ARTICLE OPEN

Check for updates

A (quasi-)polynomial time heuristic algorithm for synthesizing T-depth optimal circuits

Vlad Gheorghiu[1,2], Michele Mosca[1,2,3,4] and Priyanka Mukhopadhyay[1,3]✉

We investigate the problem of synthesizing T-depth optimal quantum circuits for exactly implementable unitaries over the Clifford +T gate set. We construct a subset, $\mathbb{V}_n$, of T-depth 1 unitaries. T-depth-optimal decomposition of unitary U is $e^{i\phi}(\prod_i V_i)C$, $V_i \in \mathbb{V}_n$, C is Clifford and $|\mathbb{V}_n| \leq n \cdot 2^{5.6n}$. We use nested meet-in-the-middle technique to synthesize provably depth-optimal and T-depth-optimal circuits. For the latter, we achieve space and time complexity $O((4^{n^2})^{\lceil d/c \rceil})$ and $O((4^{n^2})^{(c-1)\lceil d/c \rceil})$ respectively (d is the minimum T-depth, $c \geq 2$ a constant). The previous best algorithm had complexity $O((3^n \cdot 2^{kn^2})^{\lceil \frac{d}{2} \rceil} \cdot 2^{kn^2})$ ($k > 2.5$ a constant). We design a more efficient algorithm with space and time complexity poly$(n, 2^{5.6n}, d)$ (or poly$(n^{\log n}, 2^{5.6n}, d)$ with weaker assumptions). The claimed efficiency, optimality depends on conjectures.

npj Quantum Information (2022)8:110 ; https://doi.org/10.1038/s41534-022-00624-1

INTRODUCTION

The notion of a quantum computer was introduced by Feynman[1] as a solution to the limitations of conventional or classical computers. In numerous fields algorithms designed for quantum computers outperform their classical counterparts. Some examples include integer factorization[2,3], searching an unstructured solution space[4]. One of the most widely used methods for describing and implementing quantum algorithms is quantum circuits, which consist of a series of elementary operations dictated by the implementing technologies.

Circuit synthesis and optimization is a significant part of any computer compilation process whose primary goal is to translate from a human-readable input (programming language) into instructions that can be executed directly on hardware. In quantum circuit synthesis, the aim is to decompose an arbitrary unitary operation into a sequence of gates from a universal set, which usually consists of Clifford group gates and at least one more non-Clifford gate[5]. The non-Clifford gates are more expensive to implement fault-tolerantly than Clifford gates. A popular universal fault-tolerant gate set is the Clifford+T, in which the cost of fault-tolerant implementation of the T gate[6–8] exceeds the cost of the Clifford group gates by as much as a factor of a hundred or more in most error correction schemes. Fault-tolerant designs and quantum error correction are essential in order to deal with errors due to noise in quantum information, faulty quantum gates, faulty quantum state preparation, and faulty measurements. In particular, for long computations, where the number of operations in the computation vastly exceeds the number of operations one could hope to execute before errors make negligible the likelihood of obtaining a useful answer, fault-tolerant quantum error correction is the only known way to reliably implement the computation. With recent advances in quantum information processing technologies[9–12] and fault-tolerant thresholds[7,13,14], as scalable quantum computation is becoming more and more viable we need efficient automated design tools targeting fault-tolerant quantum computers. And minimization of the number of T gates in quantum circuits remains an important and widely studied goal. It has been argued[15–19] that it is also important to reduce the maximum number of T gates in any circuit path. While the former metric is referred to as the T-count, the latter is called the T-depth of the circuit.

An n-qubit quantum circuit consisting of Clifford+T gates implements a $2^n \times 2^n$ unitary. In the context of reducing resources (such as T gates) necessary to implement a unitary U, two types of problems have been investigated—(a) synthesis and (b) re-synthesis. The input to an algorithm for a quantum circuit synthesis problem is a $2^n \times 2^n$ unitary matrix and the goal is to output a circuit implementing it[20,21]. When we impose additional constraints like minimizing certain resources such as T-count or T-depth[16], we often call this as (resource)-optimal synthesis problem. From here on, we focus on the T-depth as the resource being minimized. To be more precise, there can be more than one (equivalent) circuits implementing U. A T-depth-optimal synthesis algorithm is required to output a circuit with the minimum T-depth. We call this a T-depth-optimal circuit. With a slight abuse of terminology, we use the terms 'synthesis algorithm' and 'T-depth optimal synthesis algorithm' interchangeably, which should be clear from the context. It must be observed that with the addition of this tighter constraint on the output (i.e. that it be T-depth optimal), there is a probability that the complexity of the problems change. For example, it was known that a quantum circuit can be synthesized in poly(2^n) time, where 2^n is the input size[20,22]. The work in ref. [23] was the first to propose a poly(2^n) time algorithm for synthesizing T-count-optimal circuits.

With an input size $O(2^n)$, we cannot hope to get an optimal synthesis algorithm with a complexity of less than that. This makes these algorithms practically intractable after a certain value of n. Hence re-synthesis algorithms have been developed, where some

[1]Institute for Quantum Computing, University of Waterloo, Waterloo, ON, Canada. [2]softwareQ Inc., Kitchener, ON, Canada. [3]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. [4]Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada. ✉email: mukhopadhyay.priyanka@gmail.com

more information is provided as input, usually a circuit implementing U[17,24] and the task is to reduce (not minimize) the T-depth in the input circuit. In the literature, nearly every re-synthesis algorithm (usually with complexity poly(n)) does not account for the complexity of generating the initial input circuit from U. This step itself has complexity $O(2^n)$. A full study comparing these two kinds of algorithms and the quality of their results is beyond the scope of this work.

Despite their higher complexity compared to re-synthesis algorithms, the importance of studying optimal synthesis algorithms cannot be undermined. They can be used to assess the quality of a re-synthesis algorithm, for example, how close are their output to an optimal one. They can be used to generate the input circuit of a re-synthesis algorithm. A large circuit can be fragmented and the unitary of each part can be synthesized optimally, giving an overall reduction in resources. From a theoretical viewpoint, they shed light on the complexity of problems that are usually harder than their relaxed re-synthesis counterpart. As an illustration of the significance of developing resource-optimal synthesis algorithms, we observe the following. In our paper, we have been able to generate T-depth-optimal circuits for standard unitaries like Toffoli, Fredkin, Peres, and Quantum OR, which were not generated by the re-synthesis methods used in ref. [16]. Though this has a T-depth-optimal synthesis algorithm, it could not synthesize beyond 2-qubit unitaries with T-depth 2. For larger unitaries like the mentioned 3-qubit ones, it used peep-hole optimization, a popular re-synthesis method. Except for Toffoli, they obtained T-depth 4, even for unitaries that are Clifford equivalent to Toffoli. The approach in this paper has significantly lower complexity than the synthesis method in ref. [16] and is able to synthesize T-depth 3 circuits.

The Solovay–Kitaev algorithm[20,25] guarantees that given a unitary U, we can generate a circuit with a universal gate set like Clifford + T, such that the unitary U' implemented by the circuit is at most a certain distance from U (the distance being induced by some appropriate norm). In fact, it has been proved that we can get a Clifford + T circuit that exactly implements U, i.e. $U' = U$ (up to some global phase) if and only if the entries of U are in ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$[21]. We denote this group of unitaries by $\mathcal{J}_n$. For example, the Toffoli and Fredkin gates belong to $\mathcal{J}_3$. Thus quantum synthesis algorithms can be further subdivided into two categories: (a) exact synthesis algorithms, that output a circuit implementing $U' = U$ (e.g. refs. [23,26]) and (b) approximate synthesis algorithms, that output a circuit implementing U' such that U' is close to U (e.g. ref. [27]).

In this paper we focus on the group $\mathcal{J}_n$ of unitaries that can be exactly synthesized and consider the following synthesis problem.

MIN T-DEPTH

Given $U \in \mathcal{J}_n$ synthesize a T-depth optimal circuit for it. In the decision version of this problem we are given $U \in \mathcal{J}_n$ and $m \in \mathbb{N}$, and the goal is to decide if the minimum T-depth of U is at most m.

We consider the complexity of our exact synthesis algorithms as a function of m and $N = 2^n$. We treat arithmetic operations on the entries of U at unit cost, and we do not account for the bit complexity associated with specifying or manipulating them.

We first show (in the section "Methods") that the nested meet-in-the-middle (MITM) technique developed in ref. [23] can be applied to the problem of synthesizing provably depth-optimal circuits. This gives us a depth-optimal-synthesis algorithm with time complexity $O(|\mathcal{V}_{n,\mathcal{G}}|^{(c-1)\lceil \frac{d'}{c} \rceil})$ and space complexity $O(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil \frac{d'}{c} \rceil})$, where $\mathcal{V}_{n,\mathcal{G}}$ is the set of depth-1 n-qubit unitaries over the gate set $\mathcal{G}$, d' is the min-depth of input unitary, and $c \geq 2$ is the extent of nesting. This gives us a space–time trade-off for MITM-related techniques applied to this problem.

