} are shared across all qubit indices and all message-passing rounds. As in the main text, the update for qubit i uses the current token $q_i^{(k)}$, mean-pooled incoming messages, max-pooled incoming messages, and the rank-count component $\eta_{i,1}$.

Readout. Let $q_i = q_i^{(L)}$ be the final qubit token after L message-passing rounds. The implementation first forms invariant statistics

$$s \coloneqq \big[\text{mean}_i\, q_i,\ \max_i q_i,\ \text{std}_i\, q_i,\ \bar{c}\big],$$
$$\bar{c} \coloneqq \big[\text{mean}_i\, d_i,\ \text{mean}_i\, \rho_i^{\text{row}},\ \text{mean}_i\, \rho_i^{\text{col}},\ \text{mean}_i\, \rho_i^{\text{offrow}},\ \text{mean}_i\, \rho_i^{\text{offcol}},\ \text{mean}_i\, \eta_{i,2}\big],$$

where std_i is the elementwise standard deviation over qubit tokens. Thus the invariant readout statistics have dimension $3h + 7$. The implementation maps these statistics through a global MLP to obtain the learned summary g.

For the one-qubit action heads, the readout can be written as the separate MLPs ϕ_{H} and ϕ_{S} used in the main text. Concretely, the implementation uses a local MLP conditioned on a one-hot vector encoding the action type:

$$(q_i,\ e_{ii},\ g,\ \eta_i,\ \tau_a),$$

where $\tau_a \in \mathbb{R}^2$ encodes the action type. The two-qubit CZ head receives (p_{ij}, g), where the symmetric pair features are

$$p_{ij} \coloneqq [q_i + q_j,\ q_i \odot q_j,\ |q_i - q_j|,\ e_{ij} + e_{ji},\ e_{ij} \odot e_{ji}].$$

Here $\odot$ denotes elementwise multiplication. The value head is the MLP $\phi_V(s)$ applied directly to the invariant statistics s.

D Clifford Circuits and Tableaus

This appendix briefly reviews the *circuit model* and the *stabilizer formalism* underlying the synthesis problem studied in the main text. In the standard circuit model, a quantum algorithm is expressed as a sequence of elementary gates acting on qubits. A *universal gate set* such as Clifford+T can approximate arbitrary quantum computations to arbitrary precision. In this paper we focus on the Clifford subset, generated by H, S, and CZ, because it admits a compact exact representation unavailable for general circuits. For a more complete treatment of Clifford circuits and stabilizer tableaus, see Aaronson and Gottesman [1].

The *Pauli observables* correspond to the canonical measurement bases of a qubit. Their matrix representatives are

