Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

Optimal Clifford Synthesis as Planning

Irfansha Shaik[1,2], **Jaco van de Pol**[2]

[1]Kvantify Aps, Copenhagen, Denmark
[2]Aarhus University, Aarhus, Denmark
irsh@kvantify.com, jaco@cs.au.dk

Abstract

With the growing interest in practical quantum computing on noisy quantum hardware, quantum circuit optimization is becoming increasingly important. We consider the optimal synthesis of Clifford circuits, which are important for quantum error correction. While a scalable SAT encoding for Clifford synthesis exists, it only optimizes for binary quantum gates. Adding too many unary gates introduces unnecessary noise.

In this paper, we aim at synthesizing Clifford circuits with minimal binary gate count, and minimal unary gate count as a secondary optimization criterion. Expressing this in a performant SAT encoding is non-trivial. It turns out that planning specifications using PDDL with conditional effects are elegant and allow a flexible specification of costs. However, existing planners struggle on domains with conditional effects.

The contribution of this paper is a number of planning domains to solve the optimal binary/unary Clifford synthesis problem. The best performing solution combines the SAT approach for binary gates with a planning approach for unary gate minimization. With this hybrid approach, we present improved circuits for famous quantum error correcting codes, such as the Steane code. We believe that our planning domains with conditional effects and costs form a challenging benchmark for contemporary planners.

1 Introduction

Many important problems like drug discovery and material design require simulation of quantum systems, which is difficult for classical computers. The key idea behind quantum computing (Nielsen and Chuang 2010) is to emulate quantum-mechanical systems using hardware that exhibits quantum properties. The fundamental unit is called a qubit that, unlike classical bits, can be in a superposition of 0 and 1. Single qubit (1q) gates are used to manipulate the states of qubits and two qubit (2q) gates can create entanglement. Various quantum platforms exist, each having its own native gate set, connectivity constraints, and noise characteristics. The manipulation of qubits and interactions with the environment induce noise.

Quantum algorithms are expressed by quantum circuits, i.e., sequences of gates from some universal gate set. These need to be compiled to the native gate set of a specific platform. Quantum circuit optimization, minimizing gate count and circuit depth, is crucial to reduce noise. For example, Ettenhuber et al. (2025) reported that aggressive circuit optimization was essential to achieve quantum-chemistry calculations on the IonQ quantum platform.

While circuit optimization is crucial for practical computing, it is shown to be NP-hard (van de Wetering and Amy 2024). Major compilers like Qiskit (Qiskit contributors 2023) and Tket (Sivarajah et al. 2020) employ fast heuristics for optimization, however, the results are often far from optimal. The semantics of an n-qubit quantum circuit is captured by a $2^n \times 2^n$ complex matrix, which is impractical even for small n (Nagarajan, Lockwood, and Coffrin 2021). A popular alternative is peephole synthesis, where circuit fragments that allow a more compact representation are resynthesized.

In this paper, we focus on the important subclass of Clifford circuits, which are composed of the 1q-gates Hadamard (H) and phase (S) and the 2q-gate CNOT (conditional not, aka CX). Clifford circuits capture important features such as entanglement and superposition, and are an integral part of error correction circuitry. Interestingly, an n-qubit Clifford circuit can be efficiently represented by a stabilizer matrix of $2n \times 2n$ Booleans (Aaronson and Gottesman 2004).

The past few years showed significant progress in optimal Clifford synthesis. Since on many quantum platforms 2q gate fidelities are up to 10x lower than 1q gates, previous work has mainly focused on optimizing the CNOT gate count (cx-count) or the CNOT circuit depth (cx-depth). Bravyi, Latone, and Maslov (2022) propose normal forms that guarantee cx-count optimality. These normal forms help avoid arbitrary long 1q gate sequences. By employing a brute force search of 100 days, they were able to synthesize all 6-qubit Clifford circuits, resulting in 2.1 TB database. Shaik and van de Pol (2025) encoded the normal forms in a SAT encoding and optimized both cx-count and cx-depth, achieving cx-depth optimal synthesis for 7-qubit circuits.

While normal forms guarantee 2q-gate optimality, the scalability comes with a noise penalty caused by the introduction of too many 1q gates. Major compilers like Qiskit and Tket provide heuristic 1q gate reductions, but these may increase cx-count and cx-depth. To our knowledge, currently no approach exists that minimizes 1q gates while still guaranteeing 2-qubit gate optimality. Dedicated SAT based en-

codings scale relatively well for optimal 2q-gate synthesis, but designing a performant SAT encoding that also captures 1q-gate minimization is non-trivial. On the other hand, planning specifications in PDDL with conditional effects and costs allow for an easy, compact and flexible modelling.

Classical planning has been used in various quantum circuit optimization tasks, which can be typically encoded as graph reachability problems. Computing the shortest path corresponds to optimizing metrics such as gate count or circuit depth. Previously, Quantum Layout Synthesis, a related problem, has been solved using planning techniques. This involves scheduling given quantum gates on a quantum platform, reducing so-called SWAP gates (or depth). Several approaches were proposed based on genetic algorithms (Arufe et al. 2023), greedy search (Oddi and Rasconi 2018), and divide-and-conquer (Baioletti et al. 2025). Moreover, concise PDDL encodings were proposed as classical and temporal planning, optimizing for SWAP count (Shaik and van de Pol 2023) and circuit depth (Venturelli et al. 2017). For Circuit Synthesis, Shaik and van de Pol (2024) proposed planning specifications for pure CNOT synthesis with a single action using conditional effects. A recent paper (Pozo and Seipp 2025) used the CNOT synthesis domain as a challenging benchmark for merge and shrink heuristics. Planning domains for Clifford synthesis have not been proposed before.

The contribution of this paper is two-fold: (1) We propose concise planning domains for optimal Clifford synthesis, and compare their performance with an existing encoding in SAT. Our planning domains are compact due to the use of conditional effects and cost-specifications. (2) We exploit the flexibility of planning domains to complement cx-count optimality by also using the fewest 1q gates possible.

We deliver three planning domains: The `normalform` domain with conditional effects is based on Clifford normal forms to guarantee cx-count optimality. This domain closely resembles the previous encoding in SAT. Next, we present a `costbased` domain, which uses cost specifications to yield cx-count optimal circuits that additionally use the minimal number of 1q gates. We noticed that synthesizing plans using this domain is quite slow. Our third approach integrates SAT and planning: We use the SAT encoding to obtain a CNOT-optimal circuit, and provide the `rigid` planning domain to synthesize 1q gates within the given CNOT structure. Note that this approach may miss a smaller solution with a different CNOT structure.