Table 1. Comparison of generation time of $\mathbb{V}_n$ and $\mathcal{C}_n$.

#Qubits (n)	$\lvert\mathbb{V}_n\rvert$	Generation time (s)	$\lvert\mathcal{C}_n\rvert$	Generation time[18]
2	122	0.015	≈11,520	1 s
3	2282	2.212	≈92, 897, 280	>4 days
4	35,846	10 min 24 s	N/A	N/A

Next, we apply this technique to synthesize T-depth optimal circuits. We work with channel representation of unitaries. We define a special subset, $\mathbb{V}_n$, of T-depth-1 unitaries, which can generate a T-depth-optimal decomposition of any exactly implementable unitary (up to some Clifford). We prove $|\mathbb{V}_n| \in O(n \cdot 2^{5.6n})$. Then we give an algorithm that returns provably T-depth-optimal circuits and has time and space complexity $O((4^{n^2})^{(c-1)\lceil \frac{d}{c} \rceil})$ and $O((4^{n^2})^{\lceil \frac{d}{c} \rceil})$, respectively, where d is the min-T-depth of input unitary. This is much less than the complexity of the algorithm in ref. [16]. It had a complexity $O((3^n|\mathcal{C}_n|)^{\lceil \frac{d}{2} \rceil} \cdot |\mathcal{C}_n|)$, where $\mathcal{C}_n$ is the set of n-qubit Clifford operators. $|\mathcal{C}_n| \in O(2^{kn^2})$[28–30], for some constant $k > 2.5$. In ref. [16] the authors iteratively used $\mathcal{C}_n$, as indicated by the stated complexity. It took more than 4 days to generate $\mathcal{C}_3$[16]. In fact, in ref. [16] the largest circuit optimally synthesized had 2 qubits and had T-depth 2. We use much smaller sets, which has cardinality $O(4^{n^2})$ and can be derived from $\mathbb{V}_n$. We can generate $\mathbb{V}_3$ in a few seconds (Table 1). This gives a (rough) indication of the computational advantage one can have if algorithms are designed with such smaller sets, and thus the motivation to come up with alternate representations.

To improve the efficiency further, we develop another algorithm, MIN-T-DEPTH, whose complexity depends on some conjectures that have been motivated by the polynomial complexity algorithm in ref. [23] for synthesizing T-count optimal circuits. At this point, our conjectures do not seem to be derived from the ones in ref. [23]. If our assumptions are true, then this algorithm returns T-depth-optimal circuits with space and time complexity poly($n, 2^{5.6n}, d$). Under a weaker assumption, this complexity is poly($n^{\log n}, d, 2^{5.6n}$).

Apart from T-depth-optimal circuit synthesis algorithms for exactly implementable unitaries, the generating set $\mathbb{V}_n$, has found other applications like optimal synthesis algorithms for approximately implementable unitaries[31].

The technique of meet-in-the-middle (MITM) and its variant (nested MITM) was used for the exact synthesis of provably T-count optimal circuits in refs. [23,26] as well as provably depth optimal circuits in ref. [16]. This MITM technique has also been used with deterministic walks in ref. [32] to construct a parallel framework for the synthesis of T-count optimal circuits. The time as well as space complexity of the algorithms in refs. [26,32] is $O((2^n)^m)$ where m is the T-count of the $2^n \times 2^n$ input unitary. (The T-count of a unitary is the minimum number of T gates required to implement it.) The time and space complexity of the algorithm in ref. [16] is $O((3^n \cdot 2^{kn^2})^{\lceil \frac{d}{2} \rceil} \cdot 2^{kn^2})$, where k is a constant and d is the min-T-depth. The first T-count-optimal synthesis algorithm which reduces the complexity to poly($2^n, m$), assuming some conjectures, was given in ref. [23].

RESULTS AND DISCUSSION

Preliminaries

We write $[K] = \{1, 2, \ldots, K\}$. We assume that a set has distinct elements. We denote the $n \times n$ identity matrix by $\mathbb{I}_n$ or $\mathbb{I}$ if the dimension is clear from the context. The size of an n-qubit unitary is denoted by $N = 2^n$. We call the number of non-zero entries in a matrix as its Hamming weight.

The *single qubit Pauli matrices* are as follows:

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

Parenthesized subscripts are used to indicate qubits on which an operator acts. For example, $\mathrm{X}_{(1)} = \mathrm{X} \otimes \mathbb{I}^{\otimes(n-1)}$ implies that Pauli X matrix acts on the first qubit and the remaining qubits are unchanged.

The n-qubit Pauli operators are: $\mathcal{P}_n = \{Q_1 \otimes Q_2 \otimes \ldots \otimes Q_n : Q_i \in \{\mathbb{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}\}$.

The *single-qubit Clifford group* $\mathcal{C}_1$ is generated by the Hadamard and phase gates: $\mathcal{C}_1 = \langle \mathrm{H}, \mathrm{S} \rangle$where

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

When $n > 1$ the *n-qubit Clifford group* $\mathcal{C}_n$ is generated by these two gates (acting on any of the n qubits) along with the two-qubit CNOT $= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \mathrm{X}$ gate (acting on any pair of qubits). Cliffords map Paulis to Paulis, up to a possible phase of -1, i.e. for any $P \in \mathcal{P}_n$ and any $C \in \mathcal{C}_n$ we have $CPC^\dagger = (-1)^b P'$ for some $b \in \{0, 1\}$ and $P' \in \mathcal{P}_n$. In fact, given two Paulis (neither equal to the identity), it is always possible to efficiently find a Clifford which maps one to the other.

Fact 2.1 (Gosset et al.[26]) For any $P, P' \in P_n \backslash \{\mathrm{I}\}$ there exists a Clifford $C \in C_n$ such that $CPC^\dagger = P'$. A circuit for C over the gate set {H, S, CNOT} can be computed efficiently (as a function of n).

The group $\mathcal{J}_n$ is generated by the n-qubit Clifford group along with the T gate. Thus

$$\mathcal{J}_1 = \langle \mathrm{H}, \mathrm{T} \rangle \quad \text{and} \quad \mathcal{J}_n = \left\langle \mathrm{H}_{(i)}, \mathrm{T}_{(i)}, \mathrm{CNOT}_{(i,j)} : i, j \in [n] \right\rangle$$

It can be easily verified that $\mathcal{J}_n$ is a group, since the H and CNOT gates are their own inverses and $T^{-1} = T^7$. We denote the group of unitaries exactly synthesized over the Clifford + T gate set by $\mathcal{J}_n$. Some elements of this group cannot be exactly synthesized over this gate set without ancilla qubits[21].

Channel representations

An n-qubit unitary U can be completely determined by considering its action on a Pauli $P_s \in \mathcal{P}_n : UP_sU^\dagger$. Since $\mathcal{P}_n$ is a basis for the space of all Hermitian $N \times N$ matrices we can write

$$UP_sU^\dagger = \sum_{P_r \in \mathcal{P}_n} \widehat{U}_{rs} P_r, \qquad \text{where} \quad \widehat{U}_{rs} = \frac{1}{2^n} \mathrm{Tr}(P_r U P_s U^\dagger). \tag{1}$$

This defines a $N^2 \times N^2$ matrix $\widehat{U}$ with rows and columns indexed by Paulis $P_r, P_s \in \mathcal{P}_n$. We refer to $\widehat{U}$ as the channel representation of U[26].

By Hermitian conjugation each entry of the matrix $\widehat{U}$ is real. The channel representation respects matrix multiplication, i.e. $\widehat{UV} = \widehat{U}\widehat{V}$. Setting $V = U^\dagger$ and using the fact that $\widehat{U^\dagger} = \left(\widehat{U}\right)^\dagger$, we see that the channel representation $\widehat{U}$ is unitary. If $U \in \mathcal{J}_n$, implying its entries are in the ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$[21], then from Eq. (1) the entries of $\widehat{U}$ are in the same ring. Since $\widehat{U}$ is real, its entries are from the subring

$$\mathbb{Z}\left[\frac{1}{\sqrt{2}}\right] = \left\{ \frac{a + b\sqrt{2}}{\sqrt{2}^k} : a, b \in \mathbb{Z}, \quad k \in \mathbb{N} \right\}$$

The channel representation identifies unitaries that differ by a global phase. We write the following for the groups in which global phases are modded out.

$$\widehat{\mathcal{J}_n} = \left\{ \widehat{U} : U \in \mathcal{J}_n \right\}, \quad \widehat{\mathcal{C}_n} = \left\{ \widehat{C} : C \in \mathcal{C}_n \right\}$$

Each $Q \in \widehat{\mathcal{C}_n}$ is a unitary matrix with one nonzero entry in each row and each column, equal to ± 1. This is because Cliffords map Paulis to Paulis up to a possible phase of -1. The converse also holds: if $W \in \widehat{\mathcal{J}_n}$ has this property then $W \in \widehat{\mathcal{C}_n}$. Since the definition of T-count is insensitive to the global phase, it is well-defined in the channel representation and so $\mathcal{T}(\widehat{U})$ is defined to be equal to $\mathcal{T}(U)$. If a unitary U requires ancilla to be implemented, then we can consider the unitary that acts on the joint state space of input and ancilla qubits. From here on, with a slight abuse of notation when we write $U \in \mathcal{J}_n$ we assume it is the unitary that acts on this joint state space.

