Reducing Quantum Circuit Synthesis to #SAT

Dekel Zak
Leiden University, The Netherlands

Jingyi Mei
Leiden University, The Netherlands

Jean-Marie Lagniez
Artois University, Lens, France

Alfons Laarman
Leiden University, The Netherlands

Abstract

Quantum circuit synthesis is the task of decomposing a given quantum operator into a sequence of elementary quantum gates. Since the finite target gate set cannot exactly implement any given operator, approximation is often necessary. Model counting, or #SAT, has recently been demonstrated as a promising new approach for tackling core problems in quantum circuit analysis. In this work, we show for the first time that the universal quantum circuit synthesis problem can be reduced to maximum model counting. We formulate a #SAT encoding for exact and approximate depth-optimal quantum circuit synthesis into the Clifford+T gate set. We evaluate our method with an open-source implementation that uses the maximum model counter d4Max as a backend. For this purpose, we extended d4Max with support for complex and negative weights to represent amplitudes. Experimental results show that existing classical tools have potential for the quantum circuit synthesis problem.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Quantum computation theory

Keywords and phrases Maximum weighted model counting, quantum circuit synthesis

Digital Object Identifier 10.4230/LIPIcs.CP.2025.38

Supplementary Material *Software (Source Code)*: `https://github.com/System-Verification-Lab/Quokka-Sharp.git`

Funding This work is supported by the Dutch National Growth Fund, as part of the Quantum Delta NL program.

1 Introduction

Quantum algorithms are typically specified using higher-level constructs such as (classical) procedures and quantum Fourier transform (QFT) [30]. For their efficient implementation on physical devices, which typically operate with a finite gate set, they need to be broken down into a quantum circuit, a sequence of quantum gates. Error-corrected hardware implementations, which realize the ideal quantum computational model formalized in a quantum Turing machine [8], often use the universal Clifford+T set $\{S, H, CX, T\}$ [20, 2, 42]. Moreover, better optimal synthesis solutions can have significant implications for quantum complexity theory, such as reducing the stabilizer rank of magic gadgets [31], and relating classical and quantum resources [11, 12]. However, exact synthesis is not always feasible due to the discrete nature of this gate set, often necessitating approximation techniques to balance precision and circuit depth [30]. The computational complexity of synthesis is formidable, with classical approaches exhibiting doubly exponential time in the worst case [33], underscoring the need for scalable and efficient methods.

Classical methods, like decision diagram [53, 13, 50], tree-automata [16], and SAT [48] have proven to be highly effective in analyzing quantum circuits. Recent advances have highlighted weighted model counting, or #SAT, as a powerful tool for addressing hard problems in quantum circuit analysis, including simulation [38] and equivalence checking [39]. These methods leverage off-the-shelf solvers to tackle the exact versions of these problems, known as #P-complete [18]. Building on this promise, we explore whether #SAT can be harnessed for the universal quantum circuit synthesis problem – a challenge that also requires approximation since discrete universal gate sets cannot exactly implement arbitrary quantum operators. The relevant complexity class here is QMA, as approximate circuit equivalence checking is complete for it [26]. Moreover, the Solovay-Kitaev theorem [17] guarantees that any unitary can be ϵ-approximated with a $\mathsf{polylog}(\frac{1}{\epsilon})$-depth circuit.

This work presents a novel reduction of quantum circuit synthesis to maximum model counting, focusing on depth-optimal and approximate synthesis into the Clifford+T gate set. To achieve this, we overcome two obstacles. First, to enable reduction to the maximum weighted model counting, we show how equivalence checking can be performed with a single model counting call, contrasting previous approaches [39] which required linearly many calls. In the process, we provide two new ways of encoding equivalence checking. Second, we show how the fidelity (a measure of circuit similarity) between two circuits can be computed by weighted model counting, generalizing our exact equivalence checking methods to support approximate equivalence checking. Our approach then encodes the synthesis task as a weighted conjunctive normal form (CNF) formula, where satisfying assignments correspond to valid gate sequences and weights reflect approximation fidelity. We demonstrate that this reduction enables both exact and approximate synthesis, with applications to circuit optimization. We provide an open-source implementation, called `Quokka#-syn`, to validate our method.

The scalability of quantum circuit synthesis remains a critical bottleneck, with existing methods struggling to handle large qubit counts or gate depths without resorting to corner-case omissions [37, 44]. We experimentally evaluate our method to compare the different encodings and test its scalability. In addition, we compare it against a state-of-the-art tool focusing on depth-optimal approximate and exact synthesis. While the comparison has limitations, it shows that our #SAT-based approach has merit and can offer improvement over existing methods. However, it falls short of fully resolving the synthesis problem's inherent complexity. We identify multiple avenues for improvement, such as enhancing maximum model counters with support for incremental solving. By laying this groundwork, we pave the way for integrating advanced classical solvers into the quantum computing toolkit, advancing the practical realization of quantum algorithms.

2 Preliminaries

2.1 Quantum Computing

In this section, we give the necessary notions and notations on quantum computing. For an entailed explanation, for example on tensor product ($\otimes$), see [42].

Quantum states. Let $\mathcal{H}^{\otimes n}$ be the 2^n-dimensional Hilbert space. An n-qubit quantum state, denoted as $|\varphi\rangle$ using Dirac notation, is a column vector $\left[\alpha_{00\dots00}, \dots, \alpha_{11\dots11}\right]^T$ in $\mathcal{H}^{\otimes n}$, where $|\alpha_b|^2 \in [0,1]$ are *amplitudes*, satisfying: $|\alpha_{00\dots00}|^2 + \dots + |\alpha_{11\dots11}|^2 = 1$. Its complex adjoint $\langle\varphi|$ is a row vector with conjugated entries: $\langle\varphi| = |\varphi\rangle^\dagger = \left[\alpha^*_{00\dots00}, \dots, \alpha^*_{11\dots11}\right]$, therefore

$\langle\varphi|\varphi\rangle = 1$. A quantum state vector can be decomposed in the *computational basis*, written as $|\varphi\rangle = \sum_{b\in\{0,1\}^n} \alpha_b |b\rangle$, where $|b\rangle$ is a *computational basis state* defined as a vector with all entries setting to 0 except at index b setting to 1.

Another way to represent a state $|\varphi\rangle$ is as a *density matrix* $\rho = |\varphi\rangle\langle\varphi|$. The *trace* of a density matrix is 1, denoted as $\mathrm{tr}(\rho) = 1$, where trace is defined as the sum of the diagonal elements of a matrix. A density matrix can be decomposed in the *Pauli basis* as in Equation 1. To understand Pauli basis decomposition, we introduce *Pauli matrices* and *Pauli strings*. The Pauli matrices are $I = \left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right], X = \left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right], Y = \left[\begin{smallmatrix}0&-i\\i&0\end{smallmatrix}\right], Z = \left[\begin{smallmatrix}1&0\\0&-1\end{smallmatrix}\right]$. An n-qubit *Pauli string* $\mathcal{P}$ is a parallel composition of n Pauli gates, such that $\mathcal{P} \in \{I, X, Y, Z\}^{\otimes n}$. For instance, $X \otimes Z \otimes I$ is a three-qubit Pauli string. It is worth noting that while a density matrix ρ may contain complex numbers, the basis *coefficients* β_i in Equation 1 are all real numbers [39]. Throughout this paper, we denote $[n] = \{0, ..., n-1\}$.

$$\rho = \sum_{j\in[4^n]} \beta_j \cdot \mathcal{P}_j \qquad \text{for the Pauli strings} \quad \mathcal{P}_j \in \{I, X, Y, Z\}^{\otimes n} \tag{1}$$

Quantum gates. An n-qubit quantum gate G can be expressed by a $2^n \times 2^n$ unitary matrix U, i.e., $U^\dagger \cdot U = U \cdot U^\dagger = I^{\otimes n}$. A single qubit quantum gate U operating on qubit $j \in [n]$ can be represented as $U_j = I^{\otimes j} \otimes U \otimes I^{\otimes n-j-1}$. Updating a quantum state in vector form $|\varphi\rangle$ is done by matrix-vector multiplication, i.e. $|\psi\rangle = U|\varphi\rangle$. Applying a unitary U to a density matrix $\rho = |\varphi\rangle\langle\varphi|$ should be done through *conjugation*, i.e.: $U\rho U^\dagger = U|\varphi\rangle\langle\varphi| U^\dagger = |\psi\rangle\langle\psi|$ for $|\psi\rangle = U|\varphi\rangle$. We consider the well-known universal gate set Clifford $\{S, H, CX\} + T$ and the gates *Toffoli* and CH, which are defined as follows:

$$H = 1/\sqrt{2}\left[\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right],\ S = \left[\begin{smallmatrix}1&0\\0&i\end{smallmatrix}\right],\ CX = \left[\begin{smallmatrix}I&\mathbf{0}\\\mathbf{0}&X\end{smallmatrix}\right],\ T = \left[\begin{smallmatrix}1&0\\0&i\end{smallmatrix}\right], \mathit{Toffoli} = \left[\begin{smallmatrix}I_6&\mathbf{0}\\\mathbf{0}&X\end{smallmatrix}\right],\ CH = \left[\begin{smallmatrix}I&\mathbf{0}\\\mathbf{0}&H\end{smallmatrix}\right]$$

where I_k is a k-dimensional identity matrix and $\mathbf{0}$ is the all zero matrix. We ignore the index when the dimension is 2, i.e. $I = I_2 = \left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$. We denote a gate set as $\mathcal{G}$. For an n-qubit circuit, let $\mathcal{G}(k) \subseteq \mathcal{G}$ be a subset of gates applied to $k \in [n]$ qubits, such that $\mathcal{G}(1)$ is the set of single-qubit gates, $\mathcal{G}(2)$ is the set of two-qubit gates and so on. For example, if $\mathcal{G} = \{H, T, CX, \mathit{Toffoli}\}$, $\mathcal{G}(1) = \{H, T\}$, $\mathcal{G}(2) = \{CX\}$ and $\mathcal{G}(3) = \{\mathit{Toffoli}\}$. We always assume $I \in \mathcal{G}$, also if not explicitly stated.

Quantum circuits. The evolution of a quantum system is modeled by a *quantum circuit*, a sequence of *quantum gate layers*, or *layers* in short, applied to all qubits at each time step. A *layer* D is a set of gates such that each qubit has at most one gate applied to it. Thus, it contains only mutually parallel gates. For example, a *layer* $D = \{G_i \mid i \in [n]\}$ applies a single-qubit local gate G on each qubit i. For any layer D, we will denote its unitary matrix with U_D. Let $\mathcal{D}_n$ be the set of all possible layers for n qubits. For a *circuit* $\mathcal{C} = (D^0, ..., D^{d-1})$, with $D^i \in \mathcal{D}_n$, we thus have its unitary $U_\mathcal{C} = U_{D^{d-1}} \cdots U_{D^0}$. We define a circuit's *depth* as the minimal number of layers (with parallel gates) that it contains.

