A SAT Encoding for Optimal Clifford Circuit Synthesis

Sarah Schneider* Lukas Burgholzer* Robert Wille‡†
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
‡Chair for Design Automation, Technical University of Munich, Germany
†Software Competence Center Hagenberg GmbH (SCCH), Austria
sarah.schneider@jku.at,lukas.burgholzer@jku.at,robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

ABSTRACT

Executing quantum algorithms on a quantum computer requires compilation to representations that conform to all restrictions imposed by the device. Due to device's limited coherence times and gate fidelities, the compilation process has to be optimized as much as possible. To this end, an algorithm's description first has to be *synthesized* using the device's gate library. In this paper, we consider the *optimal* synthesis of *Clifford* circuits—an important subclass of quantum circuits, with various applications. Such techniques are essential to establish lower bounds for (heuristic) synthesis methods and gauging their performance. Due to the huge search space, existing optimal techniques are limited to a maximum of six qubits. The contribution of this work is twofold: First, we propose an optimal synthesis method for Clifford circuits based on *encoding* the task as a satisfiability (*SAT*) problem and solving it using a SAT solver in conjunction with a binary search scheme. The resulting tool is demonstrated to synthesize optimal circuits for up to 26 qubits—more than four times as many as the current state of the art. Second, we experimentally show that the overhead introduced by state-of-the-art heuristics exceeds the lower bound by 27 % on average. The resulting tool is publicly available at https://github.com/cda-tum/qmap.

1 INTRODUCTION

In quantum computing, algorithms are generally described via quantum circuits—potentially consisting of high-level operations such as the quantum Fourier transform [1] or Grover iterations [2]. Alternatively, functional descriptions such as a unitary matrix, a decision diagram, or a tensor network can be used to describe the functionality of a particular algorithm. Quantum computing architectures provide a certain set of *native* gates, i.e., quantum gates that can be directly performed on the quantum computer. These native gate-sets typically consist of a few single-qubit gates, as well as a particular two-qubit gate such as the CNOT—forming a universal set of gates [1]. Similar to how, in classical computing, high-level C code has to be compiled to machine-dependent assembly, in quantum computing, an algorithm's high-level functional description has to be *synthesized* using the respective gate library [3]–[6].

Due to the limited fidelity and coherence times of (today's) quantum computers, it is critical that the synthesis is conducted as efficiently as possible. Efficiency in this regard can relate to a variety of metrics, such as minimizing the *number of gates* or the *depth* of the resulting circuit—roughly corresponding to the number of instructions or the runtime of the circuit. More precisely, the number of *two-qubit* gates has been identified as the main source for errors in currently available quantum systems and, thus, has been predominantly used as a cost metric for synthesizing circuits in the past. Hence, we also focus on minimizing the two-qubit gate count in the following.

In this work, we consider the question: Given a quantum circuit, what is the most efficient realization of the respective functionality. We particularly focus on the synthesis of *Clifford* circuits, which form a central class of quantum circuits, important for error correcting codes [7], [8] and several important quantum phenomena, such as superposition, entanglement, superdense coding, as well as teleportation [1]. Studying synthesis approaches that guarantee optimal solutions is important for establishing lower bounds on the achievable efficiency and for evaluating how far existing and future synthesis methods stray from this optimum. Even though Clifford circuits form a finite subgroup of all quantum circuits—one that is not even universal for quantum computing—the search space for the synthesis problem grows exponentially with respect to the number of considered qubits. As a result and to the best of our knowledge, the largest number of qubits for which an optimal method has been proposed is *six* [9]—significantly limiting the possibilities for truly evaluating the performance of heuristic synthesis methods.

In order to tackle this limitation, we propose to encode the optimal synthesis task as a satisfiability (*SAT*, [10]) problem and solve it using a SAT solver in conjunction with a binary search scheme. Besides their wide-spread use in the classical domain, these solvers have proven quite powerful in efficiently navigating the huge search spaces encountered in various problems related to quantum computing [11]–[16]. Experimental results demonstrate that the resulting tool [1] is capable of synthesizing optimal circuits with up to 26 qubits within a timeframe of ≈ 3 h—more than four times as many qubits as the state of the art. Furthermore, we show that the state-of-the-art heuristic for Clifford circuit synthesis (proposed in [17] and available in IBM's Qiskit [18]) exceeds this lower bound by 27 % on average.

The remainder of this work is structured as follows: Section 2 gives a brief overview of quantum computing and Clifford circuits. Section 3 reviews the synthesis problem for quantum circuits and related work. Then, Section 4 describes the proposed encoding. Based on that, Section 5 describes how to efficiently determine optimal solutions. Then, Section 6 summarizes the experimental results, before Section 7 concludes the paper.