Definition 2.1. For any non-zero $v \in \mathbb{Z}\left[\frac{1}{\sqrt{2}}\right]$ the smallest denominator exponent, denoted by sde(v), is the smallest $k \in \mathbb{N}$ for which

$$v = \frac{a + b\sqrt{2}}{\sqrt{2}^k} \qquad \text{with } a, b \in \mathbb{Z}.$$

We define sde(0) = 0. For a $d_1 \times d_2$ matrix M with entries over this ring we define

$$\mathrm{sde}(M) = \max_{a \in [d_1], b \in [d_2]} \mathrm{sde}(M_{ab})$$

T-depth

The purpose of this section is to derive a generating set consisting of T-depth 1 unitaries, such that we can write a T-depth-optimal decomposition of any exactly implementable unitary (up to global phase) as a product of elements of this set and a trailing Clifford. This set must be efficiently generated and have a finite cardinality. We first give some essential definitions.

Definition 2.2. The depth of a circuit is the length of any critical path through the circuit. Representing a circuit as a directed acyclic graph with nodes corresponding to the circuit's gates and edges corresponding to gate inputs/outputs, a critical path is a path of maximum length flowing from an input of the circuit to an output.

In other words, suppose the unitary U implemented by a circuit is written as a product $U = U_m U_{m-1} \ldots U_1$ such that each U_i can be implemented by a circuit in which all the gates can act in parallel or simultaneously. We say U_i has depth 1 and m is the depth of the circuit. We often refer to each U_i as a stage or (parallel) block. The T-depth of a circuit is the number of stages (or unitaries U_i) where the T/T† gate is the only non-Clifford gate and all the T/T† gates can act in parallel. The min-T-depth or T-depth of a unitary U is the minimum T-depth of a Clifford + T circuit that implements it (up to a global phase). We often simply say T-depth instead of 'T-depth of a unitary'. It should be clear from the context.

Any unitary U, having a circuit with T-depth t can be written as follows:

$$U = C_t \left(\overline{T}_{(1)} \ldots \overline{T}_{(n)}\right) C_{t-1} \left(\overline{T}_{(1)} \ldots \overline{T}_{(n)}\right) \ldots C_1 \left(\overline{T}_{(1)} \ldots \overline{T}_{(n)}\right) C_0 \tag{2}$$

In the above equation $\overline{T} \in \{\mathrm{T}, \mathrm{T}^\dagger, \mathbb{I}\}$ is used to indicate whether there is T, T† or $\mathbb{I}$ gate in that qubit. $C_1, C_2, C_3 \ldots C_t \in \mathcal{C}_n$. For simplicity we ignore the global phase. We can also write the above

equation as follows:

$$U = \left(C_t\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)C_t^\dagger\right)\left(C_tC_{t-1}\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)(C_tC_{t-1})^\dagger\right)\ldots$$
$$\ldots\left(C_tC_{t-1}\ldots C_1\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)(C_tC_{t-1}\ldots C_1)^\dagger\right)C_tC_{t-1}\ldots C_1C_0$$
$$= \left(C_t\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)C_t^\dagger\right)\left(C'_{t-1}\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)(C'_{t-1})^\dagger\right)\ldots\left(C'_1\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)(C'_1)^\dagger\right)C'_0$$
$$[\text{where } C'_1,\ldots C'_t\in\mathcal{C}_n]$$
$$= V_tV_{t-1}\ldots V_1C'_0 \qquad \text{where } V_j = \left(C'_j\left(\prod_{i\in[n]}\overline{T}_{(i)}\right)(C'_j)^\dagger\right) \tag{3}$$

We call each V_j as a (parallel) block. It is a product of T or T† gates on distinct qubits, conjugated by a Clifford. Thus the following set

$$\mathbb{V}'_n = \{\prod_{i\in[n]} C\overline{T}_{(i)}C^\dagger, C\in\mathcal{C}_n, \overline{T}\in\{T,T^\dagger,\mathbb{I}\}\} \tag{4}$$

can be regarded as a generating set (up to a Clifford) for the decomposition of an exactly implementable unitary. More precisely, any exactly implementable unitary U (ignoring the global phase) can be written as a product of elements from this set and a Clifford. The number of elements from $\mathbb{V}_n$ is equal to the T-depth of this decomposition or circuit. Any decomposition of U with the minimum number of parallel blocks is called a T-depth-optimal decomposition. A circuit implementing U with the minimum T-depth is called a T-depth-optimal circuit.

We can equivalently write each V_j as follows:

$$V_j = \prod_{i\in[n]}\left(C'_j\overline{T}_{(i)}C'^\dagger_j\right) \tag{5}$$

Now if $C\in\mathcal{C}_n$ then

$$CT_{(i)}C^\dagger = \tfrac{1}{2}(1+e^{\frac{i\pi}{4}})\mathbb{I}+\tfrac{1}{2}(1-e^{\frac{i\pi}{4}})CZ_{(i)}C^\dagger = \tfrac{1}{2}(1+e^{\frac{i\pi}{4}})\mathbb{I}+\tfrac{1}{2}(1-e^{\frac{i\pi}{4}})P \quad [P\in\pm\mathcal{P}_n]$$
$$= R(P) \qquad [\text{Let}] \tag{6}$$

The $R(P)$ unitaries and somewhat similar unitaries called Pauli gadgets have been studied extensively in previous works like refs. [26,33]. We believe that the conclusions derived in this paper will enhance the study of these gadgets or special unitaries, such that we can have more applications (for example, see ref. [31]).

Also $(R(P))^\dagger = CT^\dagger_{(i)}C^\dagger = R^\dagger(P)$ (let). Thus we can write Eq. (5) as follows:

$$V_j = \prod_{i\in[n]}\left(C'_j\overline{T}_{(i)}C'^\dagger_j\right) = \prod_{i=n}^{1}\widetilde{R}(P_{ij}) \qquad [\widetilde{R}\in\{R,R^\dagger\},\widetilde{R}(\mathbb{I})=\mathbb{I},P_{ij}\in\pm\mathcal{P}_n] \tag{7}$$

The second subscript of P_{ij} gives the index of the block. The ordering of the intermediate T/T† gates does not matter. It merely changes the sequence of $\widetilde{R}(P_{ij})$, but we get the same product V_j. Given a set S of qubits there are $3^{|S|}$ possible ways of placing a T/T†/$\mathbb{I}$ gate in each qubit. We call each such placement as a configuration of $\overline{T}$ gates and denote it by $\overline{T}_S$.

From Eq. (7) we get a simple way of constructing $\mathbb{V}'_n$.

1. For each $C\in\mathcal{C}_n$ do the following.
   (a) For each configuration $\overline{T}_{[n]}$ do the following.
      i. $V\leftarrow\mathbb{I}$.
      ii. For each $i\in[n]$ do the following.
         If $\overline{T}_{(i)}\neq\mathbb{I}$ then determine $P=CZ_{(i)}C^\dagger$. If $\overline{T}=$ T then $V\leftarrow V\cdot R(P)$, else if $\overline{T}=$ T† then $V\leftarrow V\cdot R^\dagger(P)$.
      iii. Include V in $\mathbb{V}'_n$ if it does not already exist.

The time complexity of this procedure is $O(|\mathcal{C}_n|)$ or $O(2^{kn^2})$, where k is a constant. A bound on $|\mathbb{V}'_n|$ can be obtained by counting all possible distinct n-length strings of $\widetilde{R}(P)$, where $\widetilde{R}\in\{R,R^\dagger\}$ and $P\in\pm\mathcal{P}_n$. Without loss of generality we can assume that every string or sequence is of length n, by filling in $R(\mathbb{I})=R^\dagger(\mathbb{I})=\mathbb{I}$. Thus it gives $|\mathbb{V}'_n|<(2\cdot2\cdot4^n)^n=4^{n+n^2}$. From ref. [26] we know that there are at most $4^{n^2}\cdot|\mathcal{C}_n|$ unitaries (up to global phase) with T-count n. So it is highly plausible that $|\mathbb{V}'_n|\in O(4^{n^2})$.

Every n-length string of $\widetilde{R}(P)$ does not have T-depth 1. We are over-counting a lot here. Our aim is to construct a more compact (smaller) set of T-depth 1 unitaries such that it is possible to write any T-depth 1 unitary as product of unitaries from this set and a Clifford. This is sufficient because it will enable us to write any T-depth-d decomposition (and hence T-depth-optimal decomposition) of a unitary as product of elements from this set and a Clifford (up to global phase). In this way, we can use information from a set of less number of unitaries in order to make more intelligent guesses about a T-depth-optimal decomposition (specially algorithm MIN-T-DEPTH). We would want to prune many Cliffords to be considered at step 1.