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

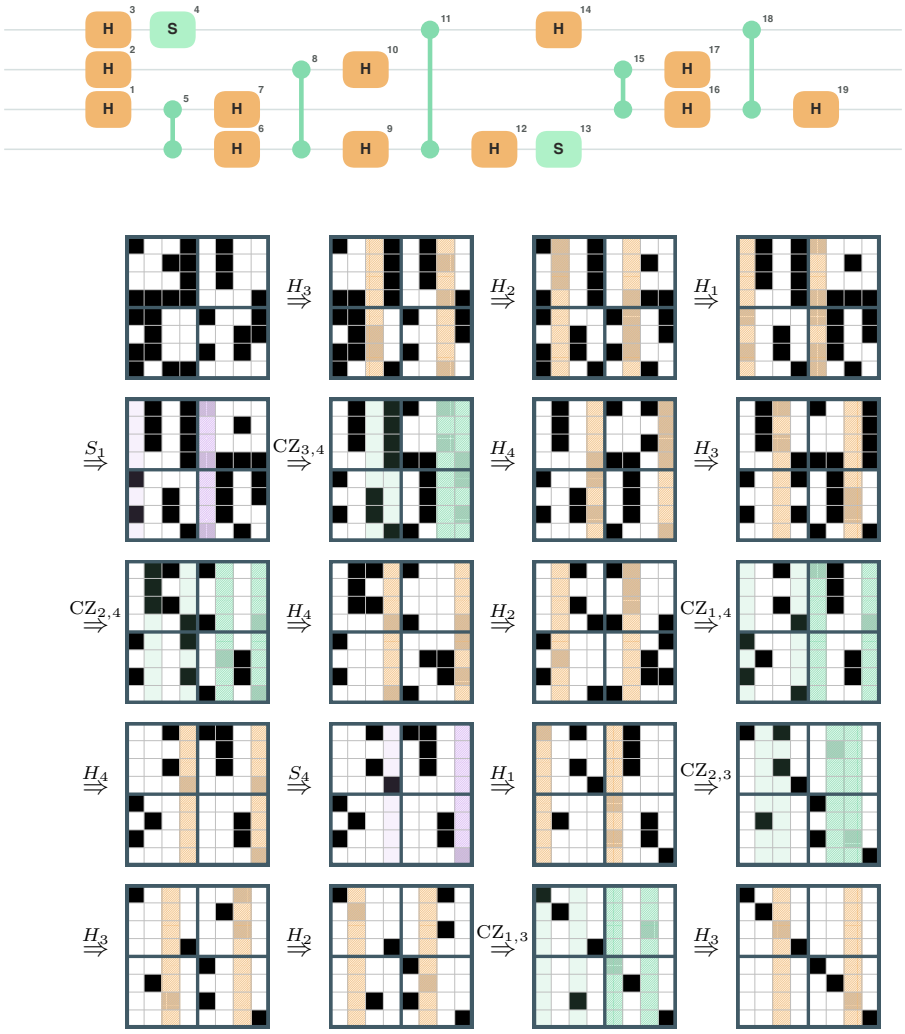


Figure A3: Full reduction sequence for the four-qubit target tableau in Figure 1. Each arrow is labelled by the next elementary gate in the displayed circuit, and the sequence ends at the identity tableau.

The *Clifford generators* are the one-qubit gates

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix},$$

and the two-qubit *controlled-Z gate*

$$\mathrm{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

A Clifford circuit can be characterized equivalently as a circuit that maps *Pauli operators* to Pauli operators under conjugation. This can be verified directly on the generators:

$$HXH = Z, \qquad HZH = X,$$
$$SXS^\dagger = Y, \qquad SZS^\dagger = Z,$$

and for the two-qubit gate CZ,

$$\mathrm{CZ}(X \otimes I)\mathrm{CZ}^\dagger = X \otimes Z, \qquad \mathrm{CZ}(Z \otimes I)\mathrm{CZ}^\dagger = Z \otimes I,$$
$$\mathrm{CZ}(I \otimes X)\mathrm{CZ}^\dagger = Z \otimes X, \qquad \mathrm{CZ}(I \otimes Z)\mathrm{CZ}^\dagger = I \otimes Z.$$

Here X_i, Y_i and Z_i denote the corresponding Pauli operator on qubit i and identity on every other qubit. A *Pauli string* is a tensor product of single-qubit Pauli operators; for example, on two qubits $Y_1 Z_2 = Y \otimes Z$.

Since each generator maps Pauli operators to Pauli operators under conjugation, any circuit built from these generators does as well. It follows that a Clifford operation can be specified exactly by how it transforms the Pauli generators $X_1, \ldots, X_n, Z_1, \ldots, Z_n$ under conjugation. Those generator images determine the full Clifford action up to global phase. The only consistency requirement is that these images preserve the original commutation relations: X_i commutes with X_j, Z_i commutes with Z_j, and X_i commutes with Z_j if and only if $i \neq j$; on the same qubit, X_i and Z_i anticommute.

The binary tableau $M_U \in \mathbb{F}_2^{2n \times 2n}$ records these generator images with their signs omitted. Each row records how an n-qubit Clifford operator U transforms one of $X_1, \ldots, X_n, Z_1, \ldots, Z_n$ under conjugation. This binary matrix is the state used by our synthesis agent. To record the signs as well, one can append a column $r_U \in \mathbb{F}_2^{2n}$, giving the signed tableau $(M_U \mid r_U)$. The generator images are Hermitian, so their phases are signs ± 1, represented by $(-1)^r$.

We encode each Pauli string using two bits per qubit: $(x_k, z_k) = (0,0), (1,0), (0,1), (1,1)$ represent I, X, Z, Y, respectively. The first n columns of M_U contain the X components and the last n columns contain the Z components. A signed row therefore has the form

$$(x_1, \ldots, x_n \mid z_1, \ldots, z_n \mid r).$$

For example, $(1,0 \mid 1,1 \mid 1)$ represents $-Y_1 Z_2$. The upper and lower halves of the tableau track the images of the X and Z generators, respectively, while its left and right halves record their X and Z components. These two divisions give four $n \times n$ blocks. With the sign column separated on the right, the full tableau has the form

	X components			Z components			Sign
X_1	$x_{1,1}$	$\cdots$	$x_{1,n}$	$z_{1,1}$	$\cdots$	$z_{1,n}$	r_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_n	$x_{n,1}$	$\cdots$	$x_{n,n}$	$z_{n,1}$	$\cdots$	$z_{n,n}$	r_n
Z_1	$x_{n+1,1}$	$\cdots$	$x_{n+1,n}$	$z_{n+1,1}$	$\cdots$	$z_{n+1,n}$	r_{n+1}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Z_n	$x_{2n,1}$	$\cdots$	$x_{2n,n}$	$z_{2n,1}$	$\cdots$	$z_{2n,n}$	r_{2n}

The binary tableau figures elsewhere in this paper omit the sign column.

