CNOT Minimal Circuit Synthesis:
A Reinforcement Learning Approach

Riccardo Romanello*, Daniele Lizzio Bosco[†‡], Jacopo Cossio[†], Dusan Sutulovic[†],
Giuseppe Serra[†], Carla Piazza[†] and Paolo Burelli[§]

Department of Environmental Sciences, Informatics and Statistics
Ca' Foscari University of Venice, Venice, Italy
[†]*Department of Mathematics, Computer Science and Physics*
University of Udine, Udine, Italy
[‡]*Department of Biology*
University of Naples Federico II, Naples, Italy
[§]*Play, Culture and AI Section*
IT University of Copenhagen, Copenhagen, Denmark
Email: riccardo.romanello@unive.it, lizziobosco.daniele@spes.uniud.it

Abstract—CNOT gates are fundamental to quantum computing, as they facilitate entanglement, a crucial resource for quantum algorithms. Certain classes of quantum circuits are constructed exclusively from CNOT gates. Given their widespread use, it is imperative to minimise the number of CNOT gates employed. This problem, known as *CNOT minimization*, remains an open challenge, with its computational complexity yet to be fully characterized. In this work, we introduce a novel reinforcement learning approach to address this task. Instead of training multiple reinforcement learning agents for different circuit sizes, we use a single agent up to a fixed size m. Matrices of sizes different from m are preprocessed using either *embedding* or *Gaussian striping*. To assess the efficacy of our approach we trained an agent with $m = 8$, and evaluated it on matrices of size n that ranges from 3 to 15. The results we obtained show that our method overperforms the state of the art algorithm as the value of n increases.

Index Terms—CNOT minimization, Reinforcement Learning, Linear Reversible Circuits, Quantum Circuit Synthesis.

I. INTRODUCTION

Quantum computing is a field at the intersection of physics and computer science. It explores the properties, limitations, and capabilities of computing devices whose internal operations are governed by the rules of quantum mechanics—quantum computers. With the term quantum computing paradigm, we refer to the set of principles and operations that define how quantum algorithms are constructed and executed. The most widely adopted quantum computing paradigm is the gate-based model, where quantum operations are represented by unitary matrices that, in turn, have a one to one correspondence to the so-called *gates*. However, practical quantum hardware imposes constraints on the direct implementation of many gates, necessitating their decomposition into smaller, hardware-compliant operations. This process is known as *quantum circuit synthesis* [1]–[3] Quantum circuit synthesis is computationally demanding, as an n-qubit gate

corresponds to a $2^n \times 2^n$ complex unitary matrix. Despite its general complexity, various subproblems within synthesis have been actively investigated, particularly those focusing on minimizing the occurrence of specific gates in a given universal gate set. A notable case is the minimization of the T-gate [4]–[6], a critical component for quantum advantage. The classical simulation of a quantum circuit is known to be exponential in the number of T-gates, making its reduction a fundamental objective [7], [8].

Another crucial operation in quantum computing is the Controlled-NOT (CNOT) gate. It is the only two-qubits gate in the Clifford+T set — one of the most popular choices for universal gate set — and is essential for achieving quantum entanglement. Reducing the number of CNOT gates is crucial for two reasons: (1) CNOT operations introduce significant noise due to their two-qubit interactions, and (2) they are a fundamental component of stabilizer circuits [9], which play a key role in quantum error correction [10]. Notably, stabilizer circuits exhibit a canonical form [11], where some computational blocks consist solely of CNOT operations. Hence, the length of a stabilizer circuit is deeply related to the length of these CNOT-only blocks, proving that minimising their length is a critical and relevant problem.

This leads to the *CNOT minimization problem*, which seeks to find an equivalent circuit with the minimal number of CNOT operations. This problem has been addressed in three different flavours: (i) any qubit can interact with any qubit; (ii) qubits are first mapped onto a *topology graph G*, and CNOTs are selected only among pair of qubits adjacent in G; (iii) ancilla qubits can be used during computation.

The *unconstrained* case can be formulated as follows.

Each CNOT gate can be represented as a binary matrix that is equal to the identity matrix except for a single off-diagonal entry indicating the qubits affected by the operation. Given an invertible binary matrix M and an elementary matrix $E_{i,j}$ corresponding to a CNOT between qubits i and j, applying the CNOT operation corresponds to the matrix multiplication

$M \leftarrow E_{i,j}M$. This transformation replaces row i of M with the bitwise XOR of rows i and j. Consequently, solving the CNOT minimization problem for a given matrix M reduces to finding the shortest sequence of such transformations that transforms M into the identity matrix. From this perspective, the task can be identified as a *planning problem*. Moreover, unlike general quantum synthesis, which operates on $2^n \times 2^n$ complex matrices, CNOT synthesis exclusively deals with matrices that have linear size with respect to n. This arises from the one-to-one correspondence between n-qubit CNOT circuits and $n \times n$ binary invertible matrices.

In this paper, we leverage these properties to address the unconstrained CNOT minimization problem with a Reinforcement Learning (RL) approach.