[1] Publicly available under the MIT license as part of Munich Quantum Toolkit (MQT) in the QMAP tool (https://github.com/cda-tum/qmap)

2 BACKGROUND

In this section, we give a brief introduction to quantum computing and the main concepts needed throughout this work. While the individual descriptions are kept brief, we refer to [1], [7], [19] for a more in-depth introduction.

2.1 Quantum Computing

In classical computing, the basic unit of information is called a *bit* and can assume either one of the two states 0 and 1. In quantum computing, the basic unit of information is called a quantum bit (*qubit*) and cannot only assume either one of two basis states, $|0\rangle$ and $|1\rangle$, but also any complex-valued linear combination (*superposition*) thereof, i.e., the state $|\psi\rangle$ of a qubit is defined by $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, with $\alpha_0, \alpha_1 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The α_i are called amplitudes and are frequently written in the form of a state vector $|\psi\rangle \equiv \begin{bmatrix}\alpha_0 & \alpha_1\end{bmatrix}^\top$. Measuring (i.e., observing) the qubit causes its state to collapse to one of the basis states $|i\rangle$—each with probability $|\alpha_i|^2$.

The basis for an n-qubit state is formed by the tensor product of single-qubit states, i.e., $|i_{n-1}\rangle \otimes \cdots \otimes |i_0\rangle \equiv |i_{n-1} \dots i_0\rangle$, with $i_j \in \{0, 1\}$ for j from 0 to $n-1$. Consequently, any n-qubit state can be described by $|\psi\rangle \equiv \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ with $\alpha_i \in \mathbb{C}$ and $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. One fundamental difference between classical and quantum states is *entanglement*. A quantum state is said to be entangled if the state of its qubits cannot be considered separately, but has to be considered as a whole.

Example 1. *The four Bell states*

$$\left|\Phi^{\pm}\right\rangle = 1/\sqrt{2}\,|00\rangle \pm |11\rangle \;\; \textit{and} \;\; \left|\Psi^{\pm}\right\rangle = 1/\sqrt{2}\,|01\rangle \pm |10\rangle$$

are some of the most well-known examples of entangled states. Measuring, e.g., the first qubit of a system in the $\left|\Psi^{+}\right\rangle$ state results in either $|01\rangle$ or $|10\rangle$—each with probability $|1/\sqrt{2}|^2 = 0.5$. Thus, the state of the second qubit after the measurement depends on the state of the first one, even if it is not measured directly.

In contrast to classical computing, any operation manipulating the state of a quantum system (a *quantum gate*) is inherently reversible. To this end, the functionality of a quantum gate acting on k qubits is described by a $2^k \times 2^k$-dimensional unitary matrix U. Applying a quantum gate g to the state $|\psi\rangle$ of a system corresponds to appropriately multiplying the corresponding unitary matrix U with the current state vector—resulting in a new state $|\psi'\rangle$.

Example 2. *Some of the most fundamental single-qubit gates are the three Paulis, whose matrices are given by*

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

In addition, the following three gates will be heavily used in the remainder of this work:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \; S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}, \; \textit{and} \; CNOT = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}.$$

A quantum circuit, like its classical counterpart, is then made up of a sequence of quantum gates. This is conveniently described as a (quantum) circuit diagram, where circuit lines represent individual qubits and labeled boxes placed on them indicate the gates that are applied in sequence. Executing a quantum circuit on some initial state $|\psi\rangle$ (typically assumed to be $|0 \dots 0\rangle$) corresponds to successively applying the individual gates to the state.

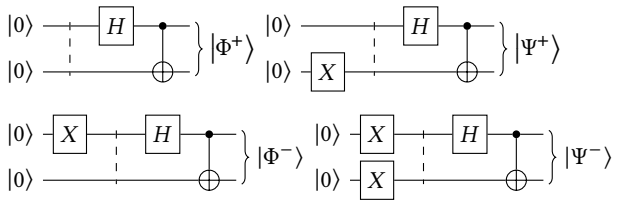


Figure 1: Quantum circuit generating the four Bell states

Example 3. *Consider again the four Bell states from Example 1. Then, Fig. 1 shows the quantum circuits to generate these states from the all-zero state $|00\rangle$. First, X gates are applied in order to prepare the various two-qubit basis states. Then, a Hadamard (H) gate is applied to the top qubit, followed by a CNOT controlled on the top qubit (indicated by •) and targeted at the bottom qubit (indicated by ⊕)—eventually resulting in the Bell states shown earlier.*