Examples. On one qubit, H exchanges X and Z, while S maps X to Y and leaves Z unchanged. Their signed tableaus are shown below, with the corresponding Pauli image written beside each row:

P	x	z	r	$HPH^\dagger$
X	0	1	0	Z
Z	1	0	0	X

P	x	z	r	$SPS^\dagger$
X	1	1	0	Y
Z	0	1	0	Z

For a two-qubit example, use the generator ordering X_1, X_2, Z_1, Z_2. The conjugation rules for $\mathrm{CZ}_{1,2}$ give

$$X_1 \mapsto X_1 Z_2, \qquad X_2 \mapsto Z_1 X_2, \qquad Z_1 \mapsto Z_1, \qquad Z_2 \mapsto Z_2,$$

and hence

P	x_1	x_2	z_1	z_2	r	$\mathrm{CZ}\, P\, \mathrm{CZ}^\dagger$
X_1	1	0	0	1	0	$X_1 Z_2$
X_2	0	1	1	0	0	$Z_1 X_2$
Z_1	0	0	1	0	0	Z_1
Z_2	0	0	0	1	0	Z_2

For example, the first row $(1,0 \mid 0,1 \mid 0)$ represents $+X_1Z_2$: conjugation by CZ maps X_1 to X_1Z_2. Although all sign bits are zero for these generators, they need not remain zero when gates are composed. Applying S twice maps X to $-X$ and leaves Z unchanged, so $M_{S^2} = I_2$ but $r_{S^2} = (1,0)^T$.

Commutation and the symplectic condition. The product of two phase-free Pauli strings corresponds to addition of their binary row vectors over $\mathbb{F}_2$. For instance, XZ has binary encoding $(1,1)$, the same as Y up to phase. To determine whether two Pauli strings commute, we pair the X components of each with the Z components of the other. For row vectors $u = (x \mid z)$ and $v = (x' \mid z')$, this gives the symplectic inner product

$$u\Omega v^T = x \cdot z' + z \cdot x', \qquad \Omega = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix},$$

where all arithmetic is modulo two. The result is zero when the Pauli strings commute and one when they anticommute. Equivalently, for Pauli strings P and Q with binary encodings u and v,

$$PQ = (-1)^{u\Omega v^T} QP.$$

Conjugation preserves these relations: X_i anticommutes with Z_i, while all other pairs of distinct generators commute. Since the generator images are stored as rows, their pairwise symplectic inner products give

$$M_U \Omega M_U^T = \Omega,$$

equivalently $M_U^T \Omega M_U = \Omega$, as used in the main text. There are no further constraints on the binary part of the tableau: valid binary Clifford actions are precisely the elements of $\mathrm{Sp}(2n, \mathbb{F}_2)$.

Why we omit the signs. The agent operates only on M_U, which determines the Clifford operation up to a Pauli correction and global phase. The omitted signs can be restored by single-qubit Pauli gates without increasing the two-qubit gate count. Thus reducing M_U to the identity solves the phase-free synthesis problem studied here; if a signed target is specified, a final Pauli correction recovers its signs. The identity circuit has $M_I = I_{2n}$ and $r_I = 0$.

Composition and gate updates. Our row convention fixes the order of tableau multiplication. Applying a gate G after a circuit U gives the operator GU, and each stored image $P' = UPU^\dagger$ is updated to $GP'G^\dagger$. On the binary tableau this is right multiplication:

$$M_{GU} = M_U M_G.$$

Thus each gate updates every row independently. The following rules update only the binary matrix M_U; sign-bit updates are omitted, and $\oplus$ denotes addition modulo two. For the one-qubit generators, the update to each row affects only the coordinates of the qubit on which the gate acts:

$$H_i : (x_i, z_i) \mapsto (z_i, x_i), \qquad S_i : (x_i, z_i) \mapsto (x_i, x_i \oplus z_i),$$

with all other qubit coordinates unchanged.