RL is a branch of Machine Learning that has shown remarkable effectiveness across a wide range of application domains, including autonomous control, strategic decision-making in board games, and even fine-tuning tasks in natural language processing. These techniques are based on the idea of *agents* that can act, by effectuating a choice between a set of possible *actions*, on a given *environment*, representing the current state of a problem. These actions are then (either positively or negatively) rewarded, based on their effect on the environment. Therefore, each RL technique can be described as the training of an agent that aims to maximize a reward.

The formulation of the CNOT minimization problems aligns naturally with RL, as: (i) the initial state of the environment is the starting matrix M; (ii) the set of actions corresponds to the possible CNOT operations; (iii) the maximal reward corresponds to finding a correct plan (as a sequence of actions) that leads to the identity matrix.

A general limitation of RL-based solutions is that every modification in the setting requires training a new agent, which is usually computationally expensive. Moreover, as the number of possible moves increases, so does the number of training samples required, highlighting scaling limitations.

To address this issue, in this work we combine an RL approach with matrix manipulations —specifically, sub-matrix embeddings and a greedy algorithm to "simplify" certain rows and columns — to enable the selection of a model trained on a given dimension m. When the model is applied to a matrix of a dimension lower or higher than m, the matrix is manipulated, allowing the initial model to find a suitable solution to the problem. We demonstrate the efficacy of our model, trained on $m = 8$, in addressing tasks of sizes ranging from 3 to 15, showing a clear advantage over the state-of-the-art heuristic algorithm Patel-Markov-Hayes [12] in most cases.

To ease further research in this area, we provide the code necessary to reproduce our experiments, including the agent and the algorithms required for matrices manipulation [1].

The paper is structured as follows. Firstly, in Section II, we introduce the reader to the state of the art for what concerns CNOT minimal circuits synthesis. The goal of Section III is to describe all the preliminaries required to fully grasp the

---

[1] https://github.com/RiccardoRomanello/CNOTRL

remainder of the paper. After that, we use Section IV to provide all the details about the proposed approach to address the CNOT minimization problem. The tests conducted to assess the efficiency of our method and the results we obtained are described in Section V. We draw some conclusions and propose potential follow-ups in Section VI.

II. RELATED WORKS

Quantum circuit synthesis has become an increasingly significant problem since the advent of physically operational quantum computers. A general synthesis algorithm takes as input a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$ and a universal gate set $\mathcal{B}$—with Clifford+T being the standard choice. The algorithm must then output a circuit composed solely of gates from $\mathcal{B}$ that implements U exactly or approximately.

Naturally, not all circuits are of equal interest. The primary objectives in quantum synthesis are: (i) minimizing circuit *depth* to mitigate decoherence effects due to the fragile quantum state, and (ii) minimizing circuit *width* to limit the number of required qubits.

Comprehensive studies on quantum circuit synthesis in its broadest scope can be found in [4], [13]–[15]. A Reinforcement Learning approach has been adopted for the synthesis of unitary matrices in its broadest scope in [16], [17].

However, synthesis of Clifford+T circuits can be tackle in terms of subproblems, including the minimization of specific gate types within the universal set. One such problem is the *CNOT gate minimization*.

An algebraic approach to CNOT minimization is presented in [18], where the authors first describe the group structure underlying quantum circuits composed of CNOT gates. These structural insights are then leveraged to devise heuristics aimed at reducing the number of CNOT operations in a given circuit.

A different perspective based on *Steiner Trees* is explored in [19]. This approach considers the practical constraints imposed by hardware connectivity, as real-world quantum devices do not typically support all-to-all qubit interactions. In this context, a CNOT gate between two qubits is feasible only if the qubits are physically connected. If they are not, additional *swap* gates—equivalent to three CNOT operations—must be introduced, further increasing circuit depth.

Another line of research tackles the problem using tools coming from logic programming. On the one hand, we have works like [20], [21] that encode the problem as SAT instance. On the other hand, in [22], [23] authors propose a solution to both the unconstrained and constrained version with an Answer Set Programming approach.

Regarding the application of Reinforcement Learning (RL) to quantum circuit synthesis, we highlight three key studies. Two of them [24], [25] focus on training RL agents to solve the general quantum circuit synthesis problem. Despite their promising results, the task does not allow for an efficient representation, and is subjected to strict scaling limitations.

The third study, [26], specifically addresses CNOT minimization using RL. However, a crucial distinction between this work and our approach is that [26] considers the *constrained*

variant of the problem, obtaining a different set of actions. Moreover, different topologies were addressed by different agents. Finally, each setting had at most 4 qubits, while in our proposed method we use up to 15 qubits.

Our work builds upon these studies by leveraging Reinforcement Learning for unconstrained CNOT minimization while integrating Answer Set Programming to preprocess larger instances, thus enabling the RL agent to operate effectively on manageable problem sizes.

III. BACKGROUND

A. Setting the Context: CNOT-Minimization Problem

Modern gate-based Quantum Computers have two important limitations:

- Only a finite set of operations can be applied to the qubit
- The quantum coherence can be maintained for a short amount of time

The finite set of operations a Quantum Computer can apply is usually called *Universal Set of Gates* $\mathcal{B}$. Quantum Algorithms are usually described as a single $2^n \times 2^n$ unitary matrix $\mathcal{U}$.