Jamiołkowski fidelity The *fidelity* between two quantum states $|\psi\rangle$ and $|\varphi\rangle$ is defined as $\mathrm{Fid}(|\varphi\rangle, |\psi\rangle) \triangleq \mathrm{tr}(|\psi\rangle\langle\psi| \cdot |\varphi\rangle\langle\varphi|) = |\langle\psi|\varphi\rangle|^2$. The fidelity between states can be extended to measure the distance between unitary operators with the help of *Jamiołkowski isomorphism* that maps unitary U on $\mathcal{H}^{\otimes n}$ to a state $|\varphi_U\rangle = (U \otimes I)|\Psi_n\rangle$ on $\mathcal{H}^{\otimes 2n}$, where $|\Psi_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} |ii\rangle$ and $|ii\rangle$ is short for $|i\rangle \otimes |i\rangle$. Thus, *Jamiołkowski fidelity* [45] between unitary operators can be formally defined as:

$$\mathrm{Fid}_J(U, V) = \mathrm{Fid}(|\varphi_U\rangle, |\varphi_V\rangle) = |\langle\varphi_U|\varphi_V\rangle|^2. \tag{2}$$

In the particular case where $\mathrm{Fid}_J(U,V) = |\langle\varphi_U|\varphi_V\rangle|^2 = 1$, it follows that $U = \lambda V$ with $|\lambda|^2 = 1$. This result arises because $|\langle\varphi_U|\varphi_V\rangle|^2 = 1$ implies $|\varphi_U\rangle = \lambda|\varphi_V\rangle$, and from the definitions of $|\varphi_U\rangle$ and $|\varphi_V\rangle$, we conclude $U = \lambda V$.

2.2 Maximum Weighted Model Counting

Let A be a set of Boolean variables $\{a_1,...,a_m\}$. A *literal* ℓ is a variable or its negation, e.g., a or $\neg a$ (written as $\overline{a}$). A *clause* is a disjunction of literals $\ell_1 \vee ... \vee \ell_m$. A propositional formula F in *Conjunctive Normal Form* (CNF) is a conjunction of clauses. We write $F(A)$ to indicate that F is defined on the set of variables A. Let $\mathbb{B}$ be $\{0,1\}$ and $\mathbb{R}$ be the set of real numbers. An *assignment* τ maps variables A to $\mathbb{B}$, where $0,1$ represents *False* and *True* respectively. The satisfiability problem is determining whether an assignment τ to A exists, for which a propositional formula $F(A)$ is true. We denote $SAT(F(A)) \triangleq \{\tau \mid \tau \text{ satisfies } F(A)\}$ as the set of all satisfying assignments for F. The model counting problem is to compute the number of satisfying assignments, denoted as $\#SAT(F) \triangleq |SAT(F)|$.

A weight function $W : A \times \mathbb{B} \to \mathbb{R}$ maps a variable with its true or false assignment, or, viewed alternatively, a literal, to a weight. Given an assignment τ, the weight of this assignment, written $W(\tau)$, is the product of the weight of each variable with its assignment $W(\tau) = \prod_{a\in A} W(a,\tau(a))$. For notational convenience, we write the weights of literals, such that for a variable $a \in A$, we denote $W(a) = W(a,1)$ and $W(\bar{a}) = W(a,0)$. If unspecified, the weight of a variable $a \in A$ is assumed to be $W(a) = W(\bar{a}) = 1$. We call such variables *unbiased.* The weighted model counting (WMC) problem asks for the sum of the weights of the satisfying assignments, i.e., $\#SAT_W(F) = \sum_{\tau\in \mathrm{SAT}(F(A))} W(\tau)$.

The maximum weighted model counting problem (MWMC) extends WMC by finding an assignment to a subset of variables that maximizes the weight of the WMC problem (Definition 1). In Section 4, we show that quantum circuit synthesis can be reduced to it.

▶ **Definition 1** (MWMC [5]). *Given a propositional formula $F(A,B)$ over disjoint sets of variables A and B, and a weight function W over $(A\cup B)\times\mathbb{B}$, the MWMC problem is to determine an assignment τ to A that maximizes $\#SAT_W(F(A,B))$.*

For notation, we define an oracle function $Max\#SAT$, which takes the quantified Boolean formula $F(A,B)$ with its weight function W and returns an assignment τ to A that maximizes the weighted model count of the formula F and its maximal weight w_{max}:

$$Max\#SAT_W(F(A,B)) = (\tau(A), w_{max}).$$

2.3 Reducing Quantum Computing to Model Counting

We present two encodings: a computational basis encoding, referred to as **CB**, where the computational basis decomposition of the state vector is encoded directly, and a Pauli basis encoding, referred to as **PB**, where the basis states are the Pauli strings and we encode the density matrix decomposition (see Equation 1). A quantum state is encoded as a set of satisfying assignments where each satisfying assignment with its weight represents a *basis state* with its *amplitude* (in **CB**) or *coefficient* (in **PB**). Here, we briefly introduce both. For details, we refer to [38, 39, 40].

For an n-qubit quantum state $|\varphi\rangle$, we denote its WMC encoding with $F_{|\varphi\rangle}$. Note that in the **PB** basis, quantum states are represented using density matrices, so the corresponding encoding is technically $F_{|\varphi\rangle\langle\varphi|}$. However, we abuse the notation and write $F_{|\varphi\rangle}$ in both bases for simplicity. We reserve propositional variables $\vec{q} = (q_0,...,q_{n-1})$ in the **CB**, and $\vec{q} = (\vec{q}_0,...,\vec{q}_{n-1}) = (x_0,z_0,...,x_{n-1},z_{n-1})$ in the **PB**. (Here, we let $\overline{xz} \equiv I$, $x\overline{z} \equiv X$, $xz \equiv Y$

and $\overline{x}z \equiv Z$, and $\overline{x_k}z_k \wedge \bigwedge_{i\in[n]\setminus\{k\}} \overline{x_i z_i} \equiv Z_k$, etc, as in [1].) The variables in $\vec{q}$ remain unbiased. In addition, when needed, we introduce auxiliary variables to represent weights. Since the assignment to these auxiliary variables is always determined by the assignment to $\vec{q}$, we often omit these variables, writing $F_{|\varphi\rangle}(\vec{q})$ instead of $F_{|\varphi\rangle}(\vec{q},\vec{u})$. Table 1 gives encoding examples.

Applying a quantum gate maps a quantum state to another. The encoding of a quantum gate G is given by a Boolean function written as $F_G(\vec{q},\vec{q}')$, where $\vec{q}$ is the input state and $\vec{q}'$ is output state after applying G, such that $F_{|\psi\rangle}(\vec{q}') = F_{|\varphi\rangle}(\vec{q}) \wedge F_G(\vec{q},\vec{q}')$ for $|\psi\rangle = G\,|\varphi\rangle$. As with state encodings, we introduce auxiliary variables when needed to represent weights introduced by the gates, and often omit these variables from the function signature. We give an example of how to encode the gates H, T, and CX in Table 2. A layer D is encoded by conjoining the encodings of local gates, each applied to the variables of the relevant qubits. For example, a two-qubit layer $D = \{H_0, T_1\}$ is encoded by $F_D(\vec{q},\vec{q}') = F_H(\vec{q}_0,\vec{q}_0') \wedge F_T(\vec{q}_1,\vec{q}_1')$.

A quantum circuit $\mathcal{C} = (D^0,\ldots,D^{d-1})$ is encoded by reserving variables $\vec{q}^0,\ldots,\vec{q}^d$ for representing the initial, intermediate, and final states, and conjoining all the encoding of layers over these variables, i.e., $F_{\mathcal{C}}(\vec{q}^0,\ldots,\vec{q}^d) = \bigwedge_{j\in[d]} F_{D^j}(\vec{q}^j,\vec{q}^{j+1})$. When we don't need to name the intermediate states $\vec{q}^1,\ldots,\vec{q}^{d-1}$, we will omit them, writing $F_{\mathcal{C}}(\vec{q},\vec{q}') = F_{\mathcal{C}}(\vec{q}^0 = \vec{q},\ldots,\vec{q}^d = \vec{q}')$.

Lemma 2 shows that both WMC encodings allow for the strong simulation of any quantum circuit according to the usual definition of computing output amplitudes or coefficients [28].

▶ **Lemma 2** ([40]). *Given an input state $|\varphi\rangle = \sum_{b\in\{0,1\}^n} \alpha_b\,|b\rangle$, such that $|\varphi\rangle\langle\varphi| = \sum_{j\in[4^n]} \beta_j \mathcal{P}_j$, an n-qubit circuit $\mathcal{C}$, a computational basis state $|b\rangle$ ($b \in \{0,1\}^n$) and a*

Table 1 State encoding in both bases: The auxiliary variables, marked in gray, depend fully on the unbiased variables $\vec{q}$ (in **CB**) and $\vec{x},\vec{z}$ (in **PB**) representing basis states.

	Comp. Basis (CB)	**Pauli Basis (PB)**	**Auxiliary weights**
Variables	$\vec{q} = (q_0,\ldots,q_{n-1})$	$\vec{q} = (x_0,z_0,\ldots,x_{n-1},z_{n-1})$	h,r,g,w
$\|0\rangle \equiv$	$F_{\|0\rangle}(q) = \overline{q}$	$F_{\|0\rangle}(x,z,g) = F_{\frac{I+Z}{2}} = \overline{x}g$	$W(g) = (1/2)$
$\|-\rangle \equiv$	$F_{\|-\rangle}(q,h,r) = h(r \Leftrightarrow q)$	$F_{\|-\rangle}(x,z,g) = F_{\frac{I-X}{2}} = \overline{z}g$	$W(h) = (1/\sqrt{2})$, $W(r) = (-1)$
[25]: $\|\mathsf{A}\rangle \equiv$	$F_{\|A\rangle}(q,h,w) = h(w \Leftrightarrow q)$	$F_{\|A\rangle}(x,z,g,h) = F_{\frac{I}{2}+\frac{X+Y}{\sqrt{2}}}$	$W(w) = \omega$ where $\omega = \sqrt{i} = \frac{i+1}{\sqrt{2}}$
$\|00\rangle \equiv$	$F_{\|00\rangle}(q_0,q_1) = \overline{q_0 q_1}$	$F_{\|00\rangle}(x_0,z_0,x_1,z_1) = \overline{x_0 x_1}$	none
$\|{+}{+}\rangle \equiv$	$F_{\|{+}{+}\rangle}(q_0,q_1,g) = g$	$F_{\|{+}{+}\rangle}(x_0,z_0,x_1,z_1) = \overline{z_0 z_1}$	$W(g) = (1/2)$

Table 2 Gate encoding in both bases using the same weights as in Table 1.