Here we make the following observation. There are $2^{O(n^2)}$ Clifford operators that can map $Z_{(i)}$ to a particular Pauli $P\in\mathcal{P}_n$. All of them lead to the same unitary $R(P)$. Similarly, there are many Cliffords such that when $\prod_i Z_{(i)}$ (where the Zs are on different qubits) is conjugated it leads to the same sequence of Paulis (ordering does not matter) i.e. it will give the same unitary $\prod_i R(P_i)$. So for our purpose, what is more important are the mappings or rather images of mappings, and not the Clifford operators. If $CPC^\dagger=P$, we call it a trivial conjugation, for any $P\in\mathcal{P}_n, C\in\mathcal{C}_n$. P, in this case, is trivially conjugated by C.

We now construct a smaller generating set, $\mathbb{V}_n$. We consider each $\widetilde{R}(P)$ as the starting unit of a string and then determine the remaining $n-1$ units. A formal constructive definition of $\mathbb{V}_n$ is as follows.

Definition 2.3. We define $\mathbb{V}_n$, a subset of n-qubit unitaries with T-depth 1, that is constructed as follows.

1. Include $\widetilde{R}(Z_{(i)})$ $(i\in[n])$ in $\mathbb{V}_n$.
2. For each $P\in\pm\mathcal{P}_n\backslash\{\mathbb{I}\}$, for each $q\in[n]$ and for each $\widetilde{R}\in\{R,R^\dagger\}$ do the following.
   (a) For each Clifford C such that $P=CZ_{(q)}C^\dagger$. (If $P=Z_{(q)}$, we will skip this iteration for $Z_{(q)}$. We will discuss later which Cliffords to consider.)
      i. For each configuration $\overline{T}_{[n]\backslash\{q\}}$ do the following.
         A. $V\leftarrow\widetilde{R}(P)$.
         B. For each $i\in[n]\backslash\{q\}$ do the following.
            If $\overline{T}_{(i)}\neq\mathbb{I}$ then determine $P'=CZ_{(i)}C^\dagger$. If $\overline{T}=$ T then $V\leftarrow V\cdot R(P')$, else if $\overline{T}=$ T† then $V\leftarrow V\cdot R^\dagger(P')$.
         C. Include V in $\mathbb{V}_n$ if it did not already exist.

Cliffords to be considered (or not considered) at step 2(a). We have explained before that for our purpose, combinations of images obtained by conjugating $Z_{(i)}$ (ordering does not matter) is the most important, in order to have distinct unitaries. So we can make some choices of Cliffords to be considered (or rather, not to be considered) at step 2(a). For this, we can make some observations.

1. If $C\left(\prod_i\overline{T}_{(i)}\right)C^\dagger=\prod_j\widetilde{R}(Z_{(j)})$ for any $C\in\mathcal{C}_n$ then it is equal to the unitary $\left(\prod_j\overline{T}_{(j)}\right)$, even if the set of indices i and j are not same. Thus we have included each $\widetilde{R}(Z_{(j)})$ at step 1. Products of these also give T-depth 1 unitaries. In step 2(a) if $P=Z_{(q)}$ then we skip the iteration. In this loop we always consider those sequences of conjugations where there is at least one non-trivial mapping. So we always start with a non-trivial conjugation.
2. If $C=\otimes_i C_i$ for some Cliffords C_i then it is easy to see that we can write $U=C\left(\prod_j\overline{T}_{(j)}\right)C^\dagger=\prod_i C_i\left(\prod_{j_i}\overline{T}_{(j_i)}\right)C_i^\dagger=\prod_i U_i$,

Table 2. Performance of our algorithm on some benchmark circuit unitaries.

Unitary	#qubits	T-depth	T-count	Optimal?	Time	Max #nodes	Prev T-depth
Toffoli	3	3	7	Yes	27m 41s	358	3[18]
Fredkin	3	3	7	Yes	29m 49s	386	4[18]
Peres	3	3	7	Yes	27m 36s	358	4[18]
Quantum OR	3	3	7	Yes	27m 35s	358	4[18]
Negated Toffoli	3	3	7	Yes	27m 12s	358	4[18]

The T-depth returned by our algorithm (3rd column) is optimal for all unitaries. In most cases it is less than the Tdepth of the circuits shown in ref. [18]. We have also tabulated the T-count (4th column) of the T-depth-optimal circuits, running time (5th column) as well as the maximum number of nodes or intermediate unitaries (6th column) that accumulate at any level while running our algorithm. The running time excludes the pre-processing time to generate $\mathbb{V}_3$.

where each U_i has T-depth 1. So it is sufficient to consider each C_i and not C.

3. Let $U = C\left(\prod_{i=a}^{b} \overline{T}_{(i)}\right)C^\dagger$ is such that $CZ_{(j)}C^\dagger = Z_{(j)}$, where $a \le j \le b$. Then we can decompose $U = U_1U_2$ where U_1 excludes $T_{(j)}$ and $U_2 = T_{(j)}$ and each is of T-depth 1. This implies we should be concerned with the images of non-trivial conjugations (more reason to separate the trivial conjugations at step 1).

To determine the Cliffords to be considered we follow the mappings given in ref. [28]. Consider $i \in [n]$. First, we fix $2(4^n - 1)4^n$ Cliffords in $\mathcal{C}_n$ that conjugate $Z_{(i)}$ or $X_{(i)}$ non-trivially. We call these coset leaders of $Z_{(i)}$. The elements of $\mathcal{C}_n$ that conjugate $Z_{(i)}$ and $X_{(i)}$ trivially, form a group isomorphic to $\mathcal{C}_{n-1}$ with the number of cosets at most $2(4^n - 1)4^n$. For example, let $C \in \mathcal{C}_n$ is a coset leader (of $Z_{(i)}$) such that $CZ_{(i)}C^\dagger = P$ where $P \neq Z_{(i)}$, then any other Clifford that does the same conjugation (which is not a coset leader of $Z_{(i)}$) is of the form CC' where $C'Z_{(i)}C'^\dagger = Z_{(i)}$. In step 2(a) (when $q = i$) we consider all these coset leaders only. Suppose C is a coset leader that conjugates $Z_{(i)}$ to $P \neq Z_{(i)}$. In the loop 2(a) we considered all possible sequences of $R(P)$ or images obtained by conjugation of $Z_{(j)}$ ($j \neq i$) by C. Let C' is non-coset leader of $Z_{(i)}$ and does the trivial conjugation of $Z_{(i)}$. Now among all the $Z_{(j)}$ ($j \neq i$) where CC' conjugates non-trivially, it has to be the coset leader of one of them. This follows from the counting argument. So again we take all possible combinations of images obtained by conjugations by CC', when the loop starts with that particular position of T/T†.

Taking product. Now suppose $U_1 = C_1\left(\prod_i \overline{T}_{(i)}\right)C_1^\dagger \in \mathbb{V}_n$ and $U_2 = C_2\left(\prod_j \overline{T}_{(j)}\right)C_2^\dagger \in \mathbb{V}_n$, and there is no qubit such that a T/T†-gate is placed in both the unitaries. Let C_1 conjugates $Z_{(j)}$ trivially if j is a qubit in which there is a T/T† gate in U_2. Similarly $C_2Z_{(i)}C_2^\dagger = Z_{(i)}$, where there is a T/T† gate on qubit i in U_1. If $[C_1, C_2] = 0$ then it is easy to check that $U = U_1U_2 = C_1C_2\left(\prod_k \overline{T}_{(k)}\right)C_2^\dagger C_1^\dagger$ has T-depth 1. If $C_2Z_{(j)}C_2^\dagger = P_j$ and $C_1P_jC_1^\dagger = P_j$ then we do not need the commutation condition. It is straightforward to check that these conditions satisfy the 3 observations made earlier. (While constructing $\mathbb{V}_n$, we can store the information about which unitaries can be multiplied to have a T-depth 1 product.) Thus we can generate T-depth 1 unitaries (without trailing Clifford) by taking product of unitaries from $\mathbb{V}_n$.

Thus, from the above discussion, we can have the following result.

Theorem 2.1. *Any $U \in \mathcal{J}_n$ with T-depth 1 can be written as follows*: $U = e^{i\phi}\left(\prod_{i=d}^{1} V_i\right)C_0$, *where* $V_i \in \mathbb{V}_n$, $C_0 \in \mathcal{C}_n$ *and* $d \ge 1$.

Proof. We ignore the global phase and the trailing Clifford. Let $U = C\left(\prod_{i\in[n]} \overline{T}_{(i)}\right)C^\dagger$ (Eq. (3)). Let $S \subseteq [n]$ is the set of qubits such that C conjugates $Z_{(i)}$ trivially, where $i \in S$. Then we can write $U = \left(\prod_{i\in S}\widetilde{R}(Z_i)\right)C\left(\prod_{i\in\bar{S}}\overline{T}_{(i)}\right)C^\dagger = \left(\prod_{i\in S}\widetilde{R}(Z_i)\right)U'$. Each of these $\widetilde{R}(Z_{(i)})$ are included in $\mathbb{V}_n$ (step 1). So now let us consider the second term, U', in the product. If $C = \otimes_j C_j$ then we can write $U' = \prod_j C_j\left(\prod_{k\in S_j}\overline{T}_{(k)}\right)C_j^\dagger = \prod_j U_j'$, where $S_j \subseteq \bar{S}$ is the set of qubits on which C_j acts. If there are no T/T† gates at any qubit of S_j then $C_jC_j^\dagger = \mathbb{I}$. Else, there exists at least one $k \in S_j$ such that C_j conjugates $Z_{(k)}$ non-trivially. In step 2 of the definition of $\mathbb{V}_n$, we have included each such U_j in our set. This proves the theorem.