Rewriting $\mathcal{U}$ in terms of gates in $\mathcal{B}$ is called *synthesis*.

The most common choice for $\mathcal{B}$ is the Clifford + T set, which contains the single-qubit H, S, and T gates, and the two-qubit gate CNOT:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1)$$

Single qubit gates can be implemented in physical quantum computers with no errors. On the other hand, two-qubit gates introduce errors that must be corrected using Quantum Error Correction (QEC) techniques. Hence, it is reasonable to think about synthesis techniques that aim at minimizing the number of CNOT gates in the resulting circuit.

CNOT's graphical counterpart is depicted in Figure 1.

Fig. 1: CNOT gate in the circuit settings. The top qubit q_0 is the control, while q_1 is the target.

The two qubits on which a CNOT acts are usually referred to as *control* and *target*. Roughly speaking, the effect of a CNOT is to *flip the target qubit if and only if the control qubit is one*. The behaviour of a CNOT gate is the same as the xor logical operation, that returns 1 if and only if exactly one of the two arguments is 1. Another example of a gate that acts in a similar manner is Toffoli, which in turn takes as input three qubits. The third one is flipped if and only if both the other two are one. Composing CNOT and TOFFOLI allows to obtain circuits that flip the target qubit if and only if the controls hold some specific truth value.

As we mentioned before, the application of a CNOT —and, in general, of a multi-qubits gate—must be done carefully. This because physical implementation of two qubits gates we are provided with are intrinsically faulty and can produce mixed states even if the input state was originally pure. Therefore, it is natural to think about reducing the number of two qubits gates—CNOTs—we adopt when we rewrite a unitary in terms of smaller terms. The **CNOT-count problem** arises here: find the implementation of a unitary matrix U with the minimum number of CNOT gates.

This kind of problem is not important just to minimize the noise inside circuits, but also for its effect on stabilizer circuits. These particular circuits are the ones that can be produced just by using gates inside the Clifford group. This set is not universal, but it describes perfectly which quantum algorithms can be simulated efficiently with classical hardware [27].

A subsequent result about stabilizer circuits comes from [28] in which authors proved the existence of a *normal form*.

Theorem 1 ([28]). *Let $\mathcal{C}$ be a circuit consisting of CNOT, Hadamard and phase gates—a stabilizer. There exists an equivalent circuit with the following normal form:*

$$H - C - P - C - P - C - H - P - C - P - C$$

where an H, C, and P indicates a layer of Hadamard, CNOT, and Phase gates, respectively.

Layers of Hadamard and Phase gates have length exactly one—being single qubit gates, they can all be parallelized. On the other hand, the whole length of the circuit depends on the length of the CNOT layers. Therefore, it is crucial to produce algorithms that can minimize the number of such gates.

Circuits composed only of CNOT gates are usually called either *CNOT circuits* or *Linear reversible circuits*. Since the effect of a CNOT gate is intrinsically classical, CNOT circuits can be represented using classical tools. In particular, any linear reversible circuit on n qubit can be completely described by an $n \times n$ boolean invertible[2] matrix M. As the reader may notice, we converted a problem on n qubit—hence, with a representation that has dimension $2^n \times 2^n$—to a problem on a $n \times n$ matrix. Because of this shift in dimension, the definition of CNOT we gave in Eq. 1 is not coherent anymore.

Definition III.1. Let M be an $n \times n$ matrix representing a linear reversible circuit. The effect of applying a CNOT with i-th row as a control and the j-th row as a target is to mupliply M by a matrix M' defined as follows. M' is the identity matrix with an additional off-diagonal 1 element in position i, j.

This particular definition of the effect of a CNOT gate, allows us to treat it as an elementary row operation.

Lemma III.1. *The action of a CNOT gate, whose control is on the i-th row and target is in the j-th row, is an elementary row operation on a $n \times n$ matrix, which adds the i-th row to the j-th row.*

---

[2]Notice that if the matrix is not invertible, than the circuit cannot be decomposed in terms of CNOT gates.

The sum must be interpreted in modulo 2. Before formally stating the CNOT minimization problem, we introduce the following theorem:

Theorem 2 ([29]). *Any invertible matrix can be transformed into an identity matrix using elementary row and/or column operations. Thus, any invertible matrix can be decomposed as a product of elementary matrices.*

Therefore, any invertible matrix M—linear reversible circuit—can be decomposed into the identity matrix using only CNOT gates.

Definition III.2. Let M be an invertible binary matrix. The CNOT minimization problem is the problem of finding the shortest sequence of k matrices $M_1, M_2, \ldots, M_k$ such that $MM_1M_2 \ldots M_k = \mathbb{I}$ where each M_i is the matrix of a CNOT.

Dually, it can also be defined as the problem of turning the identity matrix into M applying only CNOT gates.

Before moving to the complexity topics related to CNOT minimization, we would like to stress one peculiar aspect about CNOT metric. In our settings, a CNOT is thought to be applied to rows i, j and the result overwrites one of the two rows. With this definition, we are calculating the *s-XOR* metric. In the literature, however, *g-XOR* exists as well. In this case, a CNOT takes into account three indices k, i, j and the result of the XOR between i, j is stored in row k.