	Comp. Basis (CB)	**Pauli Basis (PB)**
1-qubit unitary ($U1$)	$F_{U1}(\vec{q},\vec{q}')$ where $\vec{q} = (q_0)$	$F_{U1}(\vec{q},\vec{q}')$ where $\vec{q} = (x,z)$
n-qubit unitary (Un)	$F_{Un}(\vec{q},\vec{q}')$ where $\vec{q} = (q_0,\ldots,q_n)$	$F_{Un}(\vec{q},\vec{q}')$ where $\vec{q} = (x_0,z_0,\ldots,x_{n-1},z_{n-1})$
$H = 1/\sqrt{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$F_H(q,q',r,h) = h \wedge (r \Leftrightarrow qq')$	$F_H(q,q',r) = (r \Leftrightarrow xz) \wedge (z' \Leftrightarrow x) \wedge (x' \Leftrightarrow z)$
$T = \begin{bmatrix}1 & 0\\ 0 & \omega\end{bmatrix}$	$F_T(q,q',w) = (q \Leftrightarrow q') \wedge (w \Leftrightarrow q)$	*given in* [38]
$CX = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$	$(q_0' \Leftrightarrow q_0) \wedge (q_1' \Leftrightarrow (q_1 \oplus q_0))$	*given in* [38]

Pauli string $\mathcal{P}_j \in \{I, X, Y, Z\}^{\otimes n}$, the following equations hold:

$$\#SAT_W(F_{|\varphi\rangle}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q}, \vec{q}') \wedge F_{|b\rangle}(\vec{q}')) = \alpha_b \quad \textit{in} \quad \mathbf{CB},$$
$$\#SAT_W(F_{|\varphi\rangle}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q}, \vec{q}') \wedge F_{\mathcal{P}_j}(\vec{q}')) = \beta_j \quad \textit{in} \quad \mathbf{PB}.$$

We emphasize that strong simulation is canonical for quantum complexity classes [52], and therefore generalizes naturally to computing any measurement outcome probability [38] and determining circuit equivalence [39].

3 Problem Statement

The quantum circuit synthesis problem seeks to construct a circuit that implements a given specification, which is provided as either a circuit or a unitary operator. A key component is to determine if a guessed candidate circuit is *equivalent* to the desired specification by checking unitary equivalence, as formalized in Definition 3.

▶ **Definition 3** (Unitary equivalence). *Let U, V be two n-qubit unitaries, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the Jamiołkowski fidelity between U and V is not smaller than $1-\epsilon$, i.e. $\mathrm{Fid}_J(U, V) \geq 1 - \epsilon$, where $\epsilon \in [0, 1]$.*

In the above definition, we use the Jamiołkowski fidelity defined in Equation 2 to measure the distance between two unitaries. In particular, if and only if $\mathrm{Fid}_J(U, V) = 1$ ($\epsilon = 0$), we have *exact equivalence*, denoted as $U \equiv V$. In this case, U and V are equivalent up to a global phase λ satisfying $|\lambda|^2 = 1$, i.e. $U = \lambda V$. Building on equivalence checking, Problem 4 gives the formal definition of (exact and approximate) quantum circuit synthesis.

▶ **Problem 4** (Quantum circuit synthesis). *Given a specification represented by a circuit $\mathcal{C}_1$ in a gate set $\mathcal{G}_1$ or unitary $U_{\mathcal{C}_1}$ and an accuracy parameter $\epsilon \in (0, 1]$, the approximate synthesis problem asks for a depth-optimal quantum circuit $\mathcal{C}_2$ in a target gate set $\mathcal{G}_2$, such that $U_{\mathcal{C}_1} \simeq_\epsilon U_{\mathcal{C}_2}$. For $\epsilon = 0$, this is the exact synthesis problem.*

We consider Clifford+T as the elementary target gate set for synthesis because of its universality and importance in error-corrected quantum computing [20, 2]. While unitaries can be synthesized in Clifford+T (with additional ancillas) exactly if and only if their matrix entries belong to $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ [22], all unitaries can be synthesized approximately up to an arbitrarily small ϵ [17].

In this work, we consider both exact and approximate synthesis and, for the first time, show that both problems can be reduced to MWMC.

4 Exact Quantum Circuit Synthesis

This section presents our reduction from the exact quantum circuit synthesis problem to the MWMC problem. We first provide an overview of the chosen approach. Recalling Problem 4, the exact synthesis problem has the following components:

- **Input:** A quantum circuit $\mathcal{C}_1$ in gate set $\mathcal{G}_1$, or a unitary $U = U_{\mathcal{C}_1}$, and a finite target gate set $\mathcal{G}_2$.
- **Output:** A depth-optimal quantum circuit $\mathcal{C}_2$ in gate set $\mathcal{G}_2$, such that $U_{\mathcal{C}_2} \equiv U_{\mathcal{C}_1}$, if possible.

The problem can be expressed as exhaustively searching over all possible layers $\mathcal{D}_n$ for each layer in $\mathcal{C}_2$. To achieve depth-optimality, we increment the depth d until the following holds:

$$\exists D^0, \ldots, D^{d-1} \in \mathcal{D}_n \;:\; U_{\mathcal{C}_1} \equiv U_{\mathcal{C}_2} \text{ where } \mathcal{C}_2 = (D^0, \ldots, D^{d-1}). \tag{3}$$

In the remainder of this section, we first give the encoding of the input specification. Then we present different encodings for checking the exact equivalence between the input and output. Next, we encode a generic gate layer $\mathcal{D}_n$ and extend the encoding progressively to construct candidate circuits $\mathcal{C}_2$ of increasing depth, as specified in Equation 3. We conclude by showing how MWMC can find a depth-optimal output circuit $\mathcal{C}_2$ using the encoding for generic layers and equivalence checking.

Encoding input specifications. Due to the reversibility of quantum circuits, verifying the equivalence of circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ is reducible to checking if the circuit $\mathcal{C}_2 \cdot \mathcal{C}_1^\dagger$ is equivalent to the identity. Therefore, we encode the input circuit $\mathcal{C}_1$ as its inverse $\mathcal{C}_1^\dagger$, denoted as $F_{\mathcal{C}_1^\dagger}(\vec{q}, \vec{q}')$, as explained in Section 2.1. If the input is an n-qubit unitary operator $U^\dagger$, we can encode directly it as $F_{U^\dagger}(\vec{q}, \vec{q}')$ using weighted auxiliary variables to represent the unique components of the unitary, in the same way we encode individual gates [40].

Verifying exact equivalence. The exact equivalence checking problem is efficiently tackled with WMC in [39], using a so-called *linear encoding*, which requires $2n$ separate WMC calls. In Theorem 5, we extend this encoding to *cyclic encoding* and *linear-cyclic encoding*. These two new encodings solve the problem with a single call to the weighted model counter, *as we require in the proposed synthesis approach.*

▶ **Theorem 5.** *Let $\mathcal{C}$ be an n-qubit circuit, which is encoded by $F_\mathcal{C}$ with the corresponding weight function W. Then, the following four statements are equivalent to each other:*

- *$\mathcal{C} \equiv I^{\otimes n}$ ($\mathcal{C}$ is equivalent to the identity circuit $I^{\otimes n} = I_{2^n}$).*
- *Encoding the circuit $\mathcal{C}$ in* **PB**, *the* ***linear encoding*** *has weighted model count [39]:*

$$\#SAT_W(F_\mathcal{P}(\vec{q}) \wedge F_\mathcal{C}(\vec{q}, \vec{q}') \wedge F_\mathcal{P}(\vec{q}')) = 1 \text{ for all } \mathcal{P} \in \{X_j, Z_j \mid j \in [n]\}. \tag{4}$$

- *Encoding the circuit in either* **CB** *or* **PB**, *the* ***cyclic encoding*** *has weighted model count (this approach can be viewed as checking "overlap" with the identity $I^{\otimes n}$):*

$$\#SAT_W(F_\mathcal{C}(\vec{q}, \vec{q}') \wedge F_{I^{\otimes n}}(\vec{q}, \vec{q}')) = c \text{ with } |c| = 2^n \text{ for } \mathbf{CB} \text{ and } c = 4^n \text{ for } \mathbf{PB}. \tag{5}$$

- *Encoding the circuit $\mathcal{C}$ in* **PB**, *the* ***linear-cyclic encoding*** *has weighted model count:*

$$\#SAT_W(\bigvee_{\mathcal{P} \in \{X_j, Z_j \mid j \in [n]\}} F_\mathcal{P}(\vec{q}) \wedge F_\mathcal{C}(\vec{q}, \vec{q}') \wedge F_{I^{\otimes n}}(\vec{q}, \vec{q}')) = 2n. \tag{6}$$

where $\vec{q}, \vec{q}'$ are Boolean variables encoding the initial and final quantum state, respectively. Note that $F_{I^{\otimes n}}(\vec{q}, \vec{q}') = \bigwedge_{i \in [n]} (q_i \Leftrightarrow q_i')$, where $q_i \Leftrightarrow q_i'$ is shorthand for $(x_i \Leftrightarrow x_i') \wedge (z_i \Leftrightarrow z_i')$ in the Pauli basis.

We give a detailed proof in Appendix A.1, and illustrate the theorem in Example 6.

▶ **Example 6.** Consider two circuits $\mathcal{C}_1 = (S)$ and $\mathcal{C}_2 = (T, T)$. To check their equivalence, we first encode the circuits $\mathcal{C} = \mathcal{C}_2 \cdot \mathcal{C}_1^\dagger$ and I:

$$F_\mathcal{C}(q^0, q^3) := F_{S^\dagger}(q^0, q^1) \wedge F_T(q^1, q^2) \wedge F_T(q^2, q^3) \quad \text{and} \quad F_I(q, q') := q \Leftrightarrow q'$$

Then we check if $\mathcal{C} \equiv I$ as follows:

- Linear encoding: $\#SAT_W(F_\mathcal{P}(q^0) \wedge F_\mathcal{C}(q^0, q^3) \wedge F_\mathcal{P}(q^3)) = 1$ for $\mathcal{P} \in \{X, Z\}$ in **PB**.
- Cyclic encoding: $\#SAT_W(F_\mathcal{C}(q^0, q^3) \wedge F_I(q^0, q^3)) = c$ with $|c| = 2$ (**CB**) or $c = 4$ (**PB**).
- Linear-cyclic encoding: $\#SAT_W((F_X \vee F_Z) \wedge F_\mathcal{C} \wedge F_I) = 2$ in **PB**. ⌟

Encoding synthesis layers. Building up the output circuit is done by incrementally finding a sequence of gate layers implementing the input specification, as specified in Equation 3. To explore the space of possible gate layers, we encode the complete set $\mathcal{D}_n$ by introducing gate-selecting variables. Once these variables are fixed, they uniquely determine a specific gate layer within the set.

Given the target gate set $\mathcal{G}$, we first consider single-qubit gates $\mathcal{G}(1) \subseteq \mathcal{G}$. For each gate $G \in \mathcal{G}(1)$, we define the Boolean variable $p_{G,i}$ for each qubit $i \in [n]$, where $p_{G,i}$ is true if and only if G_i is included in the layer. Thus, we can encode the single-qubit gates of the layer as:

$$F_{\mathcal{G}(1)}(\vec{q}, \vec{q}\,', \vec{p}(1)) = \bigwedge_{i\in[n]} \bigwedge_{G\in\mathcal{G}(1)} (p_{G,i} \Rightarrow F_G(q_i, q_i')), \tag{7}$$

where the variables $\vec{q}$ and $\vec{q}\,'$ encode the states before and after the layer respectively, and $\vec{p}(1) = \{p_{G,i} \mid G \in \mathcal{G}(1), i \in [n]\}$ are the single-qubit gate-selecting variables.