In ref. [26] it has been shown that $\{R(P) : P \in \mathcal{P}_n\}$ generates the T-count-optimal decomposition of any exactly implementable unitary, up to a Clifford. The channel representation inherits these decompositions and in this representation, the global phase goes away. Thus we can write the following:

$$\widehat{U} = \left(\prod_{i=d}^{1} \widehat{V}_i\right)\widehat{C_0} \tag{8}$$

Let

$$\widehat{\mathbb{V}_n} = \{\widehat{V'} : V' \in \mathbb{V}_n\}. \tag{9}$$

Fact 2.2 $|\mathbb{V}_n| \le 2n \cdot 3^{n-1} \cdot 4^n \cdot 4^n < n \cdot 2^{5.6n}$ and hence $|\widehat{\mathbb{V}_n}| < n \cdot 2^{5.6n}$.

Proof. From Definition 2.3, for each starting $R(P)/R^\dagger(P)$ there can be n positions for first T/T† gate respectively. In the remaining qubits we can have T, T† or $\mathbb{I}$. Thus there are at most 3^{n-1} ways to place the T/T† gates in remaining $(n-1)$ qubits. Given a starting Clifford and a configuration, the rest of the $R(P)$ unitaries are uniquely determined. We have discussed that we need to consider at most $2 \cdot 4^n \cdot 4^n$ Cliffords (coset leaders, as discussed before) that can map each $Z_{(i)}$ to any P[28–30]. More precisely, there are at most $2 \cdot 4^n \cdot 4^n$ choices for the starting Clifford for each of the n positions of the starting T/T† gate, which can lead to distinct strings of $R(P)$ during the construction of $\mathbb{V}_n$. So we get the stated bounds.

In Table 1 we have compared the cardinalities and generation time of $\mathbb{V}_n$ and $\mathcal{C}_n$. The latter has been used in[16] to design a T-depth-optimal-synthesis algorithm. We use the set $\mathbb{V}_n$ for our heuristic algorithm MIN-T-DEPTH. In the next section we use a bigger set with cardinality $O(4^{n^2})$, much less than $|\mathcal{C}_n| \in O(2^{kn^2})$, where $k > 2.5$. This set can be derived from $\mathbb{V}_n$, or we can simply use $\mathbb{V}_n'$. We will see in the following sections how the cardinalities of these sets make a difference in the running time and space of the various algorithms.

The following fact can be easily proved from Fact 3.2 in ref. [23].

Fact 2.3 Let $W' = \widetilde{R}(P)W$ where W and W' are unitaries, $\widetilde{R} \in \{R, R^\dagger\}$ and $P \in \pm\mathcal{P}_n$. Then $\mathrm{sde}(W') = \mathrm{sde}(W) \pm 1$ or $\mathrm{sde}(W') = \mathrm{sde}(W)$.

An $O(N^4)$ time algorithm for multiplying two $N^2 \times N^2$ unitaries $\widetilde{R(P)}$ and W (where $N = 2^n$) has been given in ref. [23]. This will help in computing $\widehat{\mathbb{V}_n}$ faster, but it will not make much

Table 3. Performance of MIN-T-DEPTH on random circuits.

#Qubits	Max. T-depth	Time (avg.) (s)	Time (std.) (s)	Max. #nodes (avg.)	Max. #nodes (std.)
2	2	0.015	0.006	2.90	3.45
	3	0.055	0.035	6.70	4.75
	4	0.184	0.153	21.1	20.1
	5	0.49	0.62	56.5	71.7
	6	1.17	0.91	99.9	73.0
	7	3.10	4.19	256.3	355.0
	8	8.39	8.36	443.0	435.2
	9	15.1	7.81	721.1	499.9
	10	38.1	32.5	1727.7	1282.7
	11	48.3	50.5	2049.7	1809.5
	12	47.6	59.5	1853.9	2218.8
	13	188.2	158.8	6705.8	5991.0
	14	547.1	921.0	12,293.9	16,351.1
	15	315.8	295.4	9515.6	7925.2
	16	238.3	169.1	7025.4	4905.8
	17	495.8	589.7	12,118.6	13,545.6
	18	408.3	265.3	9466.6	4937.4
	19	625.8	478.1	14,390.3	9332.9
	20	1008.2	656.6	15,313.2	9205.1
3	2	17.4	21.6	8.50	8.36
	3	209.4	190.8	123.5	173.0
	4	999.9	780.3	253.2	226.2
	5	3926.6	3424.7	1203.5	1968.8
	6	11,349.5	11,076.0	1024.1	643.1
	7	28,750.9	18,652.0	4481.3	4165.9

For each entry in the table, we generate 10 random circuits.

difference in the asymptotic complexity of any of our algorithms. So these are not essential for the rest of the paper.

Discussion of implementation results

We implemented our heuristic algorithm MIN-T-DEPTH (described in the section "Methods") in standard C++17 on an Intel(R) Core(TM) i7-7700K CPU at 4.2 GHz, with 8 cores and 16 GB RAM, running Debian Linux 9.13. We used OpenMP[34] for parallelization and the Eigen 3 matrix library[35] for some of the matrix operations. Our algorithm returns a T-depth-optimal decomposition of an input unitary. We can generate a circuit for each $R(P)$ using Fact 2.1 and the trailing Clifford using the algorithm in ref. [5]. We remind the reader that the numerical results of this subsection, together with instructions on how to reproduce them, are available online at https://github.com/vsoftco/t-depth. We have implemented MIN-T-DEPTH and not the optimal nested MITM algorithm because the former has better complexity.

We have synthesized T-depth-optimal circuits for three-qubit benchmark unitaries like Toffoli, Fredkin, Peres, Quantum OR, Negated Toffoli (Table 2). We found the min-T-depth of all these unitaries is 3, which is less than the T-depth of the circuits shown in ref. [16] (except Toffoli). The authors did not perform a T-depth-optimal synthesis of these 3 qubit circuits, since their algorithm required to generate a (pre-processed) set of more than 92,897,280 elements, which took more than 4 days (Table 1). The running time as well as space requirement, being an exponential (in min-T-depth) of this set, it would have been intractable on a PC. The largest T-depth-optimal circuit implemented in ref. [16] had 2-qubits and had T-depth 2. In our case the set generated during pre-processing is $\mathbb{V}_n$. In case of three qubits it has 2282 elements and takes about 2 s to be generated. The average searching time is 27.5 min. Thus our algorithm clearly outperforms the previously best T-depth-optimal synthesis algorithm in ref. [16].

We would like to mention here that for T-depth-optimal synthesis algorithms like[16] or ours, the input is a unitary matrix and no other additional information is provided. The T-depth of some unitaries may be related. For example, the authors have been pointed out that T-depth of Fredkin, Peres can be obtained from T-depth of Toffoli because they are Clifford equivalent. There are some concerns here. We do not know of any efficient test for Clifford equivalence given arbitrary exactly implementable unitaries. Second, we are unaware of any set of benchmark unitaries from which we can derive the T-depth of any exactly implementable unitary. In fact, these extra information can serve as litmus tests for the correctness of the output of any algorithm.

We have synthesized T-depth-optimal circuits for 2 and 3-qubit permutation unitaries. We found that all 2-qubit permutations are Cliffords. It took us, on average, 0.726 seconds to synthesize 2-qubit permutations. We considered about 100 random 3-qubit permutation unitaries and (due to time constraints) we synthesized completely (up to Clifford) the unitaries with T-depth at most 5. The permutations with T-depth at most 3 took on average 15 min. The permutations with T-depth at most 5 took on average 4.5 h.

We have also tested our algorithm on random 2 and 3 qubit circuits (Table 3). The input 2 and 3 qubit circuits had T-depth 2-10 and 2-7, respectively. Each line in Table 3 is computed from 10 random circuits. By Max.# nodes we mean the maximum number of unitaries selected at any level. 'avg' means we average this statistic over all unitaries considered. 'std' means we find the standard deviation of this statistic. We found out that the circuits output by our algorithm had T-depth at most of the input T-depth. Now the min-T-depth can be at most the input T-depth. We could not verify the optimality of our results, since we do not know of any T-depth-optimal synthesis algorithm that can implement such large circuits. However, this is a good indication that our algorithm MIN-T-DEPTH actually obtains the min-T-depth for most unitaries.