2.2 Clifford Circuits and the Stabilizer Formalism

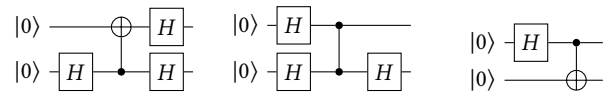
Clifford circuits, i.e., quantum circuits generated from the set of gates $\{H, S, CNOT\}$, form one of the most important classes of quantum circuits [1]. This is due to several factors:

- According to the Gottesman-Knill theorem [20], they can be simulated in polynomial time and space on classical computers using the *stabilizer* formalism.
- They can be used to describe several quantum phenomena such as superposition, entanglement, superdense coding, and teleportation [1].
- Many error correcting codes rely on them [7], [8].

The idea of the stabilizer formalism is to represent a quantum state not by a complex-valued vector of amplitudes, but by a set of operators that uniquely identify the state. A unitary operator U is said to be a *stabilizer* of a quantum state $|\psi\rangle$ if $U|\psi\rangle = |\psi\rangle$, i.e., $|\psi\rangle$ is an eigenvector of U with eigenvalue 1.

Example 4. *It is easy to see that the zero state $|0\rangle$ is stabilized by the Pauli Z operator since $Z|0\rangle = |0\rangle$. The plus state, i.e., the state $|+\rangle$ defined as $1/\sqrt{2}(|0\rangle + |1\rangle)$, is stabilized by the Pauli X operator since $X|+\rangle = |+\rangle$.*

Quantum states that can be obtained from the all-zero basis state $|0 \dots 0\rangle$ by applying Clifford operations are called *stabilizer states*. The name originates from the fact that such a state is uniquely and efficiently described by the set of operators that generate the group of its stabilizers. Specifically, any n-qubit stabilizer state can be described by a set of n Pauli strings $\pm P_{i,0}P_{i,1}P_{i,2}\dots P_{i,n-1}$, with $P_{i,j} \in \{I, X, Y, Z\}$ and $i, j \in 0, \dots, n-1$. Hence, two bits per qubit are needed to identify the Pauli operator, as well as one additional bit for the phase, which leads to a total of $n(2n+1)$ bits needed to uniquely describe a particular stabilizer state.



(a) Using H and $CNOT$ (b) Using H and CZ (c) Optimal circuit

Figure 2: Three circuits implementing the same tableau

Example 5. *The stabilizer representation of a quantum state is conveniently described by a* tableau

$$\left[\begin{array}{ccc|ccc|c} x_{0,0} & \cdots & x_{0,n-1} & z_{0,0} & \cdots & z_{0,n-1} & r_0 \\ \vdots & \iddots & \vdots & \vdots & \iddots & \vdots & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,n-1} & z_{2n-1,0} & \cdots & z_{n-1,n-1} & r_{n-1} \end{array}\right],$$

or (in a more compact fashion) by

$$\left[\mathbf{x}_0 \cdots \mathbf{x}_{n-1} \;\middle|\; \mathbf{z}_0 \cdots \mathbf{z}_{n-1} \;\middle|\; \mathbf{r}\right].$$

Here, the binary variables x_{ij} and z_{ij} specify whether the Pauli term $P_{i,j}$ is X or Z, respectively. Since $Y = iXZ$, setting $x_{ij} = z_{ij} = 1$ corresponds to $P_{i,j} = Y$. Finally, r_i describes whether the generator has a negative phase.

The Gottesman-Knill theorem states that the generators of a stabilizer state can be updated in polynomial time after the application of a Clifford operation. To this end, the following update rules for the stabilizer tableau apply:

- H on qubit j: $\mathbf{x}_j \leftrightarrow \mathbf{z}_j$ and $\mathbf{r} \oplus\!= \mathbf{x}_j\mathbf{z}_j$,
- S on qubit j: $\mathbf{z}_j \oplus\!= \mathbf{x}_j$ and $\mathbf{r} \oplus\!= \mathbf{x}_j\mathbf{z}_j$, and
- $CNOT$ with control qubit c and target qubit t, i.e., $\mathbf{x}_t \oplus\!= \mathbf{x}_c$, $\mathbf{z}_c \oplus\!= \mathbf{z}_t$, and $\mathbf{r} \oplus\!= \mathbf{x}_c\mathbf{z}_t(\mathbf{x}_t \oplus \mathbf{z}_c \oplus 1)$, with $\oplus$ denoting the XOR operation.

Using the stabilizer tableau in conjunction with these update rules allows to efficiently simulate Clifford circuits.