Similarly, for a two-qubit gates $\mathcal{G}(2) \subseteq \mathcal{G}$, we introduce quadratically many variables for all combinations of the two qubits: $\vec{p}(2) = \{p_{G,i,j} \mid G \in \mathcal{G}(2), i, j \in [n], i \neq j\}$. Then encoding is:

$$F_{\mathcal{G}(2)}(\vec{q}, \vec{q}\,', \vec{p}(2)) = \bigwedge_{i,j\in[n], j\neq i} \bigwedge_{G\in\mathcal{G}(2)} (p_{G,i,j} \Rightarrow F_{G_{i,j}}(\vec{q}, \vec{q}\,')), \tag{8}$$

Since we consider the universal gate set Clifford+T, which only includes single-qubit gates and two-qubit gates, the set of gate-selecting variables is given by $\vec{p} = \vec{p}(1) \cup \vec{p}(2)$ of the synthesis layer. It is worth noting that the identity operator I is included when considering multi-qubit circuits for completeness. For example when considering $\{CX, H, T\}$, we define the set of single-qubit gates as $G(1) = \{H, T, I\}$.

A layer is valid only when applying exactly one gate to each qubit. For example, $p_{H,1}$ and $p_{T,1}$ should not be true simultaneously. Therefore, we define $\texttt{EXO}(V) = \bigvee_{v\in V} v \wedge \bigwedge_{u,v\in V, u\neq v}(\overline{v} \vee \overline{u})$, a constraint ensuring that exactly one variable in the set V is true. Now, for each qubit $i \in [n]$, we apply:

$$F_{\texttt{EXO}}(\vec{p}) = \bigwedge_{i\in[n]} \texttt{EXO}(\vec{p}_i), \text{ where } \vec{p}_i = \{p_{G,i} \mid G \in \mathcal{G}(1)\} \cup \{p_{G,i,j}, p_{G,j,i} \mid G \in \mathcal{G}(2), j \in [n]\}. \tag{9}$$

Combining the three constraints in Equation 7, Equation 8 and Equation 9 gives us the layer encoding:

$$F_{\mathcal{D},\mathcal{G}}(\vec{q}, \vec{q}\,', \vec{p}) = F_{\mathcal{G}(1)}(\vec{q}, \vec{q}\,', \vec{p}(1)) \wedge F_{\mathcal{G}(2)}(\vec{q}, \vec{q}\,', \vec{p}(2)) \wedge F_{\texttt{EXO}}(\vec{p}) \tag{10}$$

If there are gates in the target gate set $\mathcal{G}$ applied to more than two qubits, the encoding is expanded accordingly.

▶ **Example 7.** The encoding of a layer in the universal gate set $\mathcal{G} = \{CX, H, T\}$ is as follows:

$$\begin{aligned} F_{\mathcal{D},\mathcal{G}}(\vec{q}, \vec{q}\,', \vec{p}) = & \bigwedge_{i\in[n]} \big(p_{H,i} \Rightarrow F_H(q_i, q_i') \wedge p_{T,i} \Rightarrow F_T(q_i, q_i') \wedge p_{I,i} \Rightarrow F_I(q_i, q_i')\big) \\ & \wedge \bigwedge_{i,j\in[n], j\neq i} \big(p_{CX,i,j} \Rightarrow F_{CX_{i,j}}((q_i, q_j), (q_i', q_j'))\big) \quad \wedge \quad \bigwedge_{i\in[n]} \texttt{EXO}(\vec{p}_i), \end{aligned}$$

where $\vec{p} = \bigcup_{i\in[n]} \vec{p}_i$ and $\vec{p}_i = \{p_{I,i}, p_{H,i}, p_{T,i}\} \cup \bigcup_{j\in[n], j\neq i}\{p_{CX,i,j}, p_{CX,j,i}\}$.

To encode multiple layers, we reserve state variables $\vec{q}^t$ and gate-selecting variables $\vec{p}^t$ for the t-th layer and substitute the above formula with the variables for each time step, i.e. $F_{\mathcal{D},\mathcal{G}}(\vec{q}^t, \vec{q}^{t+1}, \vec{p}^t)$. Encoding d layers is given by a conjunction of the encodings:

$$\bigwedge_{t\in[d]} F_{\mathcal{D},\mathcal{G}}(\vec{q}^t, \vec{q}^{t+1}, \vec{p}^t). \tag{11}$$

Encoding exact synthesis. After giving the encoding of the input specification $U_{\mathcal{C}_1}$, the encoding of exact equivalence checking, and the encoding of gate layers, we now show the encoding of exact synthesis. The main idea is to determine a minimal sequence of gate layers $\mathcal{C}_2 = (D^1, ..., D^d)$ such that $U_{\mathcal{C}_2} \equiv U_{\mathcal{C}_1}$, which can be represented with the cyclic encoding as:

$$Syn_{\mathbf{C},\mathbf{B},\mathcal{G},\mathcal{C}_1,d}(P, Q) \quad = \quad \overbrace{F_{U^\dagger_{\mathcal{C}_1}}(\vec{q}, \vec{q}^0) \wedge \bigwedge_{t\in[d]} F_{\mathcal{D},\mathcal{G}}(\vec{q}^t, \vec{q}^{t+1}, \vec{p}^t)}^{U_{\mathcal{C}_2}U^\dagger_{\mathcal{C}_1}} \;\wedge\; \overbrace{(\vec{q} \Leftrightarrow \vec{q}^d)}^{F_{I^{\otimes n}}}, \tag{12}$$

where $\mathbf{C}$ denotes the encoding is cyclic, $\mathbf{B}$ denotes the chosen basis (which can be either $\mathbf{PB}$ or $\mathbf{CB}$), $P = \bigcup_{t\in[d]} \vec{p}^t$ is the set of gate-selecting variables, and $Q = \vec{q} \cup \bigcup_{t\in[d+1]} \vec{q}^t \cup \vec{u}^t$ is the set of all state variables $\vec{q}$ and auxiliary variables $\vec{u}$. For a linear-cyclic checking (denoted by $\mathbf{LC}$)(only for $\mathbf{PB}$), the encoding is done by adding constraints to the initial state variables

$$Syn_{\mathbf{LC},\mathbf{PB},\mathcal{G},\mathcal{C}_1,d}(P, Q) = \bigvee_{\mathcal{P}\in\{X_j, Z_j | j\in[n]\}} F_{\mathcal{P}}(\vec{q}) \wedge Syn_{\mathbf{C},\mathbf{PB},\mathcal{G},\mathcal{C}_1,d}(P, Q). \tag{13}$$

Proposition 8 shows how the exact synthesis problem is reduced to the MWMC problem. It essentially reduces the problem of finding an assignment for the gate-selecting variables that maximizes the weighted model count of the above encodings in Equation 12 and Equation 13. The correctness of Proposition 8 follows directly from the cyclic and linear-cyclic encodings of Theorem 5, which concludes this section.

▶ **Proposition 8.** *Given a quantum circuit $\mathcal{C}_1$ (or its unitary $U_{\mathcal{C}_1}$) and an integer d, there exists a d-depth circuit $\mathcal{C}_2$ such that $U_{\mathcal{C}_1} \equiv U_{\mathcal{C}_2}$ iff*

$$Max\#SAT_W\big(Syn_{\mathbf{E},\mathbf{B},\mathcal{G},\mathcal{C}_1,d}(P, Q)\big) = (c, \tau(P)),$$

where for $\mathbf{E} = \mathbf{LC}$ (linear-cyclic encoding) and $\mathbf{B} = \mathbf{PB}$ we have that $c = 2n$, and for $\mathbf{E} = \mathbf{C}$ (cyclic encoding) if $\mathbf{B} = \mathbf{PB}$ we have that $c = 4^n$ and if $\mathbf{B} = \mathbf{CB}$ we have that $|c| = 2^n$. When the maximal value is achieved, the output circuit $\mathcal{C}_2$ can be directly determined from the satisfying assignment $\tau(P)$.

▶ **Example 9.** Let us consider the circuit $\mathcal{C}_1 = (S)$ and the target gate set $\mathcal{G} = \{CX, H, T\}$. Since it is a single-qubit circuit, the encoding of one general layer is, where for gate G we denote $F_G = F_G(q_0, q'_0)$

$$F_{\mathcal{D},\mathcal{G}}(\vec{q} = (q_0), \vec{q}' = (q'_0), \vec{p}) = \big((p_{H,0} \Rightarrow F_H) \wedge (p_{T,0} \Rightarrow F_T) \wedge (p_{I,0} \Rightarrow F_I)\big) \;\wedge\; \bigwedge_{i\in[n]} \mathtt{EXO}(\vec{p}),$$

We synthesize the circuit by calling $Max\#SAT_W$ to find an assignment for the gate-selection variables such that the function achieves maximal value:

$$Max\#SAT_W(Syn_{\mathbf{LC},\mathbf{PB},\mathcal{G},\mathcal{C}_1,1}(P, Q)) = (0.854,\ \tau_1 = \{p^0_{H,0} \leftarrow 0, p^0_{T,0} \leftarrow 1, p^0_{I,0} \leftarrow 0\})$$
$$Max\#SAT_W(Syn_{\mathbf{LC},\mathbf{PB},\mathcal{G},\mathcal{C}_1,2}(P, Q)) = (1,\ \tau_2 = \{p^0_{H,0} \leftarrow 0, p^0_{T,0} \leftarrow 1, p^0_{I,0} \leftarrow 0,$$
$$p^1_{H,0} \leftarrow 0, p^1_{T,0} \leftarrow 1, p^1_{I,0} \leftarrow 0\})$$

From the assignment τ_2, we get the synthesized circuit $\mathcal{C}_2 = (T, T)$.

5 Approximate Quantum Circuit Synthesis

In this section, we focus on approximate synthesis as given in Problem 4. The input and output of the problem are now as follows:

- **Input:** A quantum circuit $\mathcal{C}_1$ in a gate set $\mathcal{G}_1$ or directly the unitary $U_{\mathcal{C}_1}$, a finite target gate set $\mathcal{G}_2$, and an error bound $\epsilon \in (0, 1]$.
- **Output:** A depth-optimal quantum circuit $\mathcal{C}_2$ in gate set $\mathcal{G}_2$ such that $\mathcal{C}_2 \simeq_\epsilon \mathcal{C}_1$.

Since synthesis relies on equivalence checking, our first aim is to lift the latter to approximate equivalence checking. Theorem 10 shows that the cyclic encoding in both **PB** and **CB** computes the Jamiołkowski fidelity between two circuits, thus both encodings can determine approximate equivalence checking based on Definition 3.