METHODS

A faster synthesis algorithm for T-depth

In this section, we describe an exact synthesis algorithm that finds a circuit that is provably T-depth-optimal. We modify the algorithm by Amy et al.[16] and employ a nested meet-in-the-middle technique, as has been done by Mosca and Mukhopadhyay[23], to optimize the T-count. This gives a more space-efficient algorithm to get optimal depth circuits. Furthermore, we work with channel representations to get T-depth-optimal circuits. This reduces both the time and space complexity compared to the algorithm in ref. [16].

An exact algorithm for depth-optimal circuits

We first describe a general algorithm where we are given a set of gates (and their inverses), $\mathcal{G}$, with which we want to design a depth optimal circuit implementing a unitary U. The set $\mathcal{G}$ is called the instruction set. Let $\mathcal{V}_{n,\mathcal{G}}$ be the set of n-qubit unitaries of depth 1 that can be implemented by a circuit designed with the gates in $\mathcal{G}$. We state the following lemma which can be regarded as a generalization of Lemma 1 in ref. [16]. This observation allows us to search for circuits of depth d by only generating circuits of depth at most $\lceil \frac{d}{c} \rceil$ $(c \geq 2)$.

Lemma 3.1. Let $S_i \subset U(2^n)$ be the set of all unitaries implementable in depth i over the gate set $\mathcal{G}$. Given a unitary U, there exists a circuit over $\mathcal{G}$ of depth $(d_1 + d_2)$ implementing U if and only if $S_{d_1}^{\dagger} U \bigcap S_{d_2} \neq \emptyset$.

Proof. We note that $U \in S_i^\dagger = \{U^\dagger | U \in S_i\}$ if and only if U can be implemented in depth i over $\mathcal{G}$. (Though this was proved in Lemma 1 of ref. [16] we include it briefly here for completion). Let $U = U_1 U_2 \ldots U_i$ where $U_1, U_2, \ldots, U_i \in \mathcal{V}_{n,\mathcal{G}}$ and so $U^\dagger = U_i^\dagger \ldots U_2^\dagger U_1^\dagger$. As $\mathcal{G}$ is closed under inversion so $U_1^\dagger, U_2^\dagger, \ldots, U_i^\dagger \in \mathcal{V}_{n,\mathcal{G}}$, and thus a circuit of depth i over $\mathcal{G}$ implements $U^\dagger$. Since $\left(S_i^\dagger\right)^\dagger = S_i$ the reverse direction follows.

Suppose U is implementable by a circuit C of depth $d_1 + d_2$. We divide C into two circuits of depth d_1 and d_2, implementing unitaries $W_1 \in S_{d_1}$ and $W_2 \in S_{d_2}$ respectively, where $W_1 W_2 = U$. So $W_2 = W_1^\dagger U \in S_{d_1}^\dagger U$ and hence $W_2 \in S_{d_1}^\dagger U \bigcap S_{d_2}$.

In the other direction let $S_{d_1}^\dagger U \bigcap S_{d_2} \neq \emptyset$. So there exists some $W_2 \in S_{d_1}^\dagger U \bigcap S_{d_2}$. Since $W_2 \in S_{d_1}^\dagger U$ so $W_2 = W_1^\dagger U$ for some $W_1 \in S_{d_1}^\dagger$. Now $W_2 \in S_{d_2}$ and $W_1 W_2 = U$. Thus U is implementable by some circuit of depth $d_1 + d_2$.

We now describe our procedure (Nested MITM), whose pseudocode has been given Algorithm 1. The input consists of the unitary U, instruction set $\mathcal{G}$, depth d and $c \geq 2$ that indicates the extent of nesting or recursion we want in our meet-in-the-middle approach. If U is of depth at most d then the output consists of a decomposition of U into smaller depth unitaries, else the algorithm indicates that U has depth more than d. At the beginning of the algorithm we generate the set $\mathcal{V}_{n,\mathcal{G}}$.

Algorithm 1. Nested MITM

Input: (i) A unitary U, (ii) a gate set $\mathcal{G}$, (iii) depth d, (iv) $c \geq 2$
Output: A circuit (if it exists) for U such that depth is at most d.
1 Generate the set $\mathcal{V}_{n,\mathcal{G}}$ of n-qubit unitaries with depth 1. ;
2 $S_0 \leftarrow \{\mathbb{I}\}$; $i \leftarrow 1$;
3 **while** $i \leq \left\lceil \frac{d}{c} \right\rceil$ **do**
4 $S_i \leftarrow \mathcal{V}_{n,\mathcal{G}} S_{i-1}$;
5 $k = c - 1$;
6 **for** $W = W_1 W_2 \ldots W_k$ *where* $W_i \in S_i$ *or* $W_i \in S_{i-1}$ **do**
7 **if** $\exists W' \in S_i$ *such that* $W^\dagger U = W'$ **then**
8 **return** $W_1, W_2, \ldots, W_k, W'$;
9 break ;
10 **end**
11 **else if** $\exists W' \in S_{i-1}$ *such that* $W^\dagger U = W'$ **then**
12 **return** $W_1, W_2, \ldots, W_k, W'$;
13 break ;
14 **end**
15 **end**
16 $i \leftarrow i + 1$;
17 **end**
18 **if** *no decomposition found* **then**
19 **return** "U has depth more than d." ;
20 **end**

The algorithm consists of $\left\lceil \frac{d}{c} \right\rceil$ iterations and in the ith such iteration we generate circuits of depth i (S_i) by extending the circuits of depth $i-1$ (S_{i-1}) by one more level. Then we use these two sets to search for circuits of depth at most ci. The search is performed iteratively where in the kth ($1 \leq k \leq c-1$) round we generate unitaries of depth at most ki by taking k unitaries $W_1, W_2, \ldots, W_k$ where $W_i \in S_i$ or $W_i \in S_{i-1}$. Let $W = W_1 W_2 \ldots W_k$ and its depth is $k' \leq ki$. We search for a unitary W' in S_i or S_{i-1} such that $W^\dagger U = W'$. By Lemma 3.1 if we find such a unitary it would imply that depth of U is $k' + i$ or $k' + i - 1$, respectively. In the other direction if the depth of U is either $k' + i$ or $k' + i - 1$ then there should exist such a unitary W' in S_i or S_{i-1}, respectively. Thus if the depth of U is at most d then the algorithm terminates in one such iteration and returns a decomposition of U. This proves the correctness of this algorithm.

Time and space complexity. We impose a strict lexicographic ordering on unitaries such that a set S_i can be sorted with respect to this ordering in $O(|S_i| \log |S_i|)$ time and we can search for an element in this set in $O(\log |S_i|)$ time. An example of such an ordering is ordering two unitaries according to the first element in which they differ. Now consider the kth round of the ith iteration (steps 3–17 of Algorithm 1). We build unitaries W of depth at most ki using elements from S_i or S_{i-1}. Number of such unitaries is at most $|S_i|^k$. Given a W, time taken to search for W' in S_i or S_{i-1} such that $W^\dagger U = W'$ is $O(\log |S_i|)$. Since $|S_i| \leq |\mathcal{V}_{n,\mathcal{G}}|^i$, so the kth iteration of the for loop within the ith iteration of the while loop, takes time $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{(c-1)i} \log |\mathcal{V}_{n,\mathcal{G}}|\right)$. Thus the time taken by the algorithm is $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{(c-1)\lceil \frac{d}{c} \rceil} \log |\mathcal{V}_{n,\mathcal{G}}|\right)$.

In the algorithm we store unitaries of depth at most $\lceil \frac{d}{c} \rceil$. So the space complexity of the algorithm is $O\left(|\mathcal{V}_{n,\mathcal{G}}|^{\lceil \frac{d}{c} \rceil}\right)$. Since $|\mathcal{V}_{n,\mathcal{G}}| \in O(|\mathcal{G}|^n)$, so we have an algorithm with space complexity $O\left(|\mathcal{G}|^{n\lceil \frac{d}{c} \rceil}\right)$ and time complexity $O\left(n|\mathcal{G}|^{n(c-1)\lceil \frac{d}{c} \rceil} \log |\mathcal{G}|\right)$.

Reducing both space and time complexity to find T-depth optimal circuits

We now consider the special case where $\mathcal{G}$ is the Clifford+T gate set and the goal is to design a T-depth optimal circuit for a given unitary U. We work with the channel representation of unitaries. We generate the set $\mathbb{V}_n''$, which consists of products of unitaries from $\mathbb{V}_n$ and has T-depth 1. We have explained in Section T-depth how to perform such products. We can even use $\mathbb{V}_n'$ described in the previous section. In Section T-depth we gave conditions for generating these products. Thus we replace $\mathcal{V}_{n,\mathcal{G}}$ with $\widehat{\mathbb{V}_n''}$. It is easy to see that for any T-depth 1 unitary $\widehat{U}$ there exists $\widehat{V} \in \widehat{\mathbb{V}_n''}$ such that $\widehat{U} = \widehat{V}\widehat{C}$ for some Clifford $C \in \mathcal{C}_n$. This motivates us to use the following definition from ref. [26].