Example 6. *Consider the simulation of the circuit shown in Fig. 2a. It starts off in the all-zero state $|00\rangle$, which has the stabilizer tableau representation*

$$\left[\begin{array}{cc|cc|c} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right], \text{ with generators } \{ZI, IZ\} \doteq |00\rangle\,.$$

Applying the H gate to qubit 0 yields

$$\left[\begin{array}{cc|cc|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right], \text{ with generators } \{XI, IZ\} \doteq |0+\rangle\,.$$

Applying the CNOT gate with control 0 and target 1 yields

$$\left[\begin{array}{cc|cc|c} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right], \text{ with generators } \{XX, ZZ\} \doteq \left|\Phi^+\right\rangle\,.$$

Eventually, applying both H gates yields

$$\left[\begin{array}{cc|cc|c} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{array}\right], \text{ with generators } \{ZZ, XX\} \doteq \left|\Phi^+\right\rangle\,.$$

Hence, this circuit prepares the Bell state $\left|\Phi^+\right\rangle$ whose stabilizers are generated by XX and ZZ.

3 SYNTHESIS OF CLIFFORD CIRCUITS

In this section, we review what it means to *synthesize* Clifford circuits, why it is important to do this efficiently, and explore how this task has been conducted in the past.

3.1 The Synthesis Problem

Synthesis is the task of determining a quantum circuit that implements a given functionality under certain constraints. This functionality can be described in a multitude of ways, e.g., a (high-level) quantum circuit, a unitary matrix, a decision diagram, a tensor network, or—in case of Clifford circuits—a stabilizer tableau. The constraints typically originate from certain restrictions of actual quantum computers—predominantly their limited gate-set.

Example 7. *Consider the final tableau from Example 6, i.e.,*

$$\left[\begin{array}{cc|cc|c} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{array}\right].$$

As shown before, this tableau describes the Bell state $\left|\Phi^+\right\rangle$ and one possible circuit implementing this functionality has already been shown in Fig. 2a. Fig. 2b shows an alternative realization of the circuit using a controlled-Z gate instead of the CNOT gate.

In general, there are infinitely many ways to realize a given functional description as a quantum circuit. However, due to the noisy nature of near-term quantum computers and the limited coherence times of their qubits, it is important to determine *efficient* realizations. In this regard, *efficiency* can be defined in terms of the *gate-count* or the resulting circuit's *depth*—more or less translating to the number of instructions and the runtime of the quantum circuit. Specifically, a circuit is more efficiently synthesized if it has a lower number of gates or a lower depth.

Example 8. *Consider again the tableau from the previous example. Then, the most efficient circuit implementing the desired functionality only consists of two gates—a Hadamard and a single CNOT—as shown in Fig. 2c. While, in this case, the difference between these three realizations is quite small, it can grow tremendously large in more practical scenarios (as, e.g., demonstrated later in Section 6).*

Most existing solutions (which are reviewed next), strive to optimize the two-qubit gate-count of the resulting circuits, which is well motivated by the fact that two-qubit gates are the predominant source of errors in quantum circuits executed on today's devices. Thus, in this work, we also consider the two-qubit gate-count as the main objective.

3.2 Related Work and Motivation

Determining efficient realizations is a hard problem. Even for a finite group such as the Clifford group, the search space grows rapidly. In fact, the size of the n-qubit Clifford group grows as $2^{\Theta(n^2)}$. One of the key results for efficiently synthesizing Clifford circuits has been proposed by Aaronson and Gottesman [19]. They introduce a canonical form for Clifford circuits and demonstrate how this canonical representation can be generated from an arbitrary stabilizer tableau. An important corollary of their results is that any n-qubit Clifford circuit can be realized using at most $O(n^2/\log n)$ gates—establishing an (asymptotically-optimal) *upper* bound on the required resources.