▶ **Theorem 10.** *Given two unitary matrices U and V on an n-qubit Hilbert space $\mathcal{H}^{\otimes n}$, the Jamiołkowski fidelity can be computed by*

$$\mathrm{Fid}_J(U,V) \quad = \quad \begin{cases} \frac{1}{4^n} \cdot \#SAT_W(F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge (\vec{q} \Leftrightarrow \vec{q}\,')) & \textit{in} \quad \mathbf{PB} \\ \frac{1}{4^n} \cdot |\#SAT_W(F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge (\vec{q} \Leftrightarrow \vec{q}\,'))|^2 & \textit{in} \quad \mathbf{CB} \end{cases}.$$

Proof. To compute Jamiołkowski fidelity of two given n-qubit circuits U and V on $\mathcal{H}^{2^n}$, one takes the maximally entangled state $|\Psi_n\rangle$ on $\mathcal{H}^{2^n} \otimes \mathcal{H}^{2^n}$ as input and compute the fidelity between the output states $(U \otimes I^{\otimes n})|\Psi_n\rangle$ and $(V \otimes I^{\otimes n})|\Psi_n\rangle$. We first prove this in the **CB** and then move to **PB**.

In **CB**, the Jamiołkowski fidelity is given by

$$\mathrm{Fid}_J(U,V) = \mathrm{Fid}((U \otimes I^{\otimes n})|\Psi_n\rangle, (V \otimes I^{\otimes n})|\Psi_n\rangle) = \left|\langle\Psi_n|(U^\dagger \otimes I^{\otimes n})(V \otimes I^{\otimes n})|\Psi_n\rangle\right|^2$$
$$= \left|\frac{1}{2^n} \cdot \sum_{b\in\{0,1\}^n} \sum_{b'\in\{0,1\}^n} \langle i|U^\dagger V|j\rangle \cdot \langle b|I^{\otimes n}|b'\rangle\right|^2 = \left|\frac{1}{2^n} \cdot \sum_{b\in\{0,1\}^n} \langle b|U^\dagger V|b\rangle\right|^2,$$

Based on Lemma 2, it holds that:

$$\#SAT_W(F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge (\vec{q} \Leftrightarrow \vec{q}\,')) = \sum_{b\in\{0,1\}^n} \#SAT_W(F_{|b\rangle}(\vec{q}) \wedge F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge F_{|b\rangle}(\vec{q}\,'))$$
$$= \sum_{b\in\{0,1\}^n} \langle b|U^\dagger V|b\rangle,$$

giving us that $\mathrm{Fid}_J(U,V) = \frac{1}{4^n}\,|\#SAT_W(F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge (\vec{q} \Leftrightarrow \vec{q}\,'))|^2$ as we wanted to show.

Moving to **PB**, we first represent $|\Psi_n\rangle$ and the Jamiołkowski fidelity in the Pauli basis, and then explain the encoding. The density operator of the maximally entangled state can be decomposed in the Pauli basis as: $|\Psi\rangle\langle\Psi| = \frac{1}{2^n}\sum_{i\in[n],j\in[n]} |ii\rangle\langle jj| = \frac{1}{4^n}\sum_{\mathcal{P}_i\in\{X,Y,Z,I\}^{\otimes n}} \mathcal{P}_i \otimes \mathcal{P}_i^T$, which is shown in [15]. The Jamiołkowski fidelity can also be defined in the Pauli basis as:

$$\mathrm{Fid}_J(U^\dagger V, I) = \mathrm{Tr}\big((U^\dagger V \otimes I)|\Psi\rangle\langle\Psi|(V^\dagger U \otimes I)|\Psi\rangle\langle\Psi|\big)$$
$$= \frac{1}{16^n}\sum_{j,k\in[4^n]} \mathrm{Tr}\big(U^\dagger V\mathcal{P}_j V^\dagger U\mathcal{P}_k \otimes \mathcal{P}_j^T\mathcal{P}_k^T\big) = \frac{1}{8^n}\sum_{j\in[4^n]} \mathrm{Tr}\big(U^\dagger V\mathcal{P}_j V^\dagger U\mathcal{P}_j\big) \tag{14}$$

Based on Proposition 1 in [39], each of the summand $\frac{1}{2^n}\cdot\mathrm{Tr}\big(U^\dagger V\mathcal{P}_j V^\dagger U\mathcal{P}_j\big)$ can be computed by the weighted model counting of $F_{\mathcal{P}_j}(\vec{q}) \wedge F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge F_{\mathcal{P}_j}(\vec{q}\,')$, or equally, $F_{\mathcal{P}_j}(\vec{q}) \wedge F_{U^\dagger V}(\vec{q},\vec{q}\,') \wedge (\vec{q} \Leftrightarrow \vec{q}\,')$. To compute the summation, one should go over all possible Pauli strings, which is equal to setting the variables in $\vec{q}$ to be free. Hence, the Jamiolkowski fidelity is obtained by $\mathrm{Fid}_J(U,V) = \frac{1}{4^n}\#SAT_W(F_{U^\dagger V}(\vec{q}^{\,0},\vec{q}^{\,m}) \wedge (\vec{q}^{\,0} \Leftrightarrow \vec{q}^{\,m}))$. ◀

We can thus reuse the cyclic encoding from Equation 12 to encode approximate synthesis.

▶ **Proposition 11.** *Given a quantum circuit $\mathcal{C}_1$, an integer d and an error bound $\epsilon \in (0, 1]$, there exists a d-depth circuit $\mathcal{C}_2$ such that $U_{\mathcal{C}_2} \simeq_\epsilon U_{\mathcal{C}_1}$ iff $Max\#SAT_W\big(Syn_{\mathbf{C},\mathbf{B},\mathcal{G},\mathcal{C},d}(P, Q)\big) = (c, \tau(P))$, such that $\frac{1}{2^n}|c| > 1 - \epsilon$ if $\boldsymbol{B} = \mathbf{CB}$ and $\frac{1}{4^n}c > 1 - \epsilon$ if $\boldsymbol{B} = \mathbf{PB}$. In that case, $\mathcal{C}_2$ can be determined by the satisfying assignment $\tau(P)$.*

Like in exact synthesis, to get the depth-optimal ϵ-approximate circuit, we apply the above Proposition 11 for an increasing depth d. Example 12 demonstrates approximate synthesis.

▶ **Example 12.** Consider the circuit $\mathcal{C}_1 = (R_Z(\frac{\pi}{8}))$ with $\epsilon = 0.05$. Synthesizing one gate layer, we get the result as

$$Max\#SAT_W(Syn_{\mathbf{B},\mathbf{PB},\mathcal{G},\mathcal{C},1}(P, Q)) = (3.848, \{p^0_{H,0} \leftarrow 0, p^0_{T,0} \leftarrow 1, p^0_{I,0} \leftarrow 0\}),$$

which determine the output circuit as $\mathcal{C}_2 = (T)$ and the fidelity $\mathrm{Fid}_J(U_{\mathcal{C}_1}, U_{\mathcal{C}_2}) = \frac{1}{4} \times 3.848 = 0.962 > 1 - \epsilon$. Thus, the synthesis procedure stops, and the output circuit is (T).

6 Related Work

Clifford circuit synthesis. Synthesis of Clifford circuits is substantially simpler than that of universal quantum circuits due to the ability to exploit the algebraic structure of the symplectic group, which significantly constrains the search space. Any Clifford operation on n qubits can be efficiently represented by a $2n \times 2n$ symplectic matrix over $\mathbb{F}_2$, rather than the exponentially larger $2^n \times 2^n$ unitary matrix typically required for general quantum operations [14]. Building on this, Maslov and Roetteler [36] employ the Bruhat decomposition of the symplectic group to generate shorter Clifford circuits. Similarly, Rengaswamy et al. [46] develop Clifford synthesis algorithms via symplectic geometry, targeting logical-level Clifford operations with an emphasis on practical implementations on physical qubits. While these approaches produce efficient Clifford circuits, they do not guarantee optimality in terms of depth or gate count. To address this limitation, Schneider et al. [48] reformulate Clifford synthesis as a satisfiability problem. This encoding enables the use of SAT and MaxSAT solvers [10] to identify optimal Clifford circuits for a fixed depth, offering a rigorous method to achieve minimality.

Exact Clifford+T circuit synthesis. In error-corrected quantum computing, the relevant universal gate is Clifford+T. There are many works considering the exact synthesis of quantum circuits in the gate set Clifford+T[3, 37, 22, 23, 29, 43], i.e., the desired specification is realized without any rounding errors. Approaches like [22, 43] synthesize the unitary matrix representing the specification in a local fashion, i.e., column by column. However, they do not give the optimal solution and leave significant room for improvement. To achieve optimality, the meet-in-the-middle algorithm [37, 23] performs an exhaustive search over the space of all Clifford+T circuits up to a given depth.

Approximate Clifford+T circuit synthesis. Since not all unitaries can be implemented exactly in the Clifford+T gate set, other works have focused on synthesizing circuits under different approximation metrics. Most of these works [30, 47, 49, 19] focus on single-qubit operators, especially rotation gates, while [21, 44] consider multi-qubit operators.

7 Experimental Evaluation

7.1 Implementation

To evaluate and test our proposed method, we implemented it in a tool called `Quokka#-syn`, as part of the open source toolkit `Quokka#`[*]. In addition to the core method described in the previous sections, we also implemented several optimization rules. Since no existing tool can solve our problem, we extended the MWMC solver d4Max [5] with support for negative and complex weights. The following paragraphs describe these two components in more detail.

Encoding optimization. While the encoding in Equation 11 allows for any circuit of depth d, many encoded circuits are redundant as they fall in the same equivalence class. For example, two consecutive Hadamard gates can be reduced to the identity. So if the H is applied to the j-th qubit at depth t, we can safely exclude the case where another H is applied to the same qubit at depth $t+1$. In the encoding, this corresponds to enforcing that if $p_{H_{j,t}}$ is set to 1, then $p_{H_{j,t+1}}$ must be 0, which can be encoded as $\bigwedge_{k\in[d-1]} \bigwedge_{i\in[n]} (\overline{p}^{k}_{H,i} \vee \overline{p}^{k+1}_{H,i})$. We apply similar reasoning to patterns like $T^8 = I$ and $CX_{i,j}CX_{i,j} = I^{\otimes 2}$, which are never part of an optimal circuit. Eliminating such cases does not affect optimality and helps reduce the search space. We introduce additional encodings to prune such redundant structures, as detailed in Appendix A.2.

d4Max extension. To support the encodings, we first extended d4Max to support negative weights (for **PB**) and complex weights (for **CB**). For number representation, we use arbitrary precision arithmetic from the GMP library, as in the original version of d4Max. Because of the negative weights, computing an upper bound for a sub-formula becomes more complex. In the original version of d4Max, upper bounds can be easily computed by assuming the formula is a tautology. This allows the solver to prune branches of the search tree when it is clear that no better value can be reached than the current best solution. As this is non-trivial to resolve in the presence of negative weights, we turned this optimization off. Another removed feature is the ability to compute intermediate approximations before processing all connected components. As a result of these changes, the current version is slower at providing intermediate solutions compared to the original version.

7.2 Performance

In this section, we explore the performance of our method as implemented in `Quokka#-syn`. Our implementation synthesizes circuits with the gate set $\{H, CX, T, T^\dagger\}$. This set is also universal since $S = TT$, where we replace the S gate with the $T^\dagger$ gate.