For the two-qubit generator $\mathrm{CZ}_{i,j}$, the binary update couples only the coordinates of qubits i and j:

$$\mathrm{CZ}_{i,j} : \quad z_i \mapsto z_i \oplus x_j, \qquad z_j \mapsto z_j \oplus x_i,$$

with all other binary coordinates unchanged. Equivalently, appending $\mathrm{CZ}_{i,j}$ adds the X-support column of qubit j into the Z-support column of qubit i, and vice versa.

These local rules describe tableau reduction: every action available to the agent is an indexed generator, and applying that action performs an exact sparse update of the current binary state. In this phase-free binary representation, all three generator tableaus are involutions, including $M_S^2 = I$ over $\mathbb{F}_2$; this should be distinguished from the underlying quantum gate, where $S^2 = Z$ rather than the identity. Reaching the identity tableau therefore solves the group-reduction problem described in the main text, after which reversing the applied generators recovers a circuit with the original target's binary tableau. A final Pauli correction supplies any required signs.

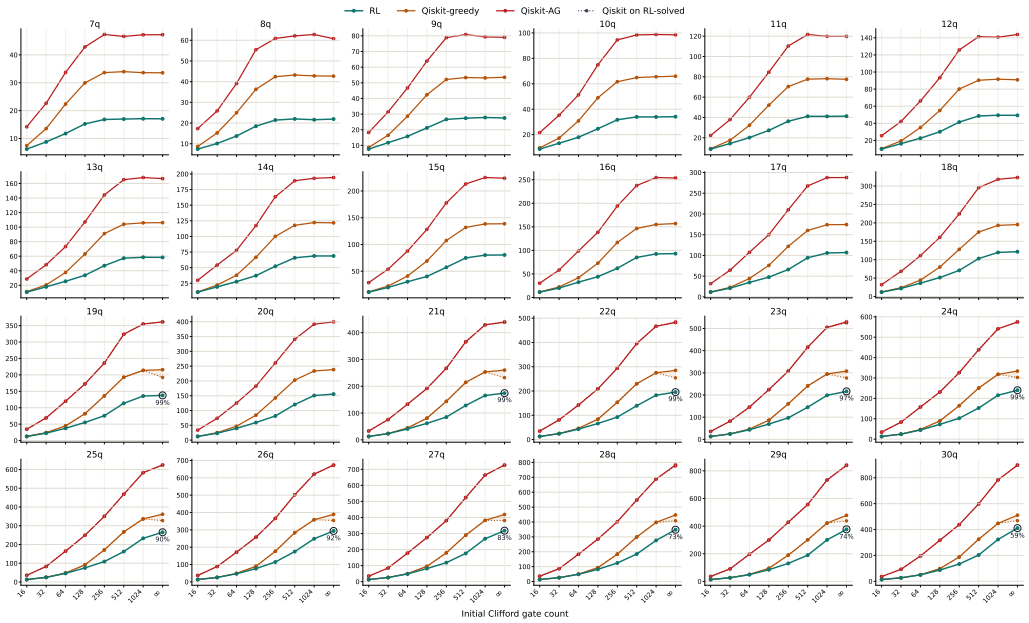


Figure A4: Full CZ-count sweep for the ten-qubit-trained model, expanding the main-text view in Figure 4 to every evaluation width from 7 to 30 qubits. Each panel fixes the evaluation qubit count and sweeps the displayed random-walk gate counts 16 through 1024, followed by the ∞ endpoint representing uniform Clifford sampling, under the same $6n^2$ rollout budget with the no-loop safeguard. Solid Qiskit curves show all-target means; dotted markers of the same colour and shaded gaps show the corresponding Qiskit means on the subset solved by the learned policy when these differ.

Evaluation subset	Cells	Succ.	6q model CZ	AG CZ	Greedy CZ	6q model lower CZ than AG / greedy
Std. 7–20 q.	98	1.000	52.97	114.06	65.14	98/98 / 93/98
Std. 21–30 q., $d \leq 128$	40	1.000	41.43	134.81	46.59	40/40 / 40/40
Std. 21–30 q., $d \geq 256$	30	1.000	383.56	479.53	263.19	23/30 / 10/30
Fully random	24	1.000	448.68	388.72	229.33	17/24 / 10/24

Table A1: Zero-shot extrapolation of the six-qubit-trained checkpoint on larger circuits. Each cell is one matched setting of target family, qubit count, and difficulty from the canonical evaluation package. Entries report mean CZ count over solved instances, and the six-qubit trained model solves all targets. The final row uses the fully random Clifford target family, which is tracked separately from the standard random-walk sweep.