In our experiments, we observe that (both SAT-based and other) planners struggle on domains with conditional effects. The existing dedicated SAT encoding significantly outperforms planners using the `normalform` domain. Using the `costbased` domain, we managed to reduce the 1q-gate count for 3 and 4 qubit circuits, while still guaranteeing cx-count optimality. With our `rigid` domain we were able to optimize the 1q-gate count even for quantum circuits on 5 qubits. Using our hybrid SAT + Planning approach, we found improved circuits for some quantum error correcting codes, such as the 7-qubit Steane code. All the domains and code are available in version 6 of our open source tool.[1]

[1]Q-Synth is available at https://github.com/irfansha/Q-Synth

2 Preliminaries

2.1 Quantum Gates

We provide an informal overview of quantum states and quantum gates.[2] The basic qubit states are called $|0\rangle$ and $|1\rangle$. An alternative basis is provided by their superpositions $|+\rangle = |0\rangle + |1\rangle$ and $|-\rangle = |0\rangle - |1\rangle$. We write I for the identity gate. Two basic 1q gates are X (bit flip) and Z (phase flip). X exchanges $|0\rangle$ and $|1\rangle$, while Z exchanges $|+\rangle$ and $|-\rangle$. These so-called Pauli gates can also be viewed as stabilizers of certain states, i.e. $\mathrm{Z}|0\rangle = |0\rangle$ and $\mathrm{X}|+\rangle = |+\rangle$. The 1q Hadamard gate is defined by $\mathrm{H}|0\rangle = |+\rangle$ and $\mathrm{H}|1\rangle = |-\rangle$. The gates above are involutions, i.e. $\mathrm{X}^2 = \mathrm{Z}^2 = \mathrm{H}^2 = \mathrm{I}$. H exchanges the X and Z bases in the following sense: $\mathrm{HXH} = \mathrm{Z}$ and $\mathrm{HZH} = \mathrm{X}$. Viewing X and Z as rotations in the complex plane, we can also apply them halfways. This yields the quantum gates $\mathrm{Sx} = \sqrt{\mathrm{X}}$ and $\mathrm{S} = \sqrt{\mathrm{Z}}$. We have $\mathrm{H}\sqrt{\mathrm{X}}\mathrm{H} = \sqrt{\mathrm{Z}}$ and $\mathrm{H}\sqrt{Z}\mathrm{H} = \sqrt{\mathrm{X}}$. Clearly, all gates mentioned so far can be built from the 1q Clifford gates H and S. To complete the Clifford gate set we need the binary CNOT gate (Conditional NOT or CX), which acts on the basis states as $\textsc{Cnot}(a, b) = (a, a \oplus b)$. Here a is the control qubit (unchanged) and b is the target qubit (flipped when $a = |1\rangle$). A 1q gate that is outside the Clifford set is $T = \sqrt{S}$. Clifford + T is a standard universal gate set.

2.2 Stabilizer Formalism

An n-qubit quantum circuit is a sequence of gates, applied to a fixed set of n qubits. In general, it can be represented by $2^n \times 2^n$ complex matrix. Clifford circuits (built from $\{\mathrm{H}, \mathrm{S}, \textsc{Cnot}\}$-gates) on the other hand can be compactly represented using the so-called stabilizer formalism (Aaronson and Gottesman 2004) using only $2n \times 2n$ Boolean matrices.[3] The following matrix illustrates an n-qubit tableau. Each row i corresponds to a stabilizer generator. Each stabilizer applies one of the operators in $\{\mathrm{I}, \mathrm{X}, \mathrm{Z}, \mathrm{XZ}\}$ to each qubit j, aptly encoded by the two bits $x_{ij}z_{ij}$ in the tableau.

$$\left(\begin{array}{ccc|ccc} q_1 & \cdots & q_n & q_1 & \cdots & q_n \\ x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{(2n)1} & \cdots & x_{(2n)n} & z_{(2n)1} & \cdots & z_{(2n)n} \end{array}\right)$$

[2]A formal description of n-qubit systems involves unitary operations over vector spaces $\mathbb{C}^{2^n}$ (Nielsen and Chuang 2010).

[3]The full tableau (Aaronson and Gottesman 2004) has an extra r-column for the local phase, but this can be recovered in post-processing (Shaik and van de Pol 2025), so we can ignore it.

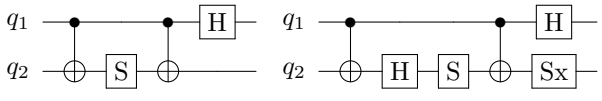

Figure 1: Example circuit (left) and its normal form (right)

$$\begin{array}{cc|cc} q_1 & q_2 & q_1 & q_2 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{cc|cc} q_1 & q_2 & q_1 & q_2 \\ \hline 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}$$

Figure 2: Identity matrix on 2 qubits (left) and the final tableau for the circuit in Figure 1 (right)

The tableau of an n-qubit Clifford circuit of m gates can be computed in $O(n^2m)$ time, by applying the tableau update rule corresponding to each gate. The initial state (the empty circuit) is simply the $2n \times 2n$ identity matrix.

We now explain the tableau update rules in detail. The gate $\textsc{Cnot}(a, b)$ is applied by adding the ath X column to the bth, and the bth Z column to the ath, i.e., for all i, we update x_{ib} to $x_{ia} \oplus x_{ib}$ and z_{ia} to $z_{ia} \oplus z_{ib}$. The tableau update rules for the 1-qubit Clifford gates are defined in Table 1.

	Base Gates		Composite gates		
	H_a	S_a	HS_a	SH_a	Sx_a
x_{ia} :	z_{ia}	x_{ia}	z_{ia}	$x_{ia} \oplus z_{ia}$	$x_{ia} \oplus z_{ia}$
z_{ia} :	x_{ia}	$x_{ia} \oplus z_{ia}$	$x_{ia} \oplus z_{ia}$	x_{ia}	z_{ia}

Table 1: Tableau update rules for the Clifford gates H and S on qubit a (on each row i). H_a swaps columns $\mathrm{X}_a/\mathrm{Z}_a$ and S_a adds column X_a to Z_a. We also show gate combinations.