We demonstrate the feasibility and scalability of our method based on two classes of benchmarks for exact synthesis: random circuits on $\{H, CX, T, T^\dagger\}$ and commonly used unitary operators such as the CH and *Toffoli* gates. We use randomly generated circuits as benchmarks, as they are the standard method for evaluating performance since they represent a hard case, which is also employed in practice to prove quantum supremacy [4].

Since our method guarantees a depth-optimal result, we compare with the state-of-the-art method `mitms` [3], which targets the same task. Our experiments were run on a single-core AMD Ryzen 9 7900X Processor and 64 GB of memory.

[*]`https://github.com/System-Verification-Lab/Quokka-Sharp`

Table 3 Synthesis benchmarks for random circuits. **CB** and **PB** indicate cyclic encoding in each basis; **LPB** uses linear-cyclic encoding in **PB**. Memory is reported by d4max; for short runtimes, usage is below 2 GB and denoted by < 2. Averages include $\pm$ standard deviation. **-** indicates all 10 samples failed; $\circ$ marks untested depths due to low prior success.

#Qb	Depth	Rate			Time (s)			Memory (GB)		
		CB	**LPB**	**PB**	**CB**	**LPB**	**PB**	**CB**	**LPB**	**PB**
2	1	1.0	1.0	1.0	0.05 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	< 2	< 2	< 2
	2	1.0	1.0	1.0	0.07 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	< 2	< 2	< 2
	3	1.0	1.0	1.0	0.11 ± 0.01	0.11 ± 0.01	0.14 ± 0.02	< 2	< 2	< 2
	4	1.0	1.0	1.0	0.29 ± 0.09	0.22 ± 0.07	0.40 ± 0.10	< 2	< 2	< 2
	5	1.0	1.0	1.0	2.46 ± 1.51	1.46 ± 0.72	4.53 ± 3.16	2.13 ± 0.03	2.09 ± 0.00	2.11 ± 0.02
	6	1.0	1.0	1.0	21.25 ± 12.54	14.26 ± 10.33	55.69 ± 42.54	2.46 ± 0.28	2.16 ± 0.07	2.38 ± 0.27
	7	0.9	0.9	0.3	115.31 ± 64.99	97.28 ± 46.98	206.37 ± 17.36	3.84 ± 1.31	2.48 ± 0.21	3.41 ± 0.54
	8	0.0	0.0	$\circ$	-	-	$\circ$	-	-	$\circ$
3	1	1.0	1.0	1.0	0.05 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	< 2	< 2	< 2
	2	1.0	1.0	1.0	0.08 ± 0.01	0.10 ± 0.01	0.10 ± 0.02	< 2	< 2	< 2
	3	1.0	1.0	1.0	1.43 ± 0.97	0.44 ± 0.24	2.45 ± 1.91	2.11 ± 0.02	< 2	2.11 ± 0.01
	4	1.0	1.0	1.0	51.56 ± 57.11	11.26 ± 4.95	105.40 ± 64.98	3.15 ± 1.25	2.15 ± 0.04	2.87 ± 0.67
	5	0.1	0.7	0.0	145.96	147.37 ± 49.30	-	5.08	3.20 ± 0.53	-
	6	$\circ$	0.0	$\circ$	$\circ$	-	$\circ$	$\circ$	-	$\circ$
4	1	1.0	1.0	1.0	0.05 ± 0.00	0.06 ± 0.00	0.05 ± 0.00	< 2	< 2	< 2
	2	1.0	1.0	1.0	0.37 ± 0.20	0.30 ± 0.14	0.62 ± 0.48	2.09	< 2	2.09 ± 0.01
	3	1.0	1.0	0.5	102.55 ± 85.73	22.65 ± 16.03	85.70 ± 42.76	4.44 ± 2.07	2.26 ± 0.13	2.69 ± 0.34
	4	0.0	0.2	0.0	-	246.80 ± 21.38	-	-	4.51 ± 0.20	-
5	1	1.0	1.0	1.0	0.05 ± 0.00	0.06 ± 0.01	0.06 ± 0.00	< 2	< 2	< 2
	2	1.0	1.0	1.0	2.93 ± 2.17	3.01 ± 1.62	11.85 ± 11.15	2.14 ± 0.04	2.11 ± 0.02	2.18 ± 0.10
	3	0.0	0.1	0.0	-	243.71	-	-	4.67	-
6	1	1.0	1.0	1.0	0.05 ± 0.00	0.12 ± 0.06	0.06 ± 0.00	< 2	< 2	< 2
	2	0.9	1.0	0.7	70.49 ± 82.13	54.47 ± 44.75	72.71 ± 69.85	3.74 ± 2.03	2.69 ± 0.61	2.80 ± 0.81
	3	0.0	0.0	0.0	-	-	-	-	-	-

Exact synthesis. In the first experiment, we compare the performances of the different encodings to determine the best one and assess the scalability of the method, as this determines its effectiveness in practice [3]. We test all three synthesis encodings for circuit inputs: **CB** with the cyclic encoding and **PB** with the cyclic and linear-cyclic encodings.

Random circuits are generated layer by layer, with each layer applying exactly one gate per qubit. Gates are selected uniformly at random from the gate set and assigned to unassigned qubits. Since a random circuit of depth d could be equivalent to a shallower circuit, we retain only those circuits that cannot be synthesized with a depth less than d, until we obtain 10 samples. Using these samples, we then evaluated all three encodings for increasing depths, stopping when fewer than half the circuits are solved within the time limit per instance. We set the time limit to 300 seconds to accommodate a large number of benchmarks. Experiments were run on circuits with 2 to 6 qubits; the 1-qubit case is omitted here as it is evaluated for approximate synthesis below.

The results are shown in Table 3. For each encoding, qubit count, and depth, we report the success rate (the fraction of random circuit samples that can be solved within the time limit), the average runtime of solved cases, and the average memory needed by d4Max for the solved cases. The results highlight the significant impact of the qubit count and depth on the runtime. For every number of qubits, there is a specific depth threshold where problem-solving becomes challenging; before reaching this point, the success rate is perfect. We observe that the linear encoding consistently demonstrates the best performance among the three encoding methods.

Table 4 Experimental results on synthesis for controlled-gates. We present the running time and memory usage reported by d4Max, as the encoding time is less than 0.01 seconds. We report N/A when d4Max does not give the memory usage, denoted by < 2000. Here '-' denotes timeout cases except for the case `mitms` with 2-qubit and 8-depth, where the data is not given in [3].

#Qubits \Depth				1	2	3	4	5	6	7	8
2	`Quokka#-syn`	Cyclic (unitary)	Time (s)	0.023	0.025	0.059	0.357	3.890	29.861	436.450	-
			Mem (MB)	< 2000	< 2000	< 2000	< 2000	2200.990	2767.230	10666.570	-
		Linear-cyclic	Time (s)	0.023	0.028	0.053	0.270	2.337	6.470	197.810	2353.696
			Mem (MB)	< 2000	< 2000	< 2000	< 2000	2148.300	2183.760	4814.550	16091.660
	`mitms`		Time (s)	0.002	0.019	0.188	0.248	12.433	32.766	-	-
			Mem (MB)	0.002	0.016	0.147	1.013	6.249	84.622	-	-
3	`Quokka#-syn`	Cyclic (unitary)	Time (s)	0.024	0.106	3.310	177.220	-	-	-	-
			Mem (MB)	< 2000	< 2000	2188.920	5398.730	-	-	-	-
		Linear-cyclic	Time (s)	0.029	0.140	3.413	76.293	3.066.258	-	-	-
			Mem (MB)	< 2000	< 2000	2148.740	2688.360	18860.120	-	-	-
	`mitms`		Time (s)	0.027	1.409	53.2	2311,023	-	-	-	-
			Mem (MB)	5.633	0.179	6.737	215.970	-	-	-	-

The second experiment compares the performance of `Quokka#-syn` with both unitary and circuit inputs to that of `mitms`, a tool that performs the same task, only with target gate set $\{H, CNOT, S, S^\dagger, T, T^\dagger\}$. For comparison, we selected the 2-qubit CH gate and the 3-qubit *Toffoli* gate. We encode the gates once as unitary matrices in **CB**, as described in Section 4, and once as Clifford+T circuits, as described in [42], in **PB**. We choose to use **CB** encoding for unitary inputs, since even for sparse unitaries, such as the Toffoli gate and its generalizations, the encoding of the Pauli decomposition may blow up, as shown in [39]. For the circuit encoding, we use the *linear-cyclic encoding* in **PB**, as it showed the best performance in the previous experiment. For both encodings, the timeout is set to one hour (including the encoding time and calling of d4Max for all depths). Since we failed to compile `mitms` within our experimental setup due to its reliance on older libraries, we instead refer to the performance data reported in [3] to provide a comparative reference. The results are shown in Table 4. While the comparison is not entirely fair – due to differences in platform, target gate set, and benchmark design – we observe that `Quokka#-syn` tends to exhibit higher memory usage but lower runtime compared to `mitms`.

It is also worth noting that the performance of **PB** with the linear-cyclic encoding given in Table 3 are better than Table 4 in many cases, for example, 3-qubit and 5-depth cases in Table 3 feature 70% success rate with average runtime of 147.37 seconds, while in Table 4, it takes 3066.26 seconds. One explanation is that, since the maximal weight is known, as shown in Theorem 5, `Quokka#-syn` provides a threshold to the solver, allowing it to terminate as soon as the optimal value is achieved. Consequently, if a circuit can be successfully synthesized within the given depth, as with the solved cases in Table 3, the tool completes early. In contrast, for unsuccessful cases, such as for each depth in Theorem 5, the solver must explore the entire search space, resulting in longer run times. In addition, we observe substantial variability in the run times, indicating that the performance is highly dependent on the specific characteristics of each case.

Approximate synthesis. Important gates that often need to be synthesized approximately in Clifford+T are the general rotation gates: R_x, R_y, R_z [30]. Our method supports a general rotation angle in circuits, i.e., the rotation angle can be any real number. The encoding will be the same Boolean formula, but with a weight function dependent on the rotation angle [38]. Thus, different rotation angles do not significantly affect the performance per depth of the approximate synthesis of the rotation gate R_z. Hence, to demonstrate the use of approximate synthesis, we consider the quantum gate $R_z(\frac{\pi}{8}) = \begin{bmatrix} e^{-i\pi/16} & 0 \\ 0 & e^{i\pi/16} \end{bmatrix}$. We

Table 5 $R_Z(\pi/8)$ synthesis using **CB** with cyclic encoding. Statistics are shown per synthesis layer. Memory usage under short runtimes is omitted (< 2). Only fidelity-improving iterations are shown. At 24 layers, the program crashed due to resource limits (**-**).

depth	1	10	15	24
#variables	11	92	137	218
#clauses	37	382	577	928
#literals	90	960	1460	2360
#Selecting variables	4	40	60	96
fidelity	0.962	0.975	0.997	**-**
Time (s)	0.021	0.092	2.685	>1560.51
mem (GB)	<2	<2	2.2	>33.31

choose to use the **CB** with the *cyclic encoding* as it outperforms the **PB** with the *cyclic encoding*, according to the results in Table 3 (Recall that **PB** with the *linear-cyclic encoding* cannot perform *approximate* synthesis). We report statistics for each synthesis layer where an improvement in the achieved fidelity is observed. The results are presented in Table 5. The corresponding output circuits are as follows.