D.1 Six-qubit Checkpoint Extrapolation

Figure A7 and Table A1 show the zero-shot behaviour of the checkpoint trained only on six-qubit targets. Under the same $6n^2$ no-loop rollout decoder used elsewhere, this checkpoint solves every larger-circuit target in the canonical sweep: all random-walk batches from 7 to 30 qubits and all fully random Clifford batches from 7 to 30 qubits. The cost is circuit quality rather than reliability. The six-qubit checkpoint remains strong on smaller and easier evaluation settings, often beating both Qiskit baselines in CZ count, but its CZ count rises sharply on high-depth and fully random targets with the most qubits. In contrast, the checkpoint further trained on ten qubits gives much lower CZ counts when it succeeds, but begins to fail on fully random targets beyond about 24 qubits.

E Decoding Details

Given a trained policy and a target tableau M_{target}, greedy decoding repeatedly applies the generator with the highest policy probability to the current tableau, stopping when the identity is reached or when the rollout budget is exhausted. The resulting reduction sequence is reversed to obtain a circuit

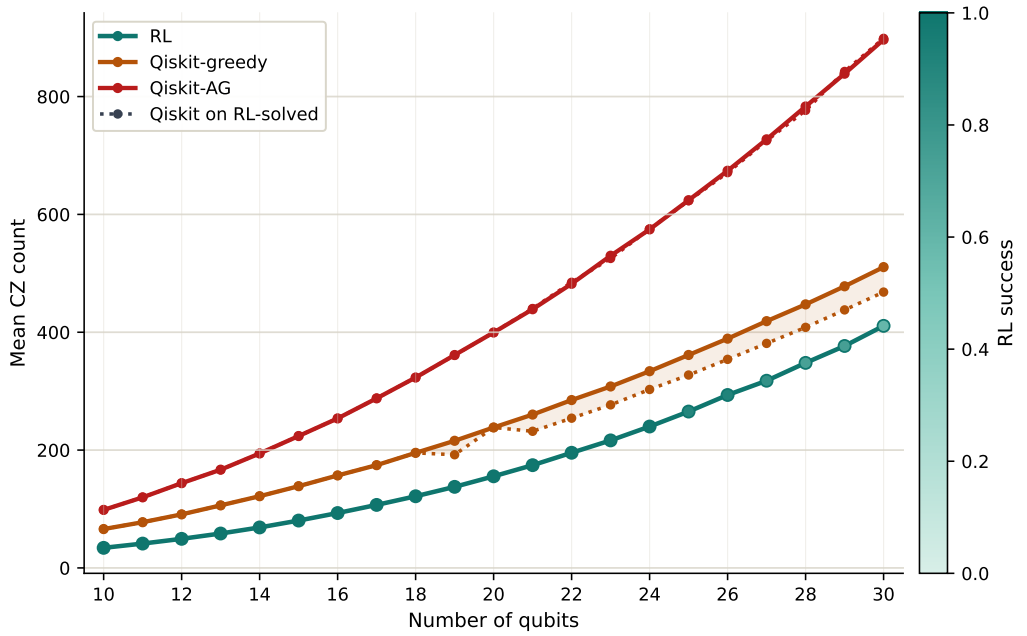


Figure A5: Uniform-target CZ-count comparison for the ten-qubit-trained checkpoint under the same $6n^2$ rollout decoder with the no-loop safeguard. The benchmark uses the separate fully random Clifford target family, with 100 held-out targets at each displayed qubit count from 10 to 30 qubits, and compares the learned policy against Qiskit's implementation of the Bravyi *et al.* greedy synthesizer and Aaronson–Gottesman on the same target sets. Learned means are computed over solved instances only, learned-policy markers are coloured by solve rate, solid Qiskit curves show all-target means, and dotted segments of the same colour show Qiskit means on the subset solved by the learned policy.

for M_{target}, as in the reverse-reduction formulation of Section 4. We also use the inverse-tableau trick of Gidney [19]: decode both M_{target} and M_{target}^{-1}, convert the inverse-tableau solution back to a circuit for M_{target}, and keep the shorter successful circuit. For sampled policy-guided rollouts, we repeat the same reduction procedure under a fixed compute budget and retain the shortest successful circuit found. In the larger-qubit sweeps, rollouts use a no-loop safeguard that rejects actions returning to a tableau already visited in the current trajectory when an alternative action is available.

All learned-policy targets and success checks use the phase-free binary symplectic tableau. For the ∞ target family generated with Qiskit's `random_clifford`, we store and evaluate only the returned `symplectic_matrix`; Pauli signs are not part of the target state. The Qiskit baselines are compared on the same target batches at the level of two-qubit cost: CZ and CX gates each count as one entangling gate, SWAP counts as three, and single-qubit, Pauli, and phase-correction gates are excluded from the reported CZ-equivalent counts.