Optimal Clifford synthesis can now be elegantly encoded as a graph reachability problem: Nodes are labeled by tableaux and edges correspond to Clifford gate updates. Finding the shortest path from the initial tableau to the final tableau corresponds to the Clifford circuit with optimal gate count. Figure 2 shows an example initial and goal tableau.

2.3 Normal Forms for 2-qubit Gate Optimization

In practice, CNOT gates can be up to 10x noisier than 1-qubit gates, thus optimizing the number of CNOT gates (cx-count) is prioritized over 1q gates. Bravyi, Latone, and Maslov (2022) proposed so-called normal forms for Clifford circuits that guarantee cx-count optimality. They present two key observations: 1) An arbitrary sequence of 1q Clifford gates can be rewritten to one of the *unique 1q-gates* {I, H, S, HS, SH, Sx}. 2) Consider a CNOT gate preceded by arbitrary sequences of 1q Clifford gates on its input qubits. This combination can always be rewritten to one of the 9 *entangling gates* in Figure 3, followed by some trailing 1q-gates.

Combining these two observations, any n-qubit Clifford circuit with k CNOT gates can be rewritten into a *normal form* with k entangling gates followed by a single layer of unique 1q-gates. For example, the circuit in Figure 1 (left) can be rewritten to its normal form (right). Here the S gate is replaced by HS, pushing an Sx gate to the right. The result is a normal form with 2 entangling gates followed by 1q gates.

Synthesis of normal forms presents two advantages: First, significant search space reduction by avoiding arbitrary sequences of 1q-gates. Moreover, only 9 entangling gates need to be considered for each CNOT gate, instead of $6{*}6{*}2 = 72$ (6 unique 1q-gates per control/target qubit and 2 CNOT orientations). Second, optimizing cx-count can be encoded as modified graph reachability, by applying an entangling gate at each step, and a layer of unique 1q-gates at the end.

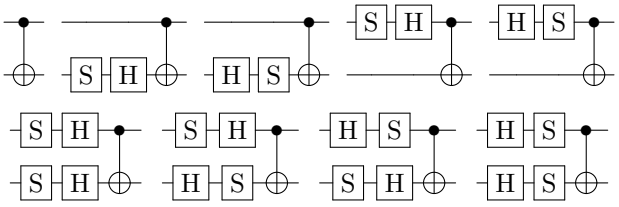

Figure 3: The nine 2-qubit gate Clifford normal forms from (Bravyi, Latone, and Maslov 2022)

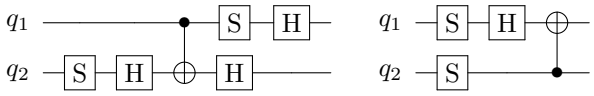

Figure 4: CNOT-optimal circuit in normal form (left) from (Shaik and van de Pol 2025) and CNOT + 1q-gate optimal circuit, not in normal form (right) as obtained in this work.

Shaik and van de Pol (2025) take advantage of these observations to employ a scalable SAT encoding that optimizes cx-count or cx-depth. Figure 4 shows the cx-count optimal circuit in normal form (left). However, it does not guarantee 1q gate optimality, thus contributing to unnecessary noise. Only by going beyond normal forms, we can further reduce the 1q-gate count as shown in Figure 4 (right). Notice that, in this example, the gate sequence is not available through normal forms. In the next section, we present planning domains for normal forms and beyond in order to minimize 1q-gates.

3 Clifford Synthesis as Planning

Reachability can be elegantly encoded as a planning problem. In this section, we present three domains for Clifford synthesis as planning.

Encoding Tableaux. For any domain, Clifford tableaux are encoded in the same way. As shown in Algorithm 1, we define types `row` and `qubit` (to label the columns). We use two predicates to represent X and Z matrix elements.

Given an n-qubit circuit, we have $2n$ rows and n qubits. A problem instance will have $2n$ objects of type `row` and n objects of type `qubit`. The initial state encodes the initial tableau (identity matrix) and the goal state encodes the target tableau. Consider the example circuits in Figure 1. Al-

Algorithm 1: Types and predicates for domains

```
(:types row qubit)
(:predicates (X ?r - row ?c - qubit)
             (Z ?r - row ?c - qubit))
```

Algorithm 2: Example snippet for tableau problem encoding

```
(:objects q1 q2 - qubit
          r1 r2 r3 r4 - row)
(:init (X r1 q1)(X r2 q2)(Z r3 q1)(Z r4 q2))
(:goal (and (X r1 q1)  (not(X r1 q2))
            (X r2 q1)      (X r2 q2)
            (X r3 q1)  (not(X r3 q2))
       (not(X r4 q1)) (not(X r4 q2))
            (Z r1 q1)      (Z r1 q2)
       (not(Z r2 q1))      (Z r2 q2)
       (not(Z r3 q1)) (not(Z r3 q2))
       (not(Z r4 q1))      (Z r4 q2)))
```

gorithm 2 shows the snippet of the problem encoding for the corresponding tableau. We see that $q1, q2$ are the two qubits and $r1, r2, r3, r4$ are the four rows. The initial state encodes the identity matrix as in Figure 2 (left) and the goal state encodes the final tableau shown in Figure 2 (right).

3.1 Normal Form based Synthesis as Planning

Normal forms for Clifford synthesis guarantee cx-count optimality as discussed in Section 2.3. Recall that any Clifford circuit with k CNOTs has an equivalent circuit with k CNOTs in normal form. Further, normal forms also provide an upper bound on the number of 1q gates. For applying any one of the 9 entangling gates (as in Figure 3), at most 4 1q gates are needed (2 on control and 2 on target). For the last layer, at most 2 one-qubit gates are needed per qubit. Thus, for an n-qubit Clifford circuit with k CNOT gates, the normal form uses at most $4k + 2n$ one-qubit gates.

In this section, we encode the normal form domain in two parts. First, we define four actions that capture the 9 entangling gates. Then we define one action per unique 1q sequence (6 in total) in the last layer.

Encoding Entangling Gates. To apply any of the entangling gates on control and target qubits, we define four actions. The 9 combinations of 1q sequences can be captured by applying {I, HS, SH} gates on control and target qubits. We define one action for each of {I, HS, SH} 1q sequences. A CNOT gate is applied only after applying these 1q gates on control and target qubits. For this, we use the `(applied_1q ?q)` predicate to track whether a 1q gate has been applied on qubit `?q`. Further, we use the `(disabled ?q)` predicate to avoid applying any gate after the last layer of 1q gates. For any 1q action, the precondition ensures that no 1q gate has been applied on the chosen qubit. The effect marks that 1q gates have been applied on chosen qubit, and applies the tableau updates for the corresponding gate as shown in Table 1. In the interest of space, we only show the action for the HS gate in Algorithm 3. Actions for I and SH can be added in a similar way.