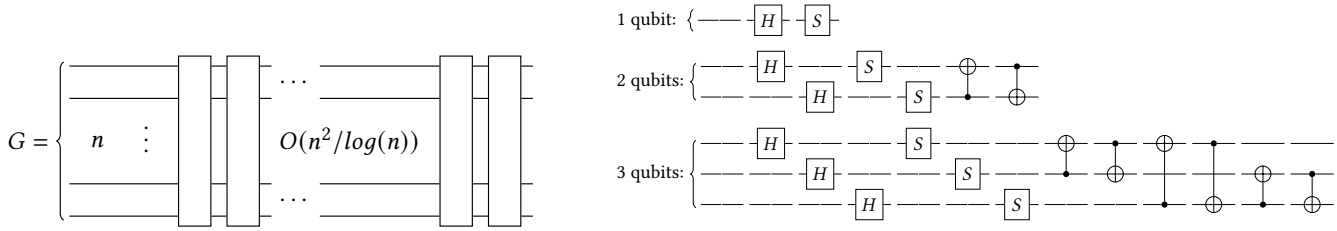


Figure 3: Search space of gate choices for the synthesis problem

In order to establish *lower* bounds on the achievable efficiency and to judge how well existing synthesis techniques perform, it is important to study synthesis approaches that guarantee *optimal* solutions. Due to the immense search space, the number of qubits for which optimal Clifford circuits are known is shockingly small. In [21], optimal Clifford circuits with up to four inputs are reported while, to the best of our knowledge, the largest number of qubits for which optimal realizations have been demonstrated is *six*. This has been shown by *Bravyi et al.* [9], where a pruned breadth-first search scheme is used to generate a database of classes of Clifford unitaries with known optimal canonical representatives—complemented with efficient means to search the created database[2].

In this work, we propose to overcome this qubit limitation by using satisfiability techniques (SAT, [10]) to encode the synthesis problem (as described next in Section 4) and solve it optimally using a SAT solver in conjunction with a binary search scheme (as described later in Section 5). SAT techniques have seen great success in the design of classical circuits and are increasingly getting used to efficiently navigate the huge search spaces of various problems in quantum computing [11]–[16].

4 ENCODING CLIFFORD CIRCUITS FOR SYNTHESIS

In the following, we construct a generic satisfiability encoding for the functionality of a Clifford circuit, i.e, the gates and the corresponding tableau. Furthermore, we show how the synthesis problem can be formulated based on this encoding and how the cost metric is specified.

4.1 Encoding the Functionality of a Clifford Circuit

As reviewed in Section 3.2, any n-qubit Clifford circuit can be realized using $O(n^2/\log n)$ gates. Each of these gates can either be a single-qubit H or S gate on any qubit, or a two-qubit CNOT between any pairs of qubits—spanning the search space for the synthesis. Thus, overall, at any point in time, there are

$$1 + 2n + 2 \cdot n(n-1)/2 = 1 + n + n^2 = O(n^2)$$

possible choices for gates. As a consequence, $O(n^4/\log n)$ Boolean variables suffice to describe all possible gates implementing a tableau, i.e., the structure of an arbitrary n-qubit Clifford circuit.

[2]For reference, this database took 6 months to generate and requires 2.1 TB of space.

Example 9. *Fig. 3 illustrates the structure of the search space for the synthesis problem and shows all different choices of gates for up to three qubits:*

- *For $n = 1$, there are just three options—doing nothing, apply a Hadamard gate, or applying an S gate.*
- *For $n = 2$, the single-qubit possibilities are complemented by two possible applications of a CNOT gate—resulting in a total of seven possible choices.*
- *For $n = 3$, this number increases to thirteen.*

Based on that, encoding the functionality of a Clifford circuit for a given input state can be accomplished by representing a stabilizer tableau for the application of every gate. Since each of these tableaus again requires to store $O(n^2)$ bits, another set of $O(n^4/\log n)$ Boolean variables are needed.

In addition, functional constraints need to be added to relate the gates of the circuit with the respective tableaus. These have the form:

$\forall$ gate g of the circuit	$O(n^2/\log n)$
$\forall$ choice c for gate g	$O(n^2)$
$\forall$ row r of the tableau for gate g	$O(n)$
update row r of tableau for gate g depending on gate choice c.	

This results in a total of $O(n^5/\log n)$ constraints to fully encode the functionality of a Clifford circuit in Boolean variables.

Example 10. *Assume that $n = 2$, as, e.g., for the circuits shown in Fig. 2. Then, an additional $n(2n+1) = 10$ Boolean variables are required per considered gate to encode the respective stabilizer tableau. As per Example 9, there are 7 possible choices per gate. This leads to a total of $7 \cdot 2 = 14$ functional constraints per gate.*

4.2 Formulating the Synthesis Problem

Based on the encoding proposed above, the synthesis problem can be formulated as illustrated in Fig. 4. Given an n-qubit Clifford circuit, the synthesis starts off by using the polynomial algorithm proposed by Gottesman and Knill [20] to construct what we refer to as the *target tableau* in the following (shown on right-hand side of Fig. 4). Alternatively, the target tableau can be explicitly given as input to the synthesis method. Starting from the tableau for the all-zero state $|0\ldots0\rangle$ (shown on the left-hand side of Fig. 4), the goal is to determine an assignment of the encoding's variables (identifying a Clifford circuit) that realizes this target tableau.