As previously mentioned, we focus on s-xor metric.

To conclude this section, we want to clarify that as stated in [30]: *The computational complexity of the CNOT-optimal circuit synthesis (with no constraints on the topology) is yet to be established.*

B. Patel-Markov-Hayes in a Nutshell

The Patel-Markov-Hayes (PMH) algorithm, introduced in [12], derives its name from its three authors. While initially not widely adopted, the increasing interest in quantum computing has led to a renewed focus on circuit optimization techniques. The significance of CNOT gate minimization, coupled with the normal form introduced for stabilizer circuits (Theorem 1), has established PMH as a state-of-the-art[3] method for CNOT minimization [31].

Let $M \in \{0,1\}^{n \times n}$ be an invertible Boolean matrix over the field $\mathbb{F}_2$. The PMH algorithm operates by processing the matrix in *stripes* of width k, where k is an integer in $\{1, 2, \ldots, n\}$. Conceptually, the algorithm partitions M into $\lceil n/k \rceil$ vertical stripes and iteratively processes them. The goal is to transform the first stripe, corresponding to columns 0 through $k-1$, into an upper triangular form by ensuring:

- $M_{i,i} = 1$ for all $i \in \{0, 1, \ldots, k-1\}$ (diagonal elements set to 1).
- $M_{i,j} = 0$ for all $i \in \{0, 1, \ldots, k-1\}, j > i$ (zeroing out elements below the diagonal).

---

[3]Not in terms of gate count: it is the only heuristic-based algorithm whose worst-case performance is proven to match the asymptotic optimum of $\mathcal{O}(n^2/\log n)$.

This transformation is achieved in two phases. First, rows with identical entries in the first k columns are combined via XOR operations to reduce redundancy. Second, a Gaussian elimination procedure is applied, ensuring that all elements below the diagonal are set to zero while maintaining ones on the diagonal. Once all stripes have been processed, the resulting matrix M' is in an upper triangular form.

The algorithm then transposes M' and reapplies the same procedure. To preserve the triangular structure obtained before transposition, it is crucial that row operations are performed such that the row with the smaller index serves as the control, preventing the introduction of new ones in previously zeroed regions of the matrix.

IV. METHOD

In the previous section we introduced the CNOT-Minimization task. Given an invertible boolean matrix $M \in \{0, 1\}^{n \times n}$, we want to turn it into the identity matrix using only elemental row operations—CNOTs.

Both exact and heuristic-based methods struggle to deal with this problem. The former class fails to scale when the size of the problem grows [22]. On the other hand, the latter type produces a gate count that is far from the optimum [12].

In this paper we propose a Deep Reinforcement-Learning (RL) method to address the CNOT-minimzation problem. Deep RL approaches are becoming more and more prominent when dealing with circuit synthesis [24], [25] because of the lack of natural methods to obtain ground truths. This particular fact does not allow the application of supervised learning techniques.

Our approach to CNOT minimization employs a PPO2 [32] agent implemented using the *stable-baselines 3* framework [33]. The agent is trained to iteratively apply CNOT operations to reduce a given matrix to the identity. The agent operates within a discrete action space of $\mathcal{O}(n^2)$ possible moves, where each move is defined as a tuple (i, j), corresponding to the application of a CNOT gate that updates row i by performing an XOR operation with row j.

Since an RL agent is trained on a specific state space, it is generally not possible to train a single model on matrices of different sizes. Consequently, a separate model, along with a new training process, is required for each n, significantly limiting the approach usability and scalability. To address this limitation, we apply matrix manipulation techniques that enable training a model on a fixed size while allowing the agent to be applied to matrices of different sizes.

A. Curriculum Learning Strategy

Due to the structural properties of the CNOT minimization problem, we adopt a curriculum learning approach to progressively increase the complexity of the training instances. The training process consists of the following stages:

1) **Permutation Matrices:** The agent first learns to optimize CNOT sequences for permutation matrices, which form a fundamental subset of invertible binary matrices.

2) **Triangular Matrices:** Once the agent demonstrates proficiency in handling permutation matrices, it advances to upper and lower triangular matrices. These matrices introduce additional structure while remaining relatively tractable.

3) **Generic Matrices:** In the final stage, the agent is trained on arbitrary invertible binary matrices, ensuring its capability to generalize to complex instances.

As the reader may note, matrices are in some sense *sorted* in terms of complexity.

Injecting our knowledge about the problem, we assume permutation matrices to be the easier to turn into the identity. Subsequently, we place the set of triangular (either upper or lower) matrices. Having one entire portion of the matrix set to 0 is clearly an easier setting than a completely generic matrix.

That being the case, the model is exposed to randomly generated matrices only after the two above mentioned phases are completed.

B. Gaussian Striping

As we mentioned before, increasing the size of the matrices handled by the agent require more and more training time. Therefore, we propose a preprocessing step that, given a model $\mathcal{A}$ that solves the CNOT minimization problem for dimension m, allows to solve instances with $n > m$. We refer to this process as *Gaussian striping*, since, informally, it edits a stripe of the input matrix adopting a gaussian-like process.