After applying 1q actions on control and target qubits, a CNOT gate as shown in Algorithm 4 is applied to complete an entangling gate. The precondition ensures that 1q gates have been applied on control and target qubits. The effect applies the tableau updates (lines 12 to 24) as discussed in Section 2.2. It also resets the `applied_1q` predicates.

Algorithm 3: Actions for HS gate in entangling gates

```
(:action hs-gate :parameters (?a - qubit)
 :precondition (and
   (not(disabled ?a))(not(applied_1q ?a)))
 :effect (and (applied_1q ?a)
   (forall(?r - row) (when ;; x_{r,a} := z_{r,a}
       (and (Z ?r ?a))
       (X ?r ?a)))
   (forall(?r - row) (when
       (and(not(Z ?r ?a)))
       (not(X ?r ?a))))
   (forall(?r - row) (when ;; z_{r,a} := x_{r,a} ⊕ z_{r,a}
       (and (Z ?r ?a)(X ?r ?a))
       (not(Z ?r ?a))))
   (forall(?r - row) (when
       (and(not(Z ?r ?a))(X ?r ?a))
       (Z ?r ?a)))))
```

Algorithm 4: CNOT action

```
(:action cnot :parameters (?a ?b - qubit)
 :precondition (and
   (connected ?a ?b)
   (not(disabled ?a))(not(disabled ?b))
   (applied_1q ?a)(applied_1q ?b))
 :effect (and (not(applied_1q ?a))
   (not(applied_1q ?b))
   (forall(?r - row) (when  ;; x_{r,b} := x_{r,a} ⊕ x_{r,b}
       (and (X ?r ?a)(X ?r ?b))
       (not (X ?r ?b))))
   (forall(?r - row) (when
       (and (X ?r ?a)(not (X ?r ?b)))
       (X ?r ?b)))
   (forall(?r - row) (when  ;; z_{r,a} := z_{r,a} ⊕ z_{r,b}
       (and (Z ?r ?a)(Z ?r ?b))
       (not (Z ?r ?a))))
   (forall(?r - row) (when
       (and (not (Z ?r ?a))(Z ?r ?b))
       (Z ?r ?a)))))
```

Most quantum platforms have limited qubit connectivity, thus CNOT gates can only be applied on adjacent or connected physical qubits. Given such connections as a coupling graph, layout restrictions can be applied on CNOT gates by defining a `connected` predicate for every connected qubit pair. For all-to-all connectivity, every qubit pair is connected. The precondition ensures that a CNOT gate is only applied on a connected qubit pair. For normal forms, CNOTs are always ordered. Thus we add `(connected ?a ?b)` where ?a is less than ?b.

Encoding Layer of Single Qubit Gates. After applying all the entangling gates, for each qubit we need to apply one of the six 1q sequences {I, H, S, HS, SH, Sx}. Here we present only one of the actions. Note that the tableau updates for the 1q gates are the same both in entangling gate or in the last layer. In the precondition, we ensure that no 1q gate has been applied thus the previous entangling gate is complete. Further, no qubit has been disabled on the chosen qubit, so the closing layer has not yet been applied. The effect first

Algorithm 5: Action for Sx gate in last layer

```
(:action sx-last :parameters (?a - qubit)
 :precondition (and
   (not(disabled ?a))(not(applied_1q ?a)))
 :effect (and (disabled ?a)
   (forall(?r - row) (when  ;;  x_{r,a} = x_{r,a} ⊕ z_{r,a}
       (and (X ?r ?a)(Z ?r ?a))
       (not (X ?r ?a))))
   (forall(?r - row) (when
       (and (not (X ?r ?a))(Z ?r ?a))
       (X ?r ?a)))))
```

marks that the last layer 1q gate has been applied by disabling the chosen qubit. Then, it applies the tableau updates for the corresponding gate as shown in Table 1. Algorithm 5 shows the action for the Sx-gate in the last layer.

In the goal, to ensure that a planner applies all the last layer 1q gates, we also add `(disabled ?q)` predicates for every qubit `?q` in the goal state. Given an n-qubit instance with k optimal CNOT gates, any optimal plan will have exactly $3*k+n$ actions. Any optimal plan corresponds to a Clifford circuit with optimal number of CNOT gates.

3.2 Cost-based Synthesis Beyond Normal Forms

While the `normalform` domain guarantees CNOT optimality, it does not guarantee 1q-gate optimality. In our running example, we showed that by going beyond normal forms, we can further reduce 1q-gate count (Figure 4, right). In this section, we discuss how to relax normal form restrictions and add cost specifications to guarantee optimality.

Relaxing Normal Form Restrictions. To also consider one qubit gate count, we need to relax normal form restrictions. There are two main restrictions in normal forms:

- Only {I, HS, SH} sequences are allowed before a CNOT.
- CNOT(a,b) gates have a fixed orientation, i.e., $a < b$.

We now allow all 1q-sequences {H, S, HS, SH, Sx} before a CNOT gate (we drop the I-gate since it corresponds to no-action). Note that we still do not allow multiple one qubit sequences as in the `normalforms` domain, to bound the search space. Furthermore, we allow CNOT gates in both orientations by extending the `connected` predicate.

Using Costs to Guarantee Optimality. First to guarantee CNOT gate optimality, choosing a CNOT gate should be costlier than choosing the maximum number of 1q gates in the optimal circuit. We use the insights from normal forms to compute these costs. Before synthesis, we do not know the optimal CNOT count. However, we can take the CNOT count of the input circuit as an upper bound k. Thus, the maximum number of 1q gates in the optimal Clifford circuit is $4k + 2n$ as discussed in Section 3.1. So, we give a cost of $4k + 2n + 1$ to every CNOT action. For 1q gates, we give a cost of 1 to the {H, S, Sx} actions and a cost of 2 to the {HS, SH} actions (since they correspond to applying two quantum gates). With these costs, any optimal plan will first minimize CNOT count and then minimize one qubit gate count.