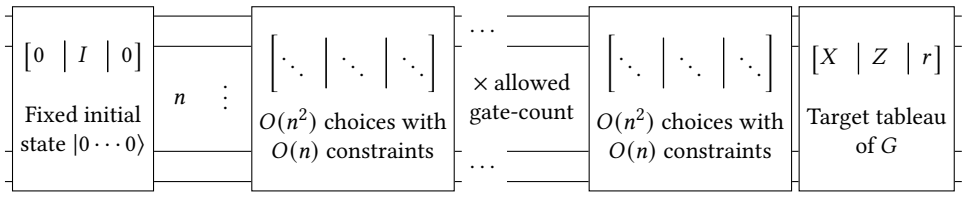


Figure 4: Formulation of the synthesis problem based on the proposed encoding

In order to efficiently guide the synthesis and to judge the quality of a solution, a cost metric (or objective function) is specified. As discussed in Section 3.1, the main focus in this work is mininizing the number of two-qubit gates which are considered the predominant source of errors. In the encoding proposed above, the number of gates in the resulting circuit corresponds to the respective gate-choice variables which are not assigned the identity. Naturally, this can be limited further to only consider two-qubit gates. The cost metric adds one additional cost constraint for each (two-qubit) gate-choice at any time-step, i.e., $O(n^4/\log n)$ constraints in total.

Example 11. *Considering the circuits shown in Fig. 2 and assuming equal costs for all gates, the following costs arise: The circuits in Fig. 2a and Fig. 2b have a cost of* 4*, while the circuit from Fig. 2c has a cost of* 2*. Due to this, the last circuit is the most optimal of these three with regard to the gate-count (in fact it is the most optimal among all two-qubit Clifford circuits realizing the considered target tableau).*

5 THE RESULTING OPTIMAL SYNTHESIS METHOD

As discussed in Section 3.2, optimal synthesis approaches are crucial for establishing lower bounds for the achievable efficiency and evaluating the performance of heuristic techniques. The encoding proposed in the previous section is naturally suited for determining optimal solutions to the synthesis problem using SAT or SMT solvers. Given a target tableau, determining an optimal circuit realizing this tableau corresponds to constructing a SAT instance based on the proposed encoding and then minimizing the given cost function.

5.1 Finding an Initial Time-Step Limit

The runtime of the synthesis significantly depends on the number of time-steps considered, since $O(n^2)$ variables and $O(n^3)$ constraints are generated for each time-step. Thus, it is critical to find a suitably low initial guess for this number. A straight-forward solution would be to take the minimum of the cost of the input circuit and the upper bound given by the canonical form of Aaronson and Gottesman [19]. While this is guaranteed to produce a satisfiable solution, i.e., the optimum is guaranteed to be in the search space of the SAT solver, it might result in an encoding that is excessively large.

Instead, the proposed encoding itself can be used to determine a more efficient initial guess by starting with a small initial limit (e.g., $T = n$) and letting the SAT solver search for *any* solution within that time limit. In general, the search for a solution terminates very quickly whenever no feasible solution exists given the prescribed limit. Thus, the limit is gradually increased in a geometric fashion until a feasible solution is found.

5.2 Performing the Optimization

Once a suitable time-step limit T has been determined—either by the method described above, from the original circuit itself, or the canonical form of Aaronson and Gottesman—there are two options for performing the actual optimization:

In standard SAT encodings, each constraint added to the problem instance is regarded a *hard* constraint, i.e., it needs to be satisfied at all times in order for the obtained solution to be valid. Using these standard capabilities of a SAT solver to tackle the underlying optimization problem, a binary search scheme can be used to determine the optimal solution, by iteratively adapting the time-step limit. This proceeds until a feasible solution is found for T time-steps but no solution can be found for $T - 1$ time-steps.

In contrast, MaxSAT formulations also accept *soft* constraints which need not be satisfied necessarily. The goal of corresponding solvers is to *maximize* the number of satisfied soft constraints. Soft constraints generally are additionally associated with a weight, which is used to determine the relative importance of the constraint.

Overall, both procedures result in synthesis methods that allow to determine *optimal* realizations of Clifford circuits—with a clear tradeoff between them. While the MaxSAT approach automatically handles the optimization, the binary search scheme requires multiple executions of the SAT solver and iterative tuning of the time-step limit. On the other hand, each individual call to the SAT solver is significantly less complex compared to the MaxSAT procedure. As demonstrated in the following evaluations, the binary search scheme runs about an order of magnitude faster for the cases considered in this work—allowing to determine optimal solutions for a wider range of qubits than ever before.