Definition 3.1 (Coset label). Let $W \in \widehat{\mathcal{T}_n}$. Its coset label $W^{(co)}$ is the matrix obtained by the following procedure. (1) Rewrite W so that each nonzero entry has a common denominator, equal to $\sqrt{2}^{\text{sde}(W)}$. (2) For each column of W, look at the first non-zero entry (from top to bottom) which we write as $v = \frac{a+b\sqrt{2}}{\sqrt{2}^{\text{sde}(W)}}$. If $a < 0$, or if $a = 0$ and $b < 0$, multiply every element of the column by -1. Otherwise, if $a > 0$, or $a = 0$ and $b > 0$, do nothing and move on to the next column. (3) After performing this step on all columns, permute the columns so that they are ordered lexicographically from left to right.

Since unitaries stored in $\mathbb{V}_n''$ are distinct, we can say that this set stores the coset labels of T-depth 1 unitaries. The following can be shown.

Theorem 3.1 (Proposition in ref. [26]). *Let* $W, V \in \widehat{\mathcal{T}_n}$. *Then* $W^{(co)} = V^{(co)}$ *if and only if* $W = VC$ *for some* $C \in \widehat{\mathcal{C}_n}$.

The nested meet-in-the-middle search for T-depth-optimal circuit is performed as described before, except for the following changes. We replace the set $\mathcal{V}_{n,\mathcal{G}}$ with the set $\widehat{\mathbb{V}_n''}$ (step 4 of Algorithm 1), for reasons described before. This helps us to generate coset labels of unitaries with increasing T-depth. We work with channel representations $\widehat{W}, \widehat{W_1}, \widehat{W_2}, \ldots$. So at the kth round of the ith iteration, we calculate $\widehat{W} = \prod_{j=1}^k \widehat{W_j}$ where $\widehat{W_j} \in S_i$ or S_{i-1}. Then we check if $\exists \widehat{W'} \in S_i$ (or S_{i-1} respectively) such that $\left(\widehat{W}^\dagger \widehat{U}\right)^{(co)} = \widehat{W'}$. If such a unitary exists it would imply $U = e^{i\phi} W W' C$ for some Clifford $C \in \mathcal{C}_n$. From Lemma 3.1 we can say that U can be implemented by a circuit

with T-depth equal to the sum of the T-depth of the circuit for W and W'.

Space and time complexity. From Fact 2.2 we know that $|\widehat{\mathbb{V}_n}| \leq n \cdot 2^{5.6n}$ and $\mathbb{V}_n''$ is formed by taking the product of unitaries from $\mathbb{V}_n$. The most naive upper bound that we can have is $|\mathbb{V}_n''| \in O(4^{n^2})$, which is the bound on $\mathbb{V}_n'$ discussed in the section "T-depth". (We believe that $|\mathbb{V}_n''|$ is much less than 4^{n^2}.) Thus analyzing in the same way as before we can say that the algorithm has space complexity $O((4^{n^2})^{\lceil \frac{d}{c} \rceil})$ and time complexity $O((4^{n^2})^{(c-1)\lceil \frac{d}{c} \rceil})$ ($c \geq 2$). This is much less than the space and time complexity of the T-depth-optimal algorithm in ref. [16]. They use the MITM technique and the space and time complexity is $O((3^n|\mathcal{C}_n|)^{\lceil \frac{d}{2} \rceil} \cdot |\mathcal{C}_n|)$. The cardinality of the n-qubit Clifford group, $\mathcal{C}_n$, is $O(2^{kn^2})$ ($k > 2.5$)[28,29]. So the space and time complexity is $O((2^{kn^2})^{\lceil \frac{d}{2} \rceil + 1} 3^{n\lceil \frac{d}{2} \rceil})$, where $k > 2.5$. Clearly, even if the extent of nesting is 2 i.e. $c = 2$, in which case our procedure becomes a MITM algorithm, we get a significant improvement in both time and space complexity.

A more efficient algorithm to synthesize *T*-depth optimal circuits

In this section, we describe an algorithm that on input a $2^n \times 2^n$ unitary U finds a T-depth optimal circuit for it and has space and time complexity poly$(n, 2^{5.6n}, d)$ with some conjecture (or poly$(n^{\log n}, d, 2^{5.6n})$ with a weaker conjecture), where d is the min-T-depth of U. We draw inspiration from some observations made in ref. [23], while developing a polynomial time algorithm for synthesizing T-count-optimal circuits. We came up with another way of pruning the search space. The numerical results of this section (Tables 2 and 3) are available online at https://github.com/vsoftco/t-depth.

Algorithm 2. $\mathcal{A}$

Input: (i) $\widehat{U}$ where $U \in \mathcal{J}_n$ and (ii) integer $d' > 0$
Output: Set $\mathcal{D}$ of decompositions $\widehat{U} = \left(\prod_{i=k}^{1} \widehat{R(P)}\right) \widehat{C_0}$ if minimum T-depth at most d'

Compute sets $\mathbb{V}_{n,j}^{-1}$ for $j = 1, 2, \ldots n$. // Can do pre-processing ;
$\text{Path}_{\widehat{U}} \leftarrow []$; $\mathcal{D} \leftarrow \emptyset$;
$\widetilde{U} = [\widehat{U}, \text{Path}_{\widehat{U}}, \text{sde}_{\widehat{U}}, \text{ham}_{\widehat{U}}]$ // Stores root node ;
$\mathcal{U}_0 = \{\widetilde{U}\}$ // Subscript indicates path T-count ;
for $i = 1, 2, \ldots, d'$ **do**
  $SH \leftarrow \emptyset$ // Stores tuples (sde, Hamming weight change, path T-count, count) ;
  $P \leftarrow \emptyset$ // Stores the path T-counts obtained ;
  for $k = i-1, \ldots, n(i-1)$ // Parent hypernodes// **do**
    for $\widetilde{U} \in \mathcal{U}_k$ // Each node within a hypernode// **do**
      for $j = 1, 2, \ldots, n$ **do**
        for *each* $\widehat{V}^{-1} \in \mathbb{V}_{n,j}^{-1}$ **do**
          $\widehat{W} \leftarrow \widehat{V}^{-1}\widehat{U}$;
          $\Delta\text{ham}_{\widehat{W}} =$ inc, dec or same if $\text{ham}_{\widehat{W}} > \text{ham}_{\widehat{U}}$, $\text{ham}_{\widehat{W}} < \text{ham}_{\widehat{U}}$ or $\text{ham}_{\widehat{W}} = \text{ham}_{\widehat{U}}$ respectively ;
          $P.append(PathT_{\widehat{W}})$ if $PathT_{\widehat{W}} \notin P$ // $PathT_{\widehat{W}}$ is the Path T-count;
          if $(sde_{\widehat{W}}, \Delta ham_{\widehat{W}}, PathT_{\widehat{W}}, count) \in SH$ **then**
            $(\text{sde}_{\widehat{W}}, \Delta\text{ham}_{\widehat{W}}, PathT_{\widehat{W}}, count) \leftarrow (\text{sde}_{\widehat{W}}, \Delta\text{ham}_{\widehat{W}}, PathT_{\widehat{W}}, count + 1)$;
          else
            $SH.append((\text{sde}_{\widehat{W}}, \Delta\text{ham}_{\widehat{W}}, PathT_{\widehat{W}}, 1))$;
          end
        end
      end
    end
  end
  for $p \in P$ **do**
    $(s, \Delta h, p, count) \leftarrow$ Element of SH such that $count$ is minimum, $s \leq d' - i - 1$ and Path T-count is p ;
    $G \leftarrow$ Product unitaries such that $\text{sde}_{\widehat{W}} = s$, $\Delta\text{ham}_{\widehat{W}} = \Delta h$ and Path T-count is p // Can be obtained by re-computing those unitaries ;
  end
  for $\widehat{U'} \in G$ **do**
    if $sde_{\widehat{U'}} == 0$ **then**
      $\widehat{C_0} = \widehat{U'}$;
      Store $\text{Path}_{\widehat{U'}}, \widehat{C_0}$ as a decomposition of $\widehat{U}$ and i as its depth in $\mathcal{D}$;
    else
      $\widetilde{U'} = [\widehat{U'}, \text{Path}_{\widehat{U'}}, \text{sde}_{\widehat{U'}}, \text{ham}_{\widehat{U'}}]$ // Build children node ;
      $\mathcal{U}_k$.append$(\widetilde{U'})$, where k is the path T-count of $\widehat{U'}$ // Store in hypernodes ;
    end
  end
  Delete all parent nodes;
end
return $\mathcal{D}$;

Algorithm 3. MIN T-DEPTH

Input: $\widehat{U}$ where $U \in \mathcal{J}_n$
Output: d, the minimum T-depth of U and its decomposition $\widehat{U} = \left(\prod_{i=k}^{1} \widehat{R(P)}\right) \widehat{C_0}$

$d' = \left\lceil \frac{\text{sde}\widehat{U}}{n} \right\rceil$ or $\left\lceil \frac{\mathcal{T}(U)}{n} \right\rceil$ if $\mathcal{T}(U)$ is known;
while 1 **do**
  $\mathcal{A}(\widehat{U}, d')$;
  if $\mathcal{D} == \emptyset$ **then**
    $d' \leftarrow d' + 1$;
  else
    In each decomposition check if consecutive unitaries can be combined to form a T-depth 1 product ;
    return a decomposition with the minimum T-depth ;
    break ;
  end
end