Algorithm 6: Action for S gate in costbased domain

```
(:action s-gate :parameters (?a - qubit)
 :precondition (and (not(applied_1q ?a)))
 :effect (and (applied_1q ?a)
   (forall(?r - row) (when ;;  z_{r,a} = z_{r,a} ⊕ x_{r,a}
       (and (Z ?r ?a)(X ?r ?a))
       (not (Z ?r ?a))))
   (forall(?r - row) (when
       (and (not(Z ?r ?a))(X ?r ?a))
       (Z ?r ?a)))
   (increase (total-cost) 1)))
```

Algorithm 7: CNOT action snippet in `costbased` domain

```
(:action cnot :parameters (?a ?b - qubit)
 :precondition (and (connected ?a ?b))
 :effect (and ...
  (increase (total-cost) (cnot-cost))))
```

Better Costs using Updated Normal Forms. In the normal form, we specified that at most 4 one qubit gates are needed per CNOT gate. However, we observed that in an alternative normal form, only 2 1q gates are needed instead of 4. Essentially, we only need to apply gates from {I, H, Sx} and {I, H, S} on the control and target qubits, respectively, for rewriting any of the 9 entangling gates. Using this, we can further reduce the CNOT action cost to $2(k + n) + 1$.

We present action snippets specified for the `costbased` domain. The precondition for the S-gate (Algorithm 6) ensures that no 1q gate has been applied on the chosen qubit. The effect applies the tableau updates and increases the total cost (standard cost function) by 1. The tableau updates are the same as in the `normalform` domain.

Next, we present a snippet of the CNOT action in Algorithm 7. The precondition ensures that control and target qubits are connected. To allow for the I-gate, we do not check for `applied_1q` predicates. The effect applies the tableau updates and increases the total cost by $2(k + n) + 1$. The tableau updates are the same as in the `normalform` domain (as in Algorithm 4). To update the cost we use a cost function `(cnot-cost)` defined in the problem instance.

Note that by simply changing CNOT cost, we can change the cost metric. For example, setting `(cnot-cost)` to 1, we optimize the total number of gates in the Clifford circuit. Similarly, one can also adjust costs for 1q gates to optimize hardware specific metrics. Here, we focus on optimizing 1q gate count while guaranteeing CNOT optimality.

3.3 Stand-alone Single Qubit Gate Resynthesis

Layout-aware optimal Clifford synthesis is proven to be NP-hard (Jiang et al. 2020). In fact, even approximate layout aware synthesis is NP-hard (Iwama, Kambayashi, and Yamashita 2002). For all-to-all connectivity, the hardness is still an open problem. As seen in the experiments (Section 4), planners struggle with finding optimal plans even for small benchmarks with 4 qubits, while an existing dedicated SAT encoding works well when optimizing cx-count/depth even for 5 and 6 qubits (Shaik and van de Pol 2025). How-

Algorithm 8: Additional predicates for `rigid` domain

```
(:types row qubit gate - object)
(:predicates (done ?g - gate)
  (oneq_gate ?g - gate ?a - qubit)
  (cnot_gate ?g - gate ?a ?b - qubit)
  (depends ?prev_g ?cur_g - gate))
```

Algorithm 9: CNOT action snippet in `rigid` domain

```
(:action cnot :parameters
   (?a ?b - qubit ?prev_g ?cur_g - gate)
 :precondition (and
   (cnot_gate ?cur_g ?a ?b)
   (depends ?cur_g ?prev_g)
   (done ?prev_g) (not (done ?cur_g)))
 :effect (and (done ?cur_g)
   ... tableau updates for CNOT ...
   (increase (total-cost) 1)))
```

ever, it introduces many 1q gates, which is undesirable.

To this end, we propose the `rigid` domain for 1q gate resynthesis as a post-processing step. From our `costbased` domain, we know that at most two 1q sequences can be applied before a CNOT gate (on the control and target qubits). Also, a last layer of 1q gates might be needed. The key idea is to fix the positions of the CNOT gates from the input circuit. The planner then needs to choose the 1q gates to apply before the CNOT gates and in the last layer. This significantly reduces the search space for the planner. We use the additional predicates listed in Algorithm 8 to fix the CNOT and 1q-gate positions. The predicates `(oneq_gate ?g ?a)` and `(cnot_gate ?g ?a ?b)` define the positions of 1q gates and CNOT gates, respectively. The predicate `(done ?g)` tracks whether a gate `?g` has been applied. We employ a total ordering based on the input circuit to avoid partial orders in choosing the 1q gates. For any CNOT, 1q gates are applied first on control and then on target qubit before applying the CNOT. Let c1 and c2 be two CNOTs in the input circuit, if c1 appears before c2 in the input circuit, then c1 is also scheduled before c2. In the last layer, 1q gates are applied in the order of qubit indices. To enforce ordering, we use the predicate `(depends ?prev_g ?cur_g)` which is precomputed based on the input circuit. In Algorithm 9, we show snippets of CNOT action in the `rigid` domain.

For the CNOT action, the precondition ensures that the previous gate `?prev_g` has been applied and the current gate `?cur_g` has not been applied yet. The predicate `(cnot_gate ?cur_g ?a ?b)` ensures that the current gate is a CNOT gate on qubits `?a` and `?b`. The predicate `(depends ?cur_g ?prev_g)` ensures that the ordering is maintained. The effect applies the tableau updates for the corresponding gate and marks the current gate `?cur_g` as done. Since the number of CNOT gates are fixed, we set their cost to 1. For 1q gates, we set similar costs as in the `costbased` domain. The only additional action we have here is `i-gate` for the identity gate, which has zero cost. In the goal, we ensure that all gates are marked as done. Interestingly, since the CNOT structure is fixed, neither the cx-count nor cx-depth is changed. Most preprocessing techniques in major compilers (like Qiskit, Tket) do not provide control over optimization metrics. Our 1q gate resynthesis can be easily integrated with existing compilers to further reduce 1q count without changing CNOT metrics.

4 Experiments and Results

We propose three experiments to address [4]:

- **RQ1:** How do planning approaches compare to existing SAT encodings for cx-count optimization?
- **RQ2:** Can planning approaches improve upon 1q-count compared to SAT encodings without changing cx-count?
- **RQ3:** Can SAT and planning approaches provide meaningful results for practically relevant Clifford circuits?