For the six-qubit Bravyi *et al.* benchmark in Section 5, Table A3 gives the decoding settings behind the headline exact-recovery curve. The greedy pass evaluates the target and inverse tableaus once and keeps the shorter successful circuit. The longer search uses the same target/inverse comparison, a rollout cap of 512 generator applications, the native CUDA rollout backend, and 2048 parallel rollout environments.

It combines results from four temperature schedules: `t40`, `t40`→`greedy`, `t40`→`t25`, and `t40`→`t12`, where `t40` samples at fixed temperature 4.0, `t40`→`t12` samples with temperature linearly decreasing from 4.0 to 1.2, and `t40`→`greedy` decreases from 4.0 to 0.05. The fixed-temperature `t40` arm alone reaches $990/1003$ exact recoveries in the recorded first-hit trace; the annealed arms contribute the final five recoveries. Each arm runs 4096 trials per circuit for each of the target and inverse tableaus, so one arm corresponds to $1003 \times 4096 \times 2 = 8{,}216{,}576$ sampled reductions.

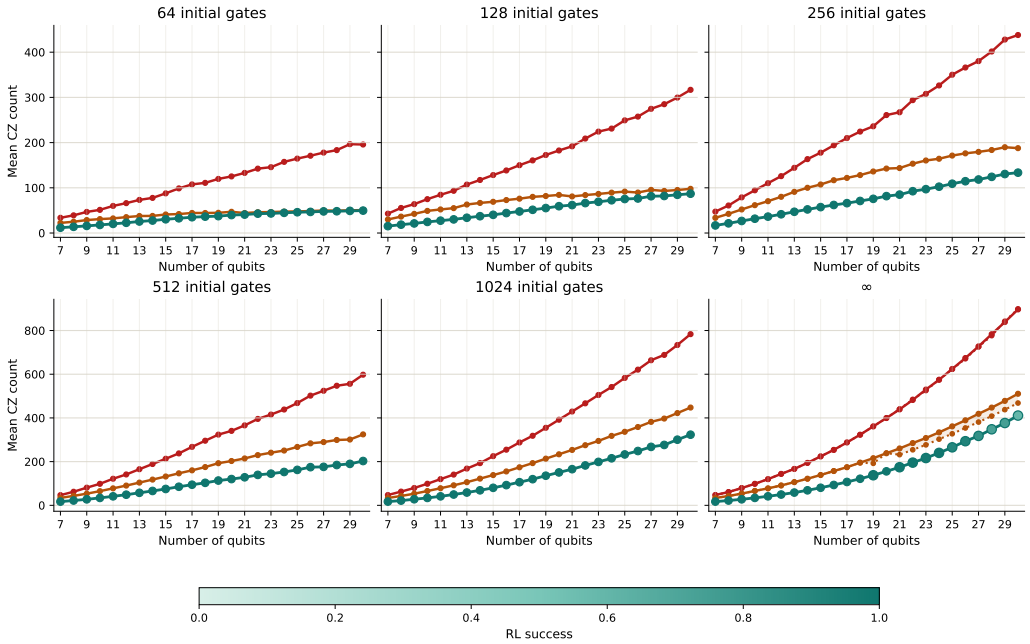


Figure A6: CZ-count comparison for the ten-qubit-trained model across the harder displayed random-walk gate counts 64, 128, 256, 512, and 1024, together with the ∞ setting for uniform Clifford targets, under the same $6n^2$ rollout budget with the no-loop safeguard. Each panel plots mean CZ count against evaluation qubit count on the full sweep from 7 to 30 qubits. Solid Qiskit curves show all-target means; dotted markers of the same colour and shaded gaps show Qiskit means restricted to targets solved by the learned policy when it does not solve every target.

The reported milestone times record when each recovery count was first reached across the four schedules: once an optimal circuit is found for a benchmark instance, later samples can still improve other instances, but the first-hit timestamp for that instance is fixed.

F Reproducibility, Compute, and Assets

The accompanying code and data release contains the code, trained checkpoints, fixed target batches, recorded result tables, figure-generation scripts, and a README with exact commands for the local evaluator, ablation validation, ensemble validation, the 1003-circuit benchmark, and the basic PPO training script. The bundled evaluator can run on CPU, with optional GPU acceleration for larger sweeps. The headline six-qubit search timings are recorded in the supplied first-hit trace and summary tables; the larger-qubit evaluation tables record the elapsed times of learned rollouts and Qiskit baselines for each setting. A live browser demo for the released six-qubit model, together with the recovered Bravyi *et al.* benchmark circuits, is hosted at `y-richie-y.github.io/clifford/`.