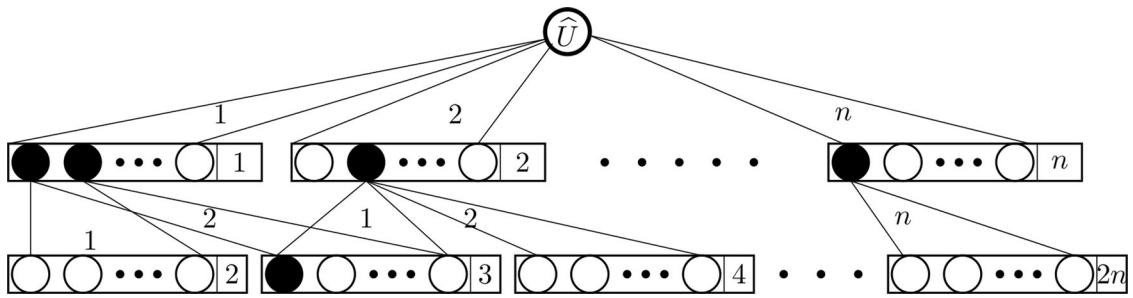

Fig. 1 The tree built in $\mathcal{A}$ (Algorithm 2). Each node stores a unitary, the root at level 0 storing $\widehat{U}$. The edges are labeled by unitaries in $\widehat{\mathbb{V}_n}^{-1}$. A child node unitary is obtained by multiplying the edge unitary with the parent node unitary. The edges are grouped into hyper-edges, where each hyper-edge is labeled by a unitary in $\widehat{\mathbb{V}_{nj}}^{-1}$. The nodes are grouped into hyper-nodes, where each hyper-node has a number indicating the number of $\widehat{R(P)}^{-1}$ in the path from the root to each node in this hyper-node. Within each hyper-node we select some nodes according to some criteria and the nodes in the next level are built from these selected (black) nodes.

The input of our algorithm is the channel representation of a $2^n \times 2^n$ unitary U. From Theorem 2.1 we know there exists a T-depth-optimal decomposition of $\widehat{U}$ as follows : $\widehat{U} = \left(\prod_{i=d''}^{1} \widehat{V}_i\right)\widehat{C_0}$, where $C_0 \in \mathcal{C}_n$, $\widehat{V}_i \in \mathbb{V}_n$, $d \le d'' \le dn$ and d is the T-depth of U. We iteratively try to guess the blocks $\widehat{V}_i$ by looking at the change in some 'properties' of the matrix $\widehat{V}_i^{-1}\widehat{U'}$ where $\widehat{U'} = \prod_{j=d''}^{i+1} \widehat{V}_j^{-1}\widehat{U}$. If we have the correct sequence then we should reach $\widehat{C_0}$, a matrix consisting of exactly one $+1$ or -1 in each row and column. As in ref. [23] we consider two properties of the resultant matrices—their sde and Hamming weight. The intuition is as follows. Consider a unitary $\widehat{W}$ and we multiply it by $\widehat{V_1} \in \widehat{\mathbb{V}_n}$. Let $\widehat{Y} = \widehat{WV_1}$, $\Delta_s = \text{sde}(\widehat{W}) - \text{sde}(\widehat{Y})$ and $\Delta_h = \text{ham}(\widehat{W}) - \text{ham}(\widehat{Y})$, where ham(.) is the Hamming weight. Now we multiply $\widehat{Y}$ by $\widehat{V_i}^{-1}$ where $\widehat{V_i} \in \widehat{\mathbb{V}_n}$. Let $\widehat{Z} = \widehat{YV_i^{-1}}$, $\Delta_s^i = \text{sde}(\widehat{Y}) - \text{sde}(\widehat{Z})$ and $\Delta_h^i = \text{ham}(\widehat{Y}) - \text{ham}(\widehat{Z})$. If $V_i = V_1$ then $\Delta_s = -\Delta_s^i$ and $\Delta_h = -\Delta_h^i$. But if $V_i \neq V_1$ then with high probability we do not expect to see this kind of change. This helps us to distinguish the V_i's in at least one T-depth-optimal decomposition.

The pseudocode for algorithm MIN T-DEPTH has been given in Algorithm 3. We iteratively call the sub-procedure $\mathcal{A}(\widehat{U}, d')$ with the value $d' \in \mathbb{Z}$ increasing in each iteration. We accumulate all decompositions returned by $\mathcal{A}$. Then in MIN T-DEPTH we check if in each such decomposition we can combine consecutive unitaries to form a T-depth 1 unitary (refer to the section "T-depth"). We output a decomposition with the minimum T-depth. Here let us explain the starting value for d'. If we know that any circuit requires at least x T gates to implement U, we know that the T-depth of any circuit implementing U will be at least $\lceil \frac{x}{n} \rceil$. Thus if we know $\mathcal{T}(U)$ i.e. the T-count of U we can start the iterations with $d' = \lceil \frac{\mathcal{T}(U)}{n} \rceil$. If we do not know that, we can consider $\text{sde}(\widehat{U})$. Due to Fact 2.3 we know $\mathcal{T}(U) \ge \text{sde}(\widehat{U})$, so we can also start the iterations with $d' = \lceil \frac{\text{sde}(\widehat{U})}{n} \rceil$. We can also determine stopping criteria from these information. For example, if we get a decomposition with T-depth $\lceil \frac{\mathcal{T}(U)}{n} \rceil$, then we can stop immediately. Alternatively, we can generate the set $\mathbb{V}_n''$, described for our nested-MITM algorithm and stop as soon as we get a decomposition in $\mathcal{A}$.

It will be useful if we depict the procedure $\mathcal{A}$ using a tree (Fig. 1), where each node stores a unitary. The root (depth 0) stores $\widehat{U}$. The edges are labeled by unitaries from $\widehat{\mathbb{V}_n}^{-1}$, which is defined as

$$\widehat{\mathbb{V}_n}^{-1} = \{\widehat{V}^{-1} : \widehat{V} \in \widehat{\mathbb{V}_n}\}$$

This is a set of n-qubit unitaries with T-depth 1 (refer to the section "T-depth"). A child node unitary is obtained by multiplying the parent unitary with the unitary of the edge. We refer to these two types of unitaries as 'node-unitary' and 'edge-unitary' respectively. The product of the edge unitaries on a path from the root to a non-root node is referred to as the 'path unitary' with respect to the non-root node. By 'path T-count' of a non-root node, we refer to the sum of the number of $R(P)$ terms in the edge-unitaries. Each $R(P)$ has one T-gate. At each depth of the tree, we group the nodes into some 'hypernodes' such that the path T-count of each node within a hypernode is the same. At this point it will be useful to observe $\widehat{\mathbb{V}_n}^{-1} = \bigcup_{1 \le j \le n} \widehat{\mathbb{V}_{nj}}^{-1}$,

where $\widehat{\mathbb{V}_{n,j}}^{-1}$ is the set of unitaries with j number of $\widehat{R(P)}^{-1}$. In Fig. 1 we have grouped the edges such that the edge-unitaries within one such 'hyperedge' are from $\widehat{\mathbb{V}_{n,j}}^{-1}$ for some j.

At each depth, within each such hypernode we sub-divide the nodes according to the sde of its unitary and change in Hamming weight of this unitary compared to the parent node-unitary. By change in Hamming weight we mean if it has increased or decreased or remains unchanged, with respect to the Hamming weight of the parent node. Within each hypernode we select the set of nodes with minimum cardinality such that sde of its unitaries can be reduced to 0 within depth d' of the tree. We build the nodes in the next level from the 'selected' node-unitaries only. We stop building the tree as soon as we reach a node-unitary with sde 0, indicating we reached a Clifford. If we have not reached any Clifford within depth d' we quit and conclude that minimum T-depth of $\widehat{U}$ is more than d'. A pseudocode of the procedure $\mathcal{A}$ has been given in Algorithm 2. The number of hypernodes in depth i can be at most $ni - i + 1$, since the path T-count of any unitary can be at most ni and at least i. Also, since the sde can change by at most 1 after multiplying by any $\widehat{R(P)}^{-1}$ (Fact 2.3), then after multiplying by any unitary in $\mathbb{V}_{n,j}^{-1}$ sde of any unitary can change by at most j. So (at step 25 of Algorithm 2) we select the minimum sized set among those sets of unitaries which has the potential to reach the Clifford within the remaining steps.

To analyze the space and time complexity of our algorithm we make the following conjecture.

Conjecture 1
(a) While dividing the nodes according to their sde and change in Hamming weight within any hypernode, the minimum cardinality of any set (such that its sde can be potentially reduced to 0) is bounded by poly(2^n). (b) Also, we get at least one T-depth-optimal decomposition.

So our conjecture has two parts. (a) bounds the size of the tree and thus determines the complexity of the algorithm. (b