4.1 E1: CNOT Optimization via Normal-forms

To answer RQ1, we compare our `normalform` domain with a SAT encoding for cx-count optimization in Q-Synth (Shaik and van de Pol 2025). We use the random Clifford circuits on 3-5 qubits from that paper. For a comparison, we consider four different planners that handle conditional effects: Lama (Richter and Westphal 2010), Madagascar (Rintanen 2014), Scorpion (Pozo and Seipp 2025) and Symk (Speck and Helmert 2025). As baselines, we consider Lama as available in FastDownward (Helmert, Röger, and Karpas 2011), and Madagascar, a state-of-the-art SAT based planner. While planners struggle with conditional effects, there has been recent progress. Pozo and Seipp (2025) present abstraction heuristics that can handle conditional effects, implemented in Scorpion planner. They considered the pure CNOT synthesis planning domain (Shaik and van de Pol 2024) with conditional effects as a benchmark. The CNOT synthesis planning domain has a single CNOT action similar to ours with conditional effects for (simpler) tableau updates. Thus, we consider the best performing configuration on the CNOT synthesis domain, Saturated Cost Partitioning over Merge&Shrink heuristics, available in Scorpion planner. We also consider Symk, a symbolic planner, that effectively handles conditional effects. For Q-Synth, we use the default recommended options. For each circuit, we measure time, cx- and 1q-count, given 1800s time and 8GB memory.

Results and Discussion. Table 3 shows the results for Experiment 1. Clearly, existing SAT based approaches significantly outperform planners with the `normalform` domain. All planners can optimally solve 3-qubit instances but struggle with 4 and 5-qubit instances. We expected that Madagascar would perform well since the domain formulation is close to the dedicated SAT encoding in Q-Synth. However, Madagascar could not solve any 4 or 5-qubit instances. Scorpion and Symk could optimally solve one 4-qubit circuit each. Scorpion uses all the memory for other instances whereas Symk results in time outs. Interestingly, Lama finds optimal solutions for all 4-qubit instances, but it does not

[4]Full code, data and logs are available at https://doi.org/10.5281/zenodo.18678724.

guarantee optimality. For 5-qubit instances, Lama provides some solution within timelimit (often memory outs), but far from optimal. For cx-count optimization, currently dedicated SAT encodings are the best performing approaches. While constructing the `normalform` domain is very easy and elegant, improvements in planners for conditional effects are clearly needed to be useful in practice.

4.2 E2: 1q Optimization Beyond Normal-forms

To answer RQ2, we compare our `costbased` and `rigid` domains with Q-Synth for 1q-count optimization without increasing the cx-count. The `costbased` domain optimizes both cx and 1q-counts while the `rigid` domain optimizes only 1q-count. Thus, for `rigid`, we use cx-optimized circuits from Q-Synth as input circuits. We use the same benchmarks and setup as E1 in Section 4.1, removing the Madagascar planner since it does not support costs.

Results and Discussion. Table 4 shows the results for Experiment 2. Similar to E1, planners struggle with 4 and 5-qubit instances even for the `costbased` domain. However, for 3-qubit instances, planners provide significant reductions in 1q-count while also guaranteeing cx-count optimality. For example, in 03q_99346, all three planners reduce 1q-count from 17 to 3 while maintaining a cx-count of 4. Again, Lama provides useful reductions even for 4-qubit instances. The results from our `rigid` domain are even more interesting. Since the CNOT structure is fixed, planners can optimize the 1q-count even for 4-qubit instances. All three planners provide optimal solutions for 3 and 4-qubit instances within the time limit. Even for 5-qubit instances, Lama provides reductions in 1q-count for all instances. As expected, restricting the structure of CNOTs in rigid domain comes at the cost of increased 1q-count. For example, in 03q_33936, `costbased` domain has only 3 1q gates while the `rigid` domain results in 9 1q gates.

4.3 E3: Better Circuits for Steane and Shor Codes

From E2, we observed that planners can reduce 1q count without changing cx-count for small random Clifford circuits. The next natural question is whether these approaches can provide meaningful results for relevant Clifford circuits. To answer RQ3, we consider well-known quantum error correcting codes (QECC) like Steane (Steane 1999) and Shor codes (Shor 1995). For Steane code, the best known circuits are achieved by manually applying equivalences (Mondal and Parhi 2024). While, existing SAT encodings can further optimize cx-count/depth metrics, they introduce many 1q gates. Our `costbased` domain is an option, however we observed in E2 that it does not scale well beyond 4 qubits. Thus, we investigate if SAT approaches together with the `rigid` domain can provide better QECC circuits.

In addition to the standard 7-qubit Steane code, we also consider the 11-qubit circuit implementing Shor code with phase flip detection (IBM 2024) as benchmarks. First, we use the SAT encoding from Q-Synth to optimize with two combination metrics: (a) cx-cd: cx-count + cx-depth, and (b) cx-dc: cx-depth + cx-count. For combination metrics, Q-Synth first computes the optimal value for the first metric and then optimizes the second metric without changing the first metric. Per combination and circuit we give 3 hours and 8 GB memory, resulting in four optimized circuits. For each of them, we apply our `rigid` domain to further reduce 1q gate count without changing the CNOT structure. We use all three planners similar to E2, but now with 3 hours and 80 GB memory. The following table shows the metrics for relevant optimization where Q is Q-Synth, Q+R is Q-Synth + the `rigid` domain, and MP2024 are the best known results from Mondal and Parhi (2024) for the Steane code.

	Shor Code (11q)			Steane Code (7q)		
Setting	cx-c	cx-d	1q	cx-c	cx-d	1q
original	20	14	7	11	5	3
MP2024	-	-	-	10	7	**3**
Q (cx-cd)	**11**	5	10	**9**	8*	20
Q+R (cx-cd)	**11**	5	**3**	**9**	8*	**3**
Q (cx-dc)	14	**4**	35	10	**4**	25
Q+R (cx-dc)	14	**4**	**5**	10	**4**	**3**

Table 2: E3: Optimization of Shor and Steane code circuits.

Q-Synth was able to compute optimal circuits for 3 combinations out of 4. For cx-cd, Q-Synth could provide a cx-count optimal circuit for the Steane code, but only with a near-optimal cx-depth (marked *). Clearly, the SAT encodings add many 1q gates for all variants. Our `rigid` domain was able to significantly reduce the 1q-count for all four benchmarks. Regarding various planners, Lama did not produce any solution within the time limit. Scorpion was able to solve 3 out of 4 benchmarks optimally, while one instance ran out of memory even with 80 GB. Symk, on the other hand, could solve all benchmarks optimally within 1 hour and 5 GB memory each. Since our optimal 1q-counts are at most 5, we conjecture that having a low makespan significantly helps Symk. Although the tableau size increases quadratically with the qubit count, Symk (being a symbolic planner) does not run into memory issues for more qubits.