6 EXPERIMENTAL RESULTS

The observations and resulting strategies proposed above can be used with any SAT-engine that allows for optimization and the inclusion of certain SMT theories, such as Z3 [22] or OptiMathSAT [23]. Based on the encoding in Section 4, the methods proposed in Section 5 have been implemented in C++ (using the SMT-solver Z3 [22]) and are publicly available under the MIT license as part of the Munich Quantum Toolkit (MQT) in the QMAP tool (https://github.com/cda-tum/qmap). The respectively obtained results are summarized in the following.

All evaluations have been conducted on an AMD Ryzen 9 5950X processor with 5 GHz and 128 GiB of main memory, running Ubuntu 20.04 LTS with *qiskit-terra* version 0.21.0 and Z3 version 4.9.2. In a first series of evaluations, we explored the range of qubits that optimal solutions can be determined for using the proposed techniques within a time limit of 3 h for each run.

Table 1: Experimental Results (averaged over ten random stabilizer tableaus per row)

	Proposed Optimal Solutions				Comparison Against Heuristic Solutions			
	MaxSAT		Binary Search		A.-G. [19]		*Bravyi et al.* [17]	
n	t [s]	$\|G\|$	t [s]	$\|G\|$	t [s]	$\|G\|$	t [s]	$\|G\|$
4	1 095.1	**20**	176.9	**20**	4.3	61 (+205.0%)	5.4	24 (+20.0%)
5	1 009.9	**25**	252.3	**25**	3.7	71 (+184.0%)	7.3	30 (+20.0%)
6	1 289.8	**55**	335.2	**55**	3.4	90 (+ 63.6%)	10.3	69 (+25.5%)
7	2 536.0	**72**	490.6	**72**	4.6	121 (+ 68.1%)	13.6	91 (+26.4%)
8	2 601.4	**85**	588.2	**85**	5.5	141 (+ 65.9%)	16.5	107 (+25.9%)
9	4 952.9	**108**	822.0	**108**	5.5	177 (+ 63.9%)	22.3	135 (+25.0%)
10	3 135.9	**114**	803.0	**114**	6.3	200 (+ 75.4%)	24.0	145 (+27.2%)
11	6 052.9	**136**	1 114.2	**136**	9.4	209 (+ 53.7%)	30.4	173 (+27.2%)
12	4 913.5	**162**	1 235.1	**162**	8.8	246 (+ 51.9%)	36.7	207 (+27.8%)
13	5 956.7	**194**	1 675.1	**194**	9.4	297 (+ 53.1%)	44.8	249 (+28.4%)
14	8 379.6	**198**	2 056.5	**198**	10.0	331 (+ 67.2%)	51.5	254 (+28.3%)
15	-	-	2 126.7	**237**	12.6	360 (+ 51.9%)	60.5	307 (+29.5%)
16	-	-	2 493.3	**260**	12.7	436 (+ 67.7%)	71.5	336 (+29.2%)
17	-	-	3 196.7	**290**	16.5	470 (+ 62.1%)	87.2	374 (+29.0%)
18	-	-	3 211.6	**291**	16.8	521 (+ 79.0%)	96.9	375 (+28.9%)
19	-	-	4 463.2	**354**	19.8	552 (+ 55.9%)	112.6	456 (+28.8%)
20	-	-	4 171.8	**391**	20.7	631 (+ 61.4%)	125.4	505 (+29.2%)
21	-	-	5 770.4	**437**	23.7	694 (+ 58.8%)	146.4	565 (+29.3%)
22	-	-	6 476.2	**465**	27.7	720 (+ 54.8%)	166.2	600 (+29.0%)
23	-	-	6 070.9	**507**	23.8	773 (+ 52.5%)	186.8	656 (+29.4%)
24	-	-	7 515.5	**539**	25.2	820 (+ 52.1%)	220.4	697 (+29.3%)
25	-	-	8 265.8	**544**	31.5	899 (+ 65.3%)	234.7	705 (+29.6%)
26	-	-	10 406.6	**601**	29.8	964 (+ 60.4%)	268.9	776 (+29.1%)

n: the number of qubits t [s]: runtime of the synthesis method $|G|$: the number of gates produced by the synthesis and the relative difference to the proposed optimal solution

To this end, starting from four qubits, ten random stabilizer tableaus per number of qubits were generated using IBM Qiskit, and handed to the proposed synthesis algorithm. The left half of Table 1 shows the respectively obtained results—with n denoting the number of qubits, t the averaged runtime of the respective scheme, and $|G|$ the averaged two-qubit gate count (timeouts are marked as –). These results show that, using the MaxSAT approach, allows to determine optimal results for up to 14 qubits, which is more than double the amount of qubits compared to the approach in [9]—at the cost of very long runtimes. In comparison, the binary search approach runs an order of magnitude faster, which allows it to determine optimal solutions up 26 qubits—more than four times as many as the current state of the art.