- **1 Layer:** with $\epsilon = 0.1$, $\mathcal{C}' = (T)$
- **10 Layer:** with $\epsilon = 0.1$, $\mathcal{C}' = (T^\dagger, H, T^\dagger, H, T^\dagger, H, T^\dagger, H, T^\dagger, H)$
- **15 Layer:** with $\epsilon = 0.01$, $\mathcal{C}' = (H, T^\dagger, H, T, H, T, H, T^\dagger, H, T, H, T, H, T^\dagger, H)$

Other tools also target approximate synthesis. For example, the optimal tool `mitms` and other non-optimal tools such as `gridsynth`[47] use operator norm as a metric, while [41] uses the Jamiołkowski fidelity, though the implementation of the latter one is not open-source. We do not include a performance comparison because, to the best of our knowledge, no available tool performs optimal approximate synthesis with fidelity as a metric.

8 Conclusion

The results presented in this work demonstrate that maximum weighted model counting can be effectively employed for both exact and approximate quantum circuit synthesis. This approach benefits from the generality and extensibility of weighted model counting techniques as used in circuit simulation and equivalence checking, particularly their compatibility with diverse gate sets and representation bases. However, in its current form, the maximum weighted model counting exhibits limited performance in this new context.

There are, nonetheless, several promising directions for further development: (1) The current implementation builds upon a prototype version of `d4Max`, which lacks several optimizations to accommodate the extension to negative and complex weights. The design of dedicated algorithms tailored to this application remains an open avenue for future research. (2) Our layer-by-layer synthesis approach could benefit from incremental solving, as used in bounded model checking [9, 24]. Although incremental approaches have been studied for sampling and weighted model counting [54, 55], their application to maximum weighted model counting has not yet been explored. (3) Since the inception of the Model Counting Competition,[†] continuous progress has been observed in the capabilities of model counters. By contributing our benchmarks to the community, we anticipate further methodological

[†] `https://mccompetition.org/`

improvements that may enhance the reliability and performance of this approach. (4) The particular characteristics in the encoded CNF, such as an abundance of XOR clauses [40], are not yet exploited by the solver. This could lead to significant performance gains [32]. (5) Lastly, incorporating symmetry considerations may further improve the efficiency of model counting, as has been demonstrated in related domains [7].

The depth-optimal synthesis presented here enables more efficient implementations of multi-qubit gates, a critical requirement in fault-tolerant and error-corrected quantum computing [27]. Future work will also address minimizing the T count, which remains an important objective in error-corrected architectures [30]. While the current study focuses on the Clifford+T gate set, the underlying encoding is readily generalizable to other gate sets. Moreover, extending maximum weighted model counting to the stochastic SAT setting [35, 34] could broaden its applicability to further quantum circuit optimization problems. This line of work may also lead to the derivation of novel lower bounds for computationally hard problems in quantum circuit analysis, based on established results from the reasoning and satisfiability domains [6].

References

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

2 Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Inf. Comput.*, 6(2):97–165, March 2006. `doi:10.26421/QIC6.2-1`.

3 Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 32(6):818–830, 2013. `doi:10.1109/TCAD.2013.2244643`.

4 Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. `doi:10.1038/s41586-019-1666-5`.

5 Gilles Audemard, Jean-Marie Lagniez, and Marie Miceli. A new exact solver for (weighted) max#sat. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:20, Haifa, Israel, August 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.SAT.2022.28`.

6 Max Bannach and Markus Hecher. On weighted maximum model counting: complexity and fragments. In *36th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2024, Herndon, VA, USA, October 28-30, 2024*, pages 1–9. IEEE, 2024. `doi:10.1109/ICTAI62512.2024.00010`.

7 Anicet Bart, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. Symmetry-driven decision diagrams for knowledge compilation. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 51–56. IOS Press, 2014. `doi:10.3233/978-1-61499-419-0-51`.

8 Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 11–20. ACM, 1993. `doi:10.1145/167088.167097`.

9 Armin Biere. Bounded model checking. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 739–764. IOS Press, 2021. `doi:10.3233/FAIA201002`.

10 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

11 Sergey Bravyi, Dan E. Browne, Padraic Calpin, Earl T. Campbell, David Gosset, and Mark Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3:181, September 2019. `doi:10.22331/q-2019-09-02-181`.

12 Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Phys. Rev. X*, 6:021043, June 2016. `doi:10.1103/PhysRevX.6.021043`.

13 Lukas Burgholzer and Robert Wille. Advanced equivalence checking for quantum circuits. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(9):1810–1824, September 2020. `doi:10.1109/tcad.2020.3032630`.

14 A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction and orthogonal geometry. *Phys. Rev. Lett.*, 78:405–408, January 1997. `doi:10.1103/PhysRevLett.78.405`.

15 Senrui Chen, Sisi Zhou, Alireza Seif, and Liang Jiang. Quantum advantages for pauli channel estimation. *Physical Review A*, 2021. URL: `https://api.semanticscholar.org/CorpusID:237213441`.

16 Yu-Fang Chen, Kai-Min Chung, Ondrej Lengál, Jyun-Ao Lin, Wei-Lun Tsai, and Di-De Yen. An automata-based framework for verification and bug hunting in quantum circuits. *Proc. ACM Program. Lang.*, 7(PLDI):1218–1243, June 2023. `doi:10.1145/3591270`.

17 Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm. *Quantum Inf. Comput.*, 6(1):81–95, 2006. `doi:10.26421/QIC6.1-6`.

18 M. Van den Nest. Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond. *arXiv:0811.0898*, 2008. `doi:10.48550/arXiv.0811.0898`.

19 Austin G. Fowler. Constructing arbitrary steane code single logical qubit fault-tolerant gates. *Quantum Inf. Comput.*, 11(9&10):867–873, September 2011. `doi:10.26421/QIC11.9-10-10`.

20 Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012. `doi:10.1103/PhysRevA.86.032324`.

21 Vlad Gheorghiu, Michele Mosca, and Priyanka Mukhopadhyay. T-count and T-depth of any multi-qubit unitary. *npj Quantum Information*, 8(1), November 2022. `doi:10.1038/s41534-022-00651-y`.

22 Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford + T circuits. *Physical Review A*, 87(3):032332, 2013. `doi:10.1103/PhysRevA.87.032332`.

23 David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the T-count, 2013. `doi:10.48550/arXiv.1308.4134`.

24 Henning Günther and Georg Weissenbacher. Incremental bounded software model checking. In Neha Rungta and Oksana Tkachuk, editors, *2014 International Symposium on Model Checking of Software, SPIN 2014, Proceedings, San Jose, CA, USA, July 21-23, 2014*, pages 40–47. ACM, 2014. `doi:10.1145/2632362.2632374`.

25 Yifei Huang and Peter Love. Approximate stabilizer rank and improved weak simulation of Clifford-dominated circuits for qudits. *Phys. Rev. A*, 99:052307, May 2019. `doi:10.1103/PhysRevA.99.052307`.

26 Zhengfeng Ji and Xiaodi Wu. Non-identity check remains QMA-complete for short circuits. *arXiv:0906.5416*, 2009. `doi:10.48550/arXiv.0906.5416`.

27 N. Cody Jones. Logic synthesis for fault-tolerant quantum computers, 2013. `doi:10.48550/arXiv.1310.7290`.

28 Aleks Kissinger and John van de Wetering. Simulating quantum circuits with zx-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 7(4):044001, 2022. doi:10.48550/arXiv.2109.01076.

29 Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates, 2013. doi:10.48550/arXiv.1206.5236.

30 Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum clifford and T circuits. *IEEE Transactions on Computers*, 65(1):161–172, 2016. doi:10.1109/TC.2015.2409842.

31 Lucas Kocia and Mohan Sarovar. Improved simulation of quantum circuits by fewer Gaussian eliminations. *Physical Review A*, 103(2), 2021. doi:10.1103/physreva.103.022603.

32 Frédéric Koriche, Jean-Marie Lagniez, Pierre Marquis, and Samuel Thomas. Knowledge compilation for model counting: Affine decision trees. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 947–953. IJCAI/AAAI, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6574.

33 Viktor Kuncak, Mikaël Mayer, Ruzica Piskac, and Philippe Suter. Complete functional synthesis. *ACM Sigplan Notices*, 45(6):316–329, 2010. doi:10.1145/1806596.1806632.

34 Nian-Ze Lee, Yen-Shi Wang, and Jie-Hong R Jiang. Solving stochastic boolean satisfiability under random-exist quantification. In *IJCAI*, pages 688–694, 2017. doi:10.24963/ijcai.2017/96.

35 Michael L Littman, Stephen M Majercik, and Toniann Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27:251–296, 2001. doi:10.1023/A:1017584715408.

36 Dmitri Maslov and Martin Roetteler. Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory*, 64(7):4729–4738, July 2018. doi:10.1109/tit.2018.2825602.

37 Olivia Di Matteo and Michele Mosca. Parallelizing quantum circuit synthesis. *Quantum Science and Technology*, 1(1):015003, October 2016. doi:10.1088/2058-9565/1/1/015003.

38 Jingyi Mei, Marcello M. Bonsangue, and Alfons Laarman. Simulating quantum circuits by model counting. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part III*, volume 14683 of *Lecture Notes in Computer Science*, pages 555–578. Springer, 2024. doi:10.1007/978-3-031-65633-0_25.

39 Jingyi Mei, Tim Coopmans, Marcello M. Bonsangue, and Alfons Laarman. Equivalence checking of quantum circuits by model counting. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*, volume 14740 of *Lecture Notes in Computer Science*, pages 401–421. Springer, 2024. doi:10.1007/978-3-031-63501-4_21.

40 Jingyi Mei, Jan Martens, and Alfons Laarman. Disentangling the gap between quantum and #sat. In Chutiporn Anutariya and Marcello M. Bonsangue, editors, *Theoretical Aspects of Computing - ICTAC 2024 - 21st International Colloquium, Bangkok, Thailand, November 25-29, 2024, Proceedings*, volume 15373 of *Lecture Notes in Computer Science*, pages 17–40, Berlin, Heidelberg, 2024. Springer. doi:10.1007/978-3-031-77019-7_2.

41 Richard Meister, Cica Gustiani, and Simon C Benjamin. Exploring ab initio machine synthesis of quantum circuits. *New Journal of Physics*, 25(7):073018, July 2023. doi:10.1088/1367-2630/ace077.

42 Michael A Nielsen and Isaac L Chuang. Quantum information and quantum computation. *Cambridge: Cambridge University Press*, 2(8):23, 2000.

43 Philipp Niemann, Robert Wille, and Rolf Drechsler. Advanced exact synthesis of clifford+t circuits. *Quantum Information Processing*, 19(9):317, August 2020. doi:10.1007/s11128-020-02816-0.