5 Conclusion and Future Work

We presented the first approach to synthesize Clifford circuits with the minimal number of CNOT gates, and additionally a minimal number of 1-qubit gates (using $\{\mathrm{H}, \mathrm{S}, \sqrt{\mathrm{X}}\}$). This enables optimal circuits for interesting quantum error correcting codes. We demonstrated the flexibility and ease of PDDL planning domains for quantum circuit optimization with complex cost specifications.

Interestingly, for the Steane code, we improve upon the best known results from Mondal and Parhi (2024) in both cx-count/cx-depth metrics. With cx-cd optimization, we reduce cx-count from 10 to 9 (provably optimal). With cx-dc optimization, we reduce cx-depth from 7 to 4 (optimal) with same cx-count (10) and 1q-count (3) as shown in Figure 5.

Interesting future work is to close the performance gap between SAT and planning encodings. The use of conditional effects seems to block efficient tactics applied by performant planners. Progress along the lines of Pozo and Seipp (2025) and Speck and Helmert (2025) would be welcome.

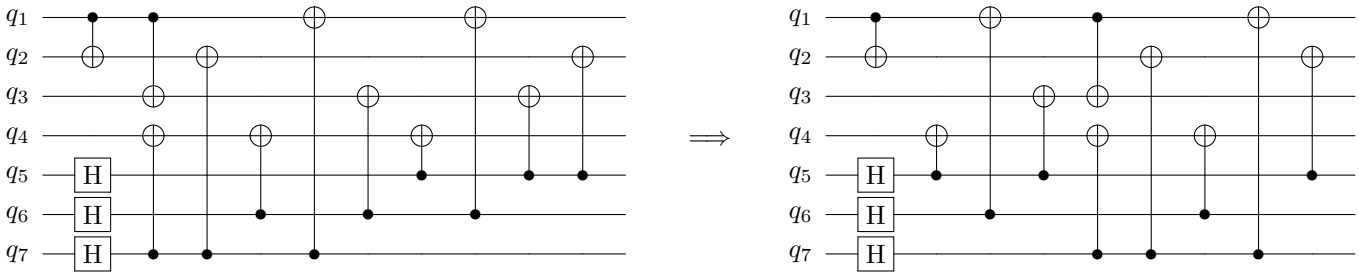

Figure 5: On the left: original unoptimized 7-qubit Steane code (cx-depth 5, cx-count 11). On the right: cx-depth+cx-count *provably optimal* Steane code (cx-depth 4, cx-count 10) with optimal `rigid` domain post-processing (back to 3 1q-gates).

	Original		Q-Synth			Lama			Madagascar			Scorpion			Symk		
benchmarks	cx	1q	cx	1q	t	cx	1q	t	cx	1q	t	cx	1q	t	cx	1q	t
03q_05306	4	14	4	14	0.2	4	18	156.6	4	18	36.4	4	16	1.1	4	16	3.8
03q_33936	3	14	3	14	0.2	3	16	16.1	3	16	4.5	3	12	0.3	3	11	2.6
03q_50494	4	18	4	18	0.2	4	17	155.1	4	17	39.0	4	15	1.1	4	14	3.4
03q_55125	3	20	3	20	0.2	3	14	15.9	3	14	2.0	3	14	0.3	3	14	2.6
03q_99346	4	17	4	17	0.2	4	15	156.1	4	16	39.4	4	17	1.1	4	12	3.4
04q_05306	6	23	6	23	0.5	6	23	MO	-	-	TO	-	-	MO	-	-	TO
04q_33936	8	13	**6**	22	0.7	**6**	21	MO	-	-	TO	-	-	MO	-	-	TO
04q_50494	7	20	7	20	0.8	7	25	MO	-	-	TO	-	-	MO	-	-	TO
04q_55125	6	10	6	10	0.5	6	26	MO	-	-	TO	-	-	MO	-	-	TO
04q_99346	7	16	**5**	21	0.5	**5**	21	MO	-	-	TO	**5**	21	382.3	**5**	21	506.5
05q_05306	12	31	**8**	33	4.0	16	45	MO	-	-	TO	-	-	MO	-	-	TO
05q_33936	13	22	**9**	37	13.8	15	48	MO	-	-	TO	-	-	MO	-	-	TO
05q_50494	17	28	**8**	34	2.6	19	50	MO	-	-	TO	-	-	MO	-	-	TO
05q_55125	13	16	**9**	35	9.3	14	50	MO	-	-	TO	-	-	MO	-	-	TO
05q_99346	11	26	**7**	25	1.7	9	33	MO	-	-	TO	-	-	MO	-	-	TO

Table 3: E1: cx-count optimization via normal forms, Q-Synth (SAT) vs planning; cx: CNOT count, 1q: 1q count, t: time in seconds; MO: memory out, TO: time out; Every instance without TO/MO is solved optimally; Best results in bold.

			`costbased` (cx+1q-count opt)									`rigid` (1q-count opt)					
	Q-Synth		Scorpion			Symk			Lama			Scorpion		Symk		Lama	
benchmarks	cx	1q	cx	1q	t	cx	1q	t	cx	1q	t	1q	t	1q	t	1q	t
03q_05306	4	14	4	**6**	9	4	**6**	6.6	4	**6**	235	**6**	0.7	**6**	3.8	**6**	0.8
03q_33936	3	14	3	**4**	0.7	3	**4**	3.6	3	**4**	14.1	6	0.4	6	2.4	6	0.4
03q_50494	4	18	4	**5**	8	4	**5**	6.2	4	**5**	165	6	0.6	6	3.2	6	2.5
03q_55125	3	20	3	**6**	1	3	**6**	3.6	3	**6**	43.7	**6**	0.5	**6**	2.4	**6**	0.5
03q_99346	4	17	4	**3**	8	4	**3**	6.1	4	**3**	126	9	0.6	9	8	9	1.1
04q_05306	6	23	-	-	MO	-	-	TO	6	**9**	MO	17	134	17	324	17	42.1
04q_33936	6	22	-	-	MO	-	-	TO	6	**7**	MO	12	23.8	12	104	12	106
04q_50494	7	20	-	-	MO	-	-	TO	7	12	MO	8	4.2	8	9.9	8	8.6
04q_55125	6	10	-	-	MO	-	-	TO	6	**8**	MO	**8**	2.6	**8**	17	**8**	37.1
04q_99346	5	21	-	-	MO	5	**7**	1092	5	9	MO	13	10.1	13	51.8	13	8.3
05q_05306	8	33	-	-	MO	-	-	TO	66	41	MO	-	MO	-	TO	**19**	1559
05q_33936	9	37	-	-	MO	-	-	TO	107	60	MO	-	MO	-	TO	**21**	TO
05q_50494	8	34	-	-	MO	-	-	TO	86	54	MO	-	MO	-	TO	**20**	TO
05q_55125	9	35	-	-	MO	-	-	TO	12	16	MO	-	MO	-	TO	**19**	TO
05q_99346	7	25	-	-	MO	-	-	TO	16	17	MO	15	453	15	311	**15**	111