In a second series of evaluations, we compared the proposed synthesis methods against the canonical form by Aaronson and Gottesman [19] and the state-of-the-art heuristic synthesis method by *Bravyi et al.* [17] which are available in IBM Qiskit. Both methods were fed the same random stabilizer tableaus as in the first series of evaluations. In order to ensure a fair comparison of the resulting gate-counts, the circuits generated by IBM Qiskit were decomposed to H, S as well as *CNOT* gates and optimized once using the Qiskit *transpile* function with optimization level 2. The correspondingly obtained results are shown in the right half of Table 1. They show that the canonical form by Aaronson and Gottesman [19] and the state-of-the-art heuristic by *Bravyi et al.* [17], respectively, exceed the lower bound by 72 % and 27 % on average.

7 CONCLUSIONS

In this work, we proposed a new method for the optimal synthesis of Clifford circuits based on the idea of encoding the task as a satisfiability problem and solving it using a SAT solver in conjunction with a binary search scheme. The resulting tool is capable of synthesizing optimal Clifford circuits for more qubits than ever before—allowing to evaluate the performance of existing and future heuristic synthesis methods on a much broader basis. We demonstrated the efficiency of the proposed approach by synthesizing optimal Clifford circuits with up to 26 qubits. Furthermore, we showed that there is still lots room for improvement in state-of-the-art heuristics for Clifford synthesis. Future work includes exploring different cost metrics as well as trying to extract a scalable heuristic from the proposed solution.

ACKNOWLEDGMENTS

The authors would like to thank R. Kueng for his valuable advice and insightful discussions over the course of this project.

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001318), was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and has been supported by the BMWK on the basis of a decision by the German Bundestag through project QuaST, as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.
[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proc. of the ACM*, 1996.
[3] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A*, 1995.
[4] D. Maslov, "On the advantages of using relative phase Toffolis with an application to multiple control Toffoli optimization," *Phys. Rev. A*, 2016.
[5] R. Wille *et al.*, "Improving the mapping of reversible circuits to quantum circuits using multiple target lines," in *Asia and South Pacific Design Automation Conf.*, 2013.
[6] A. M.-v. de Griend and R. Duncan, *Architecture-aware synthesis of phase polynomials for NISQ devices*, 2020. arXiv: 2004.06052.
[7] D. Gottesman, "Stabilizer codes and quantum error correction.," Caltech, 1997.
[8] S. J. Devitt, K. Nemoto, and W. J. Munro, "Quantum error correction for beginners," *Rep. Prog. Phys.*, 2013. arXiv: 0905.2794.
[9] S. Bravyi, J. A. Latone, and D. Maslov, *6-qubit optimal Clifford circuits*, 2020. arXiv: 2012.06074.
[10] A. Biere *et al.*, *Handbook of Satisfiability*. IOS Press, 2009.
[11] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Design Automation Conf.*, 2019.
[12] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *Int'l Conf. on CAD*, 2020.
[13] L. Berent, L. Burgholzer, and R. Wille, "Towards a SAT encoding for quantum circuits: A journey from classical circuits to Clifford circuits and beyond," in *International Conference on Theory and Applications of Satisfiability Testing*, 2022. arXiv: 2203.00698.
[14] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," in *Int'l Symp. on Nanoscale Architectures*, 2010.
[15] A. Matsuo, W. Hattori, and S. Yamashita, "Reducing the overhead of mapping quantum circuits to IBM Q system," in *IEEE International Symposium on Circuits and Systems*, 2019.
[16] G. Meuli, M. Soeken, and G. Micheli, "SAT-based {CNOT, T} quantum circuit synthesis," in *Int'l Conf. of Reversible Computation*, 2018.
[17] S. Bravyi *et al.*, "Clifford circuit optimization with templates and symbolic Pauli gates," *Quantum*, 2021. arXiv: 2105.02291.
[18] G. Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," *Zenodo*, 2019.
[19] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A*, 2004.
[20] D. Gottesman, *The Heisenberg representation of quantum computers*, 1998. arXiv: quant-ph/9807006.
[21] V. Kliuchnikov and D. Maslov, "Optimization of Clifford circuits," *Phys. Rev. A*, 2013. arXiv: 1305.0810.
[22] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2008.
[23] R. Sebastiani and P. Trentin, "OptiMathSAT: A tool for optimization modulo theories," *J Autom Reasoning*, 2020.