Let $M \in \{0,1\}^{n \times n}$ be a boolean invertible matrix with $n > m$. The modification consists of applying a slightly altered PMH-like approach to a stripe of width $k = n - m$, effectively reducing the first k columns into an upper triangular form. The same procedure is then applied after transposing the matrix, resulting in a block-diagonal structure:

$$M'' = \begin{bmatrix} I_k & 0 \\ 0 & M' \end{bmatrix},$$

where I_k is the $k \times k$ identity matrix and M' is an $(n-k) \times (n-k)$ invertible Boolean matrix. The transformed matrix M' has its first k columns reduced to an identity block, allowing us to isolate the remaining M' submatrix for further optimization.

The modified PMH procedure is shown in Algorithm 1.

Algorithm 1 Gaussian Striping Algorithm

1: **function** REDUCE(M)
2:     Apply PMH to the first $k = n - m$ columns of M.
3:     Transpose M.
4:     Apply PMH to the first $k = n - m$ columns of the transposed M.
5: **end function**

The total CNOT count required to transform M into the identity matrix is the sum of:

- The CNOT count required by the preprocessing to produce M'', and
- The CNOT count required by $\mathcal{A}$ to reduce M' to the identity.

Notice that this approach has no scalability issues since PMH tweaked version we apply has complexity $\mathcal{O}(n^3)$ on a $M \in \{0,1\}^{n \times n}$ matrix.

C. Embedding Small Matrices

Let $\mathcal{A}$ be an agent capable of solving the CNOT minimization on matrices $M \in \{0,1\}^{m \times m}$.

For matrices $M' \in n \times n$, with $n < m$ we employ an embedding strategy that maps the problem into an $m \times m$ space while preserving its structure. Given an invertible Boolean matrix $M' \in \{0,1\}^{n \times n}$ with $n < m$, we construct an augmented matrix M'':

$$M'' = \begin{bmatrix} I_k & 0 \\ 0 & M' \end{bmatrix},$$

where $k = m - n$. This embedding ensures that M'' is an $m \times m$ invertible matrix while preserving the original problem structure.

The resulting matrix M'' is then given as input to the agent $\mathcal{A}$, which optimizes the CNOT sequence. This approach allows us to unify the treatment of matrices with dimensions up to $m \times m$ while ensuring that smaller instances remain compatible with our RL-based optimization framework.

V. EXPERIMENTAL EVALUATION

To evaluate the performance of our proposal, we designed a series of experiments with varying levels of difficulty and different problem sizes. As the problem size increases, so does the complexity of training an RL agent, both in terms of computational cost and the number of required epochs. For this reason, we conducted training on size $m = 8$. This choice was made for two reasons. First, it enables fast training and evaluation. Second, it has been shown in [20], [22] that problems of this size begin to become intractable for exact methods and require heuristic approaches.

We compare our method with the original PMH algorithm for two main reasons: (i) it is one of the most widely used—implemented as a native function in QISKIT, (ii) it has provable guarantees about its worst-case performance [34].

A. Agent Training

This part of the paper is devoted to the description of the value adopted for the agent's training phase. This step took around one and a half day to complete.

1) Training Schedule and Episode Allocation: As we mentioned before, during the training phase we adopted a curricular approach exposing the model to matrices of growing complexity. A curriculum is composed of a certain number of episodes. During each episode, the model is provided with exactly one matrix.

We now describe the curriculum ranges, in terms of number of episodes and matrix complexity.

- The initial training phase focuses on *permutation matrices* and *triangular matrices*. A total of 3000 episodes are allocated to this phase, evenly distributed among the two classes, with 1000 episodes dedicated to each.

- From episode 3000 to episode 6000, training continues with matrices randomly sampled from permutation and triangular ones, reinforcing the agent's ability to generalize over these structures.
- Between episodes 6000 and 10000, the model transitions to handling *randomly generated matrices*, specifically those created using $n/2$ CNOT operations.
- The training phase from episode 10000 to 20000 is dedicated to matrices generated with n CNOT operations, increasing the problem complexity.
- More complex cases arise in subsequent stages: matrices generated using $n \log n$ CNOT operations are introduced between episodes 20000 and 50000. Matrices requiring n^2 operations are used for the final 50000 episodes.

This structured progression ensures that the RL agent is exposed to increasingly complex instances in a controlled manner, facilitating a gradual adaptation from simpler structures to highly populated matrices. By proportionally increasing the number of episodes allocated to more complex cases, we enhance the model's ability to optimize CNOT sequences effectively across different problem instances.

2) Reward Function Design: Our reward function is designed to minimize the introduction of external problem-specific heuristics, ensuring that the model learns an optimized strategy purely through interaction.

To achieve this, we structure the reward as follows:

- A reward of $+0.7$ is assigned whenever the model successfully *solves* a matrix, meaning that it transforms it into the identity matrix.
- If the matrix is not fully solved, we compare the states before and after the applied move. Let M_1 and M_2 denote the matrices before and after the action, respectively.

At this point, we distinguish between two cases. If the Hamming distance between M_1 and the identity matrix differs from that of M_2 and the identity, the reward function considers the number of diagonal and off-diagonal ones in both matrices. The assigned reward is $0.2 \times d/n - 0.1 \times \bar{d}/n^2$, where:

- d represents the change in the number of ones on the diagonal between M_1 and M_2.
- $\bar{d}$ represents the same difference but for off-diagonal ones.