44 Anouk Paradis, Jasper Dekoninck, Benjamin Bichsel, and Martin T. Vechev. Synthetiq: Fast and versatile quantum circuit synthesis. *Proc. ACM Program. Lang.*, 8(OOPSLA1):55–82, April 2024. doi:10.1145/3649813.

45 Maxim Raginsky. A fidelity measure for quantum channels. *Physics Letters A*, 290(1-2):11–18, 2001. doi:10.1016/S0375-9601(01)00640-5.

46 Narayanan Rengaswamy, A. Robert Calderbank, Henry D. Pfister, and Swanand Kadhe. Synthesis of logical clifford operators via symplectic geometry. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 791–795. IEEE, 2018. doi:10.1109/ISIT.2018.8437652.

47 Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations. *CoRR*, abs/1403.2975, 2016. doi:10.48550/arXiv.1403.2975.

48 Sarah Schneider, Lukas Burgholzer, and Robert Wille. A SAT encoding for optimal clifford circuit synthesis. In Atsushi Takahashi, editor, *Proceedings of the 28th Asia and South Pacific Design Automation Conference, ASPDAC 2023, Tokyo, Japan, January 16-19, 2023*, ASPDAC '23, pages 190–195, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3566097.3567929.

49 Peter Selinger. Efficient Clifford+T approximation of single-qubit operators, 2014. doi:10.48550/arXiv.1212.6253.

50 Meghana Sistla, Swarat Chaudhuri, and Thomas W. Reps. Symbolic quantum simulation with quasimodo. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part III*, volume 13966 of *Lecture Notes in Computer Science*, pages 213–225. Springer, Springer, 2023. doi:10.1007/978-3-031-37709-9_11.

51 Dimitrios Thanos, Tim Coopmans, and Alfons Laarman. Fast equivalence checking of quantum circuits of clifford gates. In Étienne André and Jun Sun, editors, *Automated Technology for Verification and Analysis - 21st International Symposium, ATVA 2023, Singapore, October 24-27, 2023, Proceedings, Part II*, volume 14216 of *Lecture Notes in Computer Science*, pages 199–216, Cham, 2023. Springer. doi:10.1007/978-3-031-45332-8_10.

52 Dimitrios Thanos, Alejandro Villoria, Sebastiaan Brand, Arend-Jan Quist, Jingyi Mei, Tim Coopmans, and Alfons Laarman. Automated reasoning in quantum circuit compilation. In *SPIN 2024*, 2024. URL: https://spin-web.github.io/SPIN2024/assets/preproceedings/SPIN2024-paper6.pdf.

53 Lieuwe Vinkhuijzen, Tim Coopmans, David Elkouss, Vedran Dunjko, and Alfons Laarman. LIMDD: A decision diagram for simulation of quantum computing including stabilizer states. *Quantum*, 7:1108, 2023. doi:10.22331/q-2023-09-11-1108.

54 Suwei Yang, Victor C. Liang, and Kuldeep S. Meel. INC: A scalable incremental weighted sampler. In Alberto Griggio and Neha Rungta, editors, *22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022*, volume 3, pages 205–213. TU Wien Academic Press, IEEE, 2022. doi:10.34727/2022/isbn.978-3-85448-053-2_27.

55 Suwei Yang and Kuldeep S. Meel. Towards projected and incremental pseudo-boolean model counting. *CoRR*, abs/2412.14485, 2024. doi:10.48550/arXiv.2412.14485.

A Appendix

A.1 Proof of Theorem 5

▶ **Theorem 5.** *Let $\mathcal{C}$ be an n-qubit circuit, which is encoded by $F_{\mathcal{C}}$ with the corresponding weight function W. Then, the following four statements are equivalent to each other:*

- $\mathcal{C} \equiv I^{\otimes n}$ *($\mathcal{C}$ is equivalent to the identity circuit $I^{\otimes n} = I_{2^n}$).*
- *Encoding the circuit $\mathcal{C}$ in* **PB**, *the* ***linear encoding*** *has weighted model count [39]:*

$$\#SAT_W(F_{\mathcal{P}}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q}, \vec{q}') \wedge F_{\mathcal{P}}(\vec{q}')) = 1 \text{ for all } \mathcal{P} \in \{X_j, Z_j \mid j \in [n]\}. \qquad (4)$$

- *Encoding the circuit in either* **CB** *or* **PB***, the* ***cyclic encoding*** *has weighted model count (this approach can be viewed as checking "overlap" with the identity $I^{\otimes n}$):*

$$\#SAT_W(F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{I^{\otimes n}}(\vec{q},\vec{q}')) = c \text{ with } |c| = 2^n \text{ for } \mathbf{CB} \text{ and } c = 4^n \text{ for } \mathbf{PB}. \tag{5}$$

- *Encoding the circuit $\mathcal{C}$ in* **PB***, the* ***linear-cyclic encoding*** *has weighted model count:*

$$\#SAT_W\Big(\bigvee_{\mathcal{P}\in\{X_j,Z_j\,|\,j\in[n]\}} F_{\mathcal{P}}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{I^{\otimes n}}(\vec{q},\vec{q}')\Big) = 2n. \tag{6}$$

where $\vec{q},\vec{q}'$ are Boolean variables encoding the initial and final quantum state, respectively. Note that $F_{I^{\otimes n}}(\vec{q},\vec{q}') = \bigwedge_{i\in[n]}(q_i \Leftrightarrow q_i')$, where $q_i \Leftrightarrow q_i'$ is shorthand for $(x_i \Leftrightarrow x_i') \wedge (z_i \Leftrightarrow z_i')$ in the Pauli basis.

Proof. First of all, from [39, Cor. 1], the circuit $\mathcal{C}$ is equivalent to the identity $I^{\otimes n}$ if and only if Equation 4 holds.

Next, we show that all three equations Equation 6 and Equation 5 in both bases are equivalent.

For Equation 6, as stated in Corollary 1 and the proof of Lemma 2 in [39], we have

$$\#SAT_W(F_{\mathcal{P}}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{\mathcal{P}}(\vec{q}')) \leq 1.$$

for all $\mathcal{P} \in \{X_j, Z_j \mid j \in [n]\}$. Thus we infer that

$$\sum_{\mathcal{P}\in\{X_j,Z_j|j\in[n]\}} \#SAT_W(F_{\mathcal{P}}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{\mathcal{P}}(\vec{q}')) \leq 2n,$$

where the value achieves $2n$ if and only if each of the summands achieves 1. Therefore Equation 6 is true if and only if Equation 4 is true, as demonstrated in [39, Prop. 1].

For Equation 5 in **PB**, the idea is similar. The value of the weighted model count of Equation 5 is equivalent to

$$\sum_{\mathcal{P}\in\{X,Y,Z,I\}^{\otimes n}} \#SAT_W(F_{\mathcal{P}}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}) \wedge F_{\mathcal{P}}(\vec{q})) \leq 4^n,$$

which can achieve 4^n if and only if for all 4^n Pauli strings $\mathcal{P}$, the weighted model counting of $\mathcal{P}$ achieves 1. Since $\{X_j, Z_j \mid j \in [n]\} \subseteq \{X,Y,Z,I\}^{\otimes n}$, we have *Equation* 5 $\Rightarrow$ *Equation* 4. From [51], if two unitaries are equivalent over $\{X_j, Z_j \mid j \in [n]\}$, they are equivalent over $\{X,Y,Z,I\}^{\otimes n}$. We have Equation 4$\Rightarrow$ Equation 5. Therefore Equation 4$\Leftrightarrow$ Equation 5.

For Equation 5 in **CB**, since

$$\begin{aligned}
U_{\mathcal{C}} = \lambda \cdot I_{2^n} &\Leftrightarrow \langle b| U_{\mathcal{C}} |b\rangle = \lambda \text{ for } b \in \{0,1\}^n \\
&\Leftrightarrow \sum_{b\in\{0,1\}^n} \langle b| U_{\mathcal{C}} |b\rangle = \lambda \cdot 2^n, \\
&\Leftrightarrow \sum_{b\in\{0,1\}^n} \Big(\#SAT_W\big(F_{|b\rangle}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{|b\rangle}(\vec{q}')\big)\Big) = \lambda \cdot 2^n, \qquad (\textit{Lemma } 2) \\
&\Leftrightarrow \sum_{b\in\{0,1\}^n} \Big(\#SAT_W\big(F_{|b\rangle}(\vec{q}) \wedge F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{I_{2^n}}(\vec{q},\vec{q}')\big)\Big) = \lambda \cdot 2^n, \\
&\Leftrightarrow \#SAT_W(F_{\mathcal{C}}(\vec{q},\vec{q}') \wedge F_{I_{2^n}}(\vec{q},\vec{q}')) = \lambda \cdot 2^n,
\end{aligned}$$

where $|\lambda|^2 = 1$, we have Equation 5 $\Leftrightarrow \mathcal{C} \equiv I^{\otimes n}$. ◀

A.2 Optimization rules

To optimize the synthesis encoding, we implement additional constraints.

The first set of rules ensures that we avoid redundant combinations of gates, such as HH, since they can be replaced with I gates:

- **Rule 1**: No two H gates in a row on the same qubit, since $HH = I$:

$$F^d_{R1}(P) = \bigwedge_{k\in[d-1]} \bigwedge_{i\in[n]} (\overline{p}^k_{H,i} \vee \overline{p}^{k+1}_{H,i})$$

- **Rule 2**: No 8 T gates in a row on the same qubit since $T^8 = I$:

$$F^d_{R2}(P) = \bigwedge_{k\in[d-7]} \bigwedge_{i\in[n]} \bigvee_{j\in[8]} \overline{p}^{k+j}_{T,i}$$

- **Rule 3**: No two CX gates in a row on the same qubits since $CX_{i,j} \cdot CX_{i,j} = I$:

$$F^d_{R3}(P) = \bigwedge_{k\in[d-1]} \bigwedge_{i,j\in[n],j\neq i} (\overline{p}^k_{CX,i,j} \vee \overline{p}^{k+1}_{CX,i,j})$$

The second set of rules aims to have a canonical representation for a given set of gates. Our guideline is that every non-I is pushed back as far as possible. This means not allowing any single qubit gate to follow an I gate, other than I itself. For two-qubit gates, we do not allow following I on both qubits.

- **Rule 4**: No single qubit gates other than I after an I gate:

$$F^d_{R4}(P) = \bigwedge_{k\in[d-1]} \bigwedge_{i\in[n]} (p^k_{I,i} \Rightarrow (p^{k+1}_{I,i} \vee \bigvee_{j\in[n],j\neq i} (p^{k+1}_{CX,i,j} \vee p^{k+1}_{CX,j,i})))$$

- **Rule 5**: No CX gate following I gate on both qubits:

$$F^d_{R5}(P) = \bigwedge_{k\in[d-1]} \bigwedge_{i,j\in[n],j\neq i} (p^{k+1}_{CX,i,j} \Rightarrow (\overline{p}^k_{I,i} \vee \overline{p}^k_{I,j}))$$