Table 4: E2: optimization of cx+1q-count via `costbased` and 1q-count via `rigid` domain; cx: CNOT count, 1q: 1q count, t: time in seconds; MO: memory out, TO: time out; Every instance without TO/MO is solved optimally; Best results in bold.

References

Aaronson, S.; and Gottesman, D. 2004. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5).

Arufe, L.; Rasconi, R.; Oddi, A.; Varela, R.; and González, M. Á. 2023. Solving quantum circuit compilation problem variants through genetic algorithms. *Nat. Comput.*, 22(4): 631–644.

Baioletti, M.; Fagiolo, F.; Oddi, A.; and Rasconi, R. 2025. Divide-Et-Impera Heuristic-Based Randomized Search for the Qubit Routing Problem. In *2025 IEEE International Conference on Quantum Artificial Intelligence (QAI)*, 276–281.

Bravyi, S.; Latone, J. A.; and Maslov, D. 2022. 6-qubit optimal Clifford circuits. *npj Quantum Information*, 8(1).

Ettenhuber, P.; Hansen, M. B.; Poier, P. P.; Shaik, I.; Rasmussen, S. E.; Madsen, N. K.; Majland, M.; Jensen, F.; Olsen, L.; and Zinner, N. T. 2025. Calculating the Energy Profile of an Enzymatic Reaction on a Quantum Computer. *Journal of Chemical Theory and Computation*, 21(7): 3493–3503. PMID: 40162965.

Helmert, M.; Röger, G.; and Karpas, E. 2011. Fast Downward Stone Soup: A Baseline for Building Planner Portfolios. In *ICAPS 2011 Workshop on Planning and Learning*, 28–35.

IBM. 2024. Shor Code: Correcting Phase Flip Errors. IBM Quantum Learning Course, https://quantum.cloud.ibm.com/learning/en/courses/foundations-of-quantum-error-correction/correcting-quantum-errors/shor-code.

Iwama, K.; Kambayashi, Y.; and Yamashita, S. 2002. Transformation rules for designing CNOT-based quantum circuits. In *Proceedings of the 39th Design Automation Conference, DAC 2002, New Orleans, LA, USA, June 10-14, 2002*, 419–424. ACM.

Jiang, J.; Sun, X.; Teng, S.; Wu, B.; Wu, K.; and Zhang, J. 2020. Optimal Space-Depth Trade-Off of CNOT Circuits in Quantum Logic Synthesis. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, 213–229. SIAM.

Mondal, A.; and Parhi, K. K. 2024. Optimization of quantum circuits for stabilizer codes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(8): 3647–3657.

Nagarajan, H.; Lockwood, O.; and Coffrin, C. 2021. QuantumCircuitOpt: An Open-source Framework for Provably Optimal Quantum Circuit Design. *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, 55–63.

Nielsen, M. A.; and Chuang, I. L. 2010. *Quantum circuits*, 171–215. Cambridge University Press.

Oddi, A.; and Rasconi, R. 2018. Greedy Randomized Search for Scalable Compilation of Quantum Circuits. In van Hoeve, W. J., ed., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings*, volume 10848 of *Lecture Notes in Computer Science*, 446–461. Springer.

Pozo, M.; and Seipp, J. 2025. Abstraction Heuristics for Classical Planning Tasks with Conditional Effects. In Kwok, J., ed., *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, 8608–8616. IJCAI.

Qiskit contributors. 2023. Qiskit: An Open-source Framework for Quantum Computing.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Rintanen, J. 2014. Madagascar: Scalable Planning with SAT. In *8th International Planning Competition*.

Shaik, I.; and van de Pol, J. 2023. Optimal Layout Synthesis for Quantum Circuits as Classical Planning. In *IEEE/ACM International Conference on Computer Aided Design, ICCAD 2023, San Francisco, CA, USA, October 28 - Nov. 2, 2023*, 1–9. IEEE.

Shaik, I.; and van de Pol, J. 2024. Optimal Layout-Aware CNOT Circuit Synthesis with Qubit Permutation. In *ECAI 2024*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, 4207–4215. IOS Press.

Shaik, I.; and van de Pol, J. 2025. CNOT-Optimal Clifford Synthesis as SAT. In *28th IC on Theory and Applications of Satisfiability Testing, SAT 2025, August 12-15, 2025, Glasgow, Scotland*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Shor, P. W. 1995. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52: R2493–R2496.

Sivarajah, S.; Dilkes, S.; Cowtan, A.; Simmons, W.; Edgington, A.; and Duncan, R. 2020. t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1): 014003.

Speck, D.; and Helmert, M. 2025. On Performance Guarantees for Symbolic Search in Classical Planning. In Inês; and Murano, A., eds., *Proceedings of the Twenty-Eighth European Conference on Artificial Intelligence (ECAI 2025)*. IOS Press.

Steane, A. M. 1999. Enlargement of Calderbank-Shor-Steane quantum codes. *IEEE Trans. Inf. Theory*, 45(7): 2492–2495.

van de Wetering, J.; and Amy, M. 2024. Optimising quantum circuits is generally hard. arXiv:2310.05958.

Venturelli, D.; Do, M.; Rieffel, E.; and Frank, J. 2017. Temporal Planning for Compilation of Quantum Approximate Optimization Circuits. In *Proceedings of IJCAI-17*, 4440–4446.