This formulation ensures that the model is penalized for introducing extra off-diagonal ones or removing diagonal ones. Conversely, it is rewarded for eliminating off-diagonal ones and correctly placing ones on the diagonal.

On the other hand, if the applied move does not alter the Hamming distance from the identity matrix, the action is considered unproductive, and a small penalty of $-0.001/n^2$ is applied. By structuring the reward function in this way, we encourage the model to prioritize transformations that reduce circuit complexity while avoiding redundant operations.

B. Setting

To evaluate our approach, we considered sizes ranging from $n = 3$ to $n = 15$ (corresponding to 3 and 15 qubits, respectively). For each size, we generated a set of 100 matrices by applying a setting-dependent number of CNOT gates. The following settings, increasing in complexity, were used:

- **Rare:** Generated by applying $n/2$ CNOT.
- **Medium:** Generated by applying $n \log n$ CNOT.
- **Overcooked:** Generated by applying n^2 CNOT.

For each setting, we compare the PMH algorithm with our proposed RL agent, which was trained on matrices of size 8×8. For sizes $n \leq 8$, we applied the agent after matrix embedding the initial instance in an 8×8 matrix. Conversely, for $n \geq 9$, we employed the dimensionality reduction technique, as described in Section IV, first reducing the matrix size and then applying the RL agent. In these cases, we report the total number of CNOT gates required, including both the reduction step and the agent's output.

Finally, to account for the inherent stochasticity of RL-based approaches, we evaluated the agent over 100 runs, retaining only the best results.

C. Evaluation

For each setting and matrix size, Table I reports the average number of CNOT gates required, along with the corresponding standard deviation. A graphical summary of the results is provided in Figure 2.

An initial observation is that for $n \leq 7$, the average number of CNOT gates required by both approaches remains comparable across all settings. However, as noted in [20], [22], for these problem sizes, the task can be exactly solved via logic programming, SAT solvers, etc. Consequently, heuristic-based methods are less relevant in this regime.

For $n = 8$, significant differences begin to emerge, particularly in the *Medium* and *Overcooked* settings, where the RL agent achieves an average reduction of approximately 3 and 2 CNOT gates, respectively, compared to PMH.

For $n \geq 9$, our proposed method consistently outperforms PMH across all three settings. Notably, this advantage is independent of the specific strategy used to generate the test instances, highlighting the robustness of our approach.

To further analyze the performance for $n \in \{9, \ldots, 15\}$, Figure 3 presents violin plots depicting the distribution of CNOT counts for each setting. In the *Rare* setting, PMH exhibits a few outliers contributing to a higher average CNOT count. However, the overall distributions indicate that our method consistently achieves lower CNOT counts, with a performance gap that widens as n increases.

In the *Medium* and *Overcooked* settings, the difference between the distributions becomes even more pronounced. For higher values of n, our approach consistently discovers circuits with a reduced number of CNOT gates. This trend is particularly evident starting from $n = 11$ in the *Overcooked* setting, where even the worst-performing solutions generated by our method still require fewer CNOT gates than the best solutions obtained via PMH.

D. Gaussian Striping applied to PMH

As part of our ablation study, we briefly compare the impact of our proposed method for $n \geq 9$, i.e., in cases where the

Setting Size	Rare		Medium		Overcooked	
	RL	PMH	RL	PMH	RL	PMH
3^-	2.13 ± 0.34	1.61 ± 0.95	3.40 ± 1.32	3.39 ± 1.41	3.60 ± 1.24	3.52 ± 1.51
4^-	3.01 ± 1.15	2.84 ± 0.97	6.92 ± 8.42	5.76 ± 2.07	7.19 ± 1.74	6.98 ± 1.90
5^-	2.95 ± 1.01	2.84 ± 0.86	10.37 ± 2.43	10.26 ± 2.35	11.84 ± 7.80	11.07 ± 2.37
6^-	4.00 ± 1.18	3.98 ± 1.26	14.16 ± 5.67	14.52 ± 3.38	16.20 ± 7.21	16.40 ± 2.78
7^-	4.09 ± 1.08	4.03 ± 1.20	18.87 ± 9.59	19.04 ± 4.35	22.61 ± 9.01	22.62 ± 3.64
8	5.25 ± 1.37	5.29 ± 1.26	22.82 ± 3.52	26.27 ± 5.34	28.02 ± 6.41	30.58 ± 4.07
9*	4.29 ± 1.02	4.21 ± 1.00	32.00 ± 5.26	35.55 ± 7.49	36.19 ± 5.28	40.94 ± 5.36
10*	5.63 ± 1.20	5.55 ± 1.28	40.05 ± 5.87	46.22 ± 7.63	45.58 ± 7.34	52.10 ± 7.04
11*	5.64 ± 1.37	5.50 ± 1.21	48.34 ± 6.87	57.18 ± 9.66	55.29 ± 8.47	66.98 ± 7.57
12*	6.86 ± 1.68	6.47 ± 1.62	58.88 ± 7.21	70.42 ± 9.48	63.99 ± 5.02	82.77 ± 7.80
13*	6.98 ± 1.39	6.45 ± 1.10	66.88 ± 10.15	85.82 ± 13.37	77.35 ± 6.01	103.83 ± 8.26
14*	8.17 ± 1.89	7.62 ± 1.47	79.00 ± 11.86	101.41 ± 15.29	90.95 ± 7.84	123.89 ± 9.61
15*	8.43 ± 1.49	7.93 ± 1.38	86.83 ± 11.50	118.46 ± 17.07	106.70 ± 8.70	148.57 ± 10.04