The work uses synthetic Clifford targets and public benchmark data from Bravyi *et al.*; no human-subject or scraped personal data are used. Existing software and benchmark assets are cited where used, and the code release identifies the bundled source trees, checkpoint files, and benchmark CSVs needed to reproduce the reported experiments.

The expected positive impact is improved Clifford synthesis, which may reduce two-qubit gate counts in quantum-compilation workflows. Possible negative impacts are indirect: better circuit synthesis can marginally reduce the cost of quantum experiments, including experiments whose downstream applications are outside the scope of this paper. The released assets are specialized to Clifford synthesis and do not contain sensitive data or general-purpose generative models.

Difficulty	Family	Success rate	Mean CZ	Seeds
$d = 20$	Ours	1.000	6.03 ± 0.01	3
$d = 20$	RelTransformer [39]	1.000	6.04 ± 0.01	3
$d = 20$	Transformer [45]	1.000	6.06 ± 0.03	3
$d = 20$	MLP	1.000	6.19 ± 0.01	3
$d = 20$	FlatMLP	1.000	6.14 ± 0.01	3
$d = 100$	Ours	1.000	11.99 ± 0.04	3
$d = 100$	RelTransformer [39]	1.000	12.06 ± 0.03	3
$d = 100$	Transformer [45]	1.000	12.50 ± 0.03	3
$d = 100$	MLP	1.000	13.89 ± 0.12	3
$d = 100$	FlatMLP	1.000	13.62 ± 0.16	3
$d = 1024$	Ours	1.000	13.71 ± 0.06	3
$d = 1024$	RelTransformer [39]	1.000	13.79 ± 0.15	3
$d = 1024$	Transformer [45]	1.000	14.37 ± 0.17	3
$d = 1024$	MLP	1.000	16.46 ± 0.09	3
$d = 1024$	FlatMLP	1.000	15.67 ± 0.15	3
$d = \infty$	Ours	1.000	13.49 ± 0.01	3
$d = \infty$	RelTransformer [39]	1.000	13.47 ± 0.06	3
$d = \infty$	Transformer [45]	1.000	14.12 ± 0.28	3
$d = \infty$	MLP	1.000	15.88 ± 0.05	3
$d = \infty$	FlatMLP	1.000	15.19 ± 0.16	3

Table A2: Full six-qubit architecture ablation over three seeds on fixed 100-target validation sets per difficulty. Finite d is the initial Clifford gate count used to generate each random-walk target; $d = \infty$ uses uniformly sampled Clifford targets. Entries report mean CZ count over solved instances under rollout decoding, as mean $\pm$ standard deviation across seeds; all completed-model success rates are 1.000.

Decoder stage	Exact / total	Time	Cumulative reductions	Step cap	Temperature schedules from 4.0
Greedy prefilter	507/1003	0.83 s	2,006	512	greedy
Brief search pass	610/1003	21 s	–	512	sampled policy rollouts
First reaches prior SOTA	982/1003	21.6 min	3,410,999	512	4.0
First reaches 990	990/1003	51.9 min	8,170,010	512	4.0
First reaches final count	995/1003	183.1 min	29,143,574	512	four schedules: 4.0, greedy, 2.5, 1.2

Table A3: Decoding settings for the six-qubit 1003-circuit benchmark. "Reductions" counts complete sampled reduction attempts, including both target and inverse tableaus when inverse comparison is enabled. Timings are total suite time, not per-instance time, from the recorded native CUDA rollout traces on a single NVIDIA A100 80GB GPU. The brief search pass brings every target to within one CZ gate of optimality, as reported in the main text. The full first-hit trace records the combined results of the four temperature schedules.

Hyperparameter	Value
Optimizer	PPO
Training steps	1.5×10^9
Parallel environments	2048
Rollout length	256
Curriculum range	difficulties 1–1000
Curriculum advance threshold	success $= 1.00$
Learning rate	2.5×10^{-4}
Batch size	8192
Epochs per update	5
Discount factor γ	0.99
GAE parameter λ	0.95
Policy clip coefficient	0.15
Value clip coefficient	0.2
`hamming_left` scale	0.5
Single-qubit gate cost	0.01
Terminal bonus	25

Table A4: Training hyperparameters for the main 6-qubit Clifford run.