TABLE I: Average CNOT Count for the considered sizes and settings. For sizes lower than 8, denoted with $^-$, the submatrix embedding is used before applying the RL method. For sizes higher than 8, denoted with *, is applied the Gaussian Stripe algorithm (Algorithm 1). Values are presented as mean ± standard deviation.
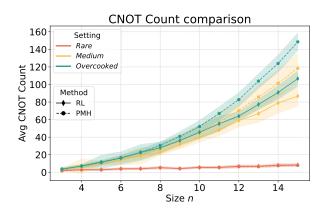

Fig. 2: Comparison between the number of CNOT generated by PMH and RL approaches as n grows, for the three different settings considered.

Size	Rare	Medium	Overcooked
9	-0.05	3.29	2.78
10	0.01	2.57	2.00
11	0.02	3.08	1.41
12	-0.17	2.29	3.19
13	-0.15	1.66	2.99
14	-0.21	1.24	3.01
15	-0.30	3.08	2.14

TABLE II: Difference in number of CNOTs used between our method and PMH*, corresponding to first reduce the matrix to a size 8×8, and then using the standard PMH algorithm.

gaussian striping is applied, with the PMH algorithm on the reduced matrices. This comparison aims to determine whether the advantage in CNOT counts arises solely from the matrix manipulation technique or from the RL agent.

We report the difference between the average number of CNOTs used by our method and by the PMH* algorithm (i.e., PMH applied to the reduced matrices obtained via gaussian striping) in Table II. In the *Rare* setting, the difference between the two approaches is close to zero, while for *Medium* and *Overcooked*, we observe a reduction of approximately $1 \sim 3$ gates. This result is consistent with those presented in Table II for $n = 8$, as the advantage of our RL agent depends only on the transformations applied to matrices of the initial size.

VI. CONCLUSIONS AND FUTURE WORK

The gate-based model of quantum computation requires quantum algorithms to be expressed in terms of fundamental operations known as quantum gates. Among these, the CNOT gate plays a crucial role, acting on two qubits and serving as the quantum analogue of the classical XOR operation.

CNOT gates are particularly significant in stabilizer circuits, a fundamental component of Quantum Error Correction. However, due to their multi-qubit nature, they introduce noise into the quantum system, necessitating their minimization. This leads to the *CNOT-minimization problem*, which seeks to transform a given CNOT circuit into an equivalent one with the minimum number of such operations.

In this work, we proposed a Reinforcement Learning-based approach to address this problem. We trained an RL agent capable of solving instances on 8 qubits—matrices of size 8×8. For circuits with less than 8 qubits, we embedded their representation in a higher dimension matrix amenable to be solved by the RL agent. On the other hand, for circuits exceeding this size, we introduced a preprocessing step, called Gaussian Striping, that partially resolves the problem by reducing the input instance to an 8-qubit subproblem.

To evaluate our approach, we conducted extensive testing on circuits up to 15 qubits, comparing our results against the Patel-Markov-Hayes (PMH) algorithm, the state-of-the-art method for CNOT minimization. Our findings demonstrate that our RL-based technique outperforms PMH when the number of qubits exceeds 8, highlighting its effectiveness for larger problem instances.

We observed that, when comparing RL with PMH combined with Gaussian Striping, the reduction in CNOT count mainly
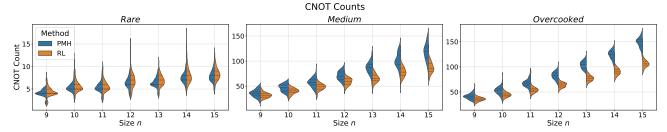
Fig. 3: Comparison between the distribution of CNOTs used by PMH and RL for $n \geq 9$, for the settings considered.

depends on the agent's performance on matrices of the size used for training. Hence, a promising direction is to optimize the RL pipeline to enable training on larger initial sizes.

Alternatively, another research avenue involves improving the preprocessing step to enhance its efficiency, minimize the number of additional CNOT gates introduced during problem reduction, and potentially extend its applicability to different synthesis approaches. This is supported by the improved performance of PMH after the application of Gaussian Striping.

By advancing in these directions, we aim to enhance the applicability of RL methods for circuit synthesis, contributing to more efficient quantum computing architectures.

REFERENCES

[1] V. Kliuchnikov, "Synthesis of unitaries with clifford+t circuits," 2013. [Online]. Available: https://arxiv.org/abs/1306.3200

[2] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, p. 818–830, 2013.

[3] A. Paradis, J. Dekoninck, B. Bichsel, and M. Vechev, "Synthetiq: Fast and versatile quantum circuit synthesis," 2024.

[4] M. Mosca and P. Mukhopadhyay, "A polynomial time and space heuristic algorithm for t-count," *Quantum Science and Technology*, vol. 7, no. 1, p. 015003, 2021.

[5] M. Amy and M. Mosca, "T-count optimization and reed–muller codes," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4771–4784, 2019.

[6] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, "An algorithm for the t-count," 2013. [Online]. Available: https://arxiv.org/abs/1308.4134

[7] C. M. Dawson and M. A. Nielsen, "The solovay-kitaev algorithm," 2005. [Online]. Available: https://arxiv.org/abs/quant-ph/0505030

[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. Cambridge University Press, 12 2010.

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

[10] A. Mondal and K. K. Parhi, "Quantum circuits for stabilizer error correcting codes: A tutorial," *IEEE Circuits and Systems Magazine*, vol. 24, no. 1, p. 33–51, 2024. [Online]. Available: http://dx.doi.org/10.1109/MCAS.2024.3349668

[11] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, vol. 70, no. 5, p. 052328, 2004.

[12] K. N. Patel, I. L. Markov, and J. P. Hayes, "Optimal synthesis of linear reversible circuits," *Quantum Info. Comput.*, vol. 8, no. 3, p. 282–294, 2008.

[13] B. Giles and P. Selinger, "Exact synthesis of multiqubit clifford+t circuits," *Physical Review A*, vol. 87, no. 3, p. 032332, 2013.

[14] V. Kliuchnikov, "Synthesis of unitaries with clifford+t circuits," 2013. [Online]. Available: https://arxiv.org/abs/1306.3200

[15] V. Kliuchnikov, D. Maslov, and M. Mosca, "Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits," *Physical Review Letters*, vol. 110, no. 19, p. 190502, 2013.

[16] M. Weiden, J. Kalloor, E. Younis, J. Kubiatowicz, and C. Iancu, "High-precision multi-qubit clifford+t synthesis by unitary diagonalization," 2025. [Online]. Available: https://arxiv.org/abs/2409.00433

[17] D. Kremer, V. Villar, H. Paik, I. Duran, I. Faro, and J. Cruz-Benito, "Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2405.13196

[18] M. Bataille, "Quantum circuits of CNOT gates: optimization and entanglement," *Quantum Information Processing*, vol. 21, no. 7, p. 269, 2022.

[19] V. Gheorghiu, J. Huang, S. M. Li, M. Mosca, and P. Mukhopadhyay, "Reducing the cnot count for clifford+t circuits on nisq architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2022.

[20] G. Meuli, M. Soeken, and G. De Micheli, "Sat-based {CNOT, T} quantum circuit synthesis," in *International Workshop on Reversible Computation*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52138256

[21] I. Shaik and J. van de Pol, "Optimal layout-aware cnot circuit synthesis with qubit permutation," 2024. [Online]. Available: https://arxiv.org/abs/2408.04349

[22] C. Piazza, R. Romanello, and R. Wille, "An asp approach for the synthesis of cnot minimal quantum circuits," in *CEUR WORKSHOP PROCEEDINGS*, vol. 3428. CEUR-WS, 2023.

[23] C. Piazza and R. Romanello, "Synthesis of cnot minimal quantum circuits with topological constraints through asp," in *CEUR WORKSHOP PROCEEDINGS*, vol. 3586. CEUR-WS, 2023, pp. 37–49.

[24] M. Weiden, J. Kalloor, E. Younis, J. Kubiatowicz, and C. Iancu, "High precision fault-tolerant quantum circuit synthesis by diagonalization using reinforcement learning," 2024. [Online]. Available: https://arxiv.org/abs/2409.00433

[25] S. van der Linde, W. de Kok, T. Bontekoe, and S. Feld, "qgym: A gym for training and benchmarking rl-based quantum compilation," 2023. [Online]. Available: https://arxiv.org/abs/2308.02536

[26] S. Aaronson and Z. Weinersmith, "Constrained quantum cnot circuit re-synthesis using deep reinforcement learning," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:235210331

[27] D. Gottesman, "The heisenberg representation of quantum computers," 1998. [Online]. Available: https://arxiv.org/abs/quant-ph/9807006

[28] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, vol. 70, no. 5, 2004.

[29] M. Artin, *Algebra*. Pearson Education, 2011.

[30] A. Bu, E. Fan, and R. Sanghyeon Joo, "Minimum synthesis cost of cnot circuits," 2024. [Online]. Available: https://arxiv.org/abs/2408.07898

[31] T. G. De Brugière, M. Baboulin, B. Valiron, S. Martiel, and C. Allouche, "Gaussian elimination versus greedy methods for the synthesis of linear reversible circuits," *ACM Transactions on Quantum Computing*, vol. 2, no. 3, 2021.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of machine learning research*, vol. 22, no. 268, pp. 1–8, 2021.

[34] T. Goubault de Brugière, M. Baboulin, B. Valiron, S. Martiel, and C. Allouche, "Decoding techniques applied to the compilation of cnot circuits for nisq architectures," *Science of Computer Programming*, vol. 214, p. 102726, 2022.