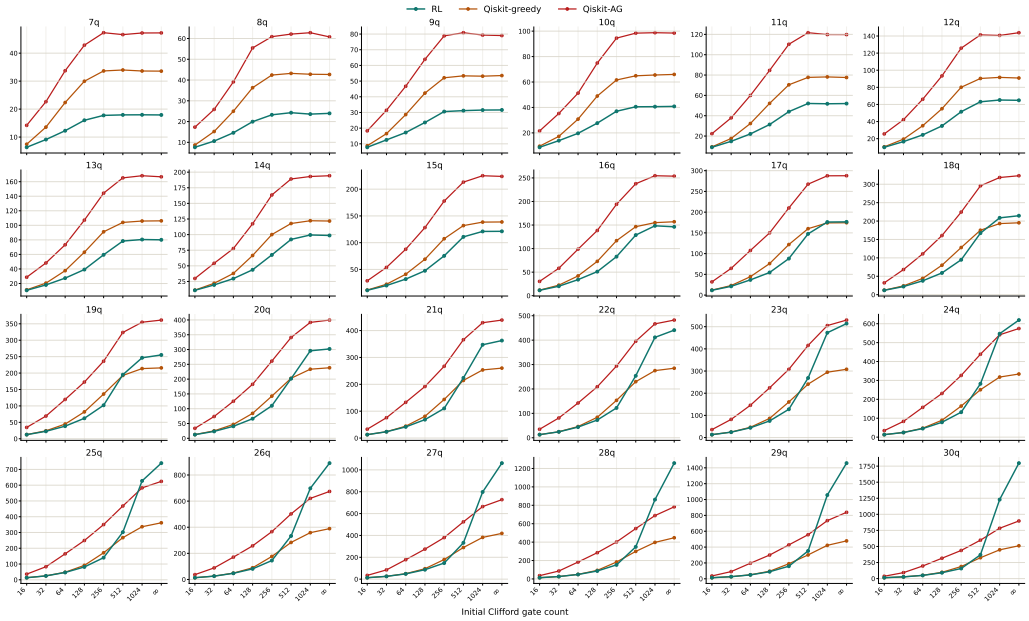


Figure A7: Full CZ-count sweep for the six-qubit-trained model, using the same layout as Figure A4. Each panel fixes the evaluation qubit count and sweeps the displayed random-walk gate counts 16 through 1024, followed by the ∞ endpoint representing uniform Clifford sampling, under the same $6n^2$ rollout budget with the no-loop safeguard. The checkpoint solves every target shown; the figure therefore directly compares mean CZ count against the two Qiskit baselines on the same target batches.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the benchmark sizes, success rates, and timing scope used in the experiments; for example, $99.2\% = 995/1003$ on the six-qubit benchmark.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 and the Discussion section describe the reliability–quality tradeoff, lack of optimality guarantees beyond six qubits, and scaling limitations of the action space.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The random-walk limit theorem and the equivariant-policy result state their assumptions and are proven in Appendices B and A.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method, target generation, decoding protocol, hyperparameters, and evaluation settings are specified in the paper and appendix; the supplement provides fixed targets, checkpoints, result tables, and commands for the main evaluations.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplementary reproducibility bundle includes the local evaluator, PPO source tree, model checkpoints, target batches, benchmark CSVs, analysis tables, figure scripts, environment setup, and reproduction commands.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 describes the MDP, reward, curriculum, and architecture; Section 5 specifies target generation and evaluation batches; Appendix E and Table A4 give decoding and training details.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The six-qubit benchmark reports exact counts over the full 1003-circuit benchmark, larger sweeps report solve rates and matched 100-target means, and the architecture ablation reports mean $\pm$ standard deviation over three seeds.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F and the supplement state that evaluations run on CPU with optional CUDA, provide recorded wall-clock timings for the headline search and sweeps, and give commands for reproducing the lightweight and full evaluations.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses synthetic Clifford targets and public benchmark data, does not involve human subjects or personal data, and releases specialized scientific software and checkpoints.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix F describes the positive compilation impact and the indirect dual-use risk that improved quantum compilation may reduce the cost of downstream quantum experiments.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets are specialized Clifford-synthesis code, synthetic targets, and small policy checkpoints, not high-risk generative models or sensitive datasets.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing benchmark data, Qiskit synthesizers, PufferLib, and related methods are cited in the manuscript; the supplement includes asset and license notes identifying bundled source trees, benchmark CSVs, and upstream dependencies.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The supplement documents the new checkpoints, generated target batches, result tables, figure data, and evaluation scripts, with README instructions and smoke checks.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods described in the paper do not use LLMs as part of the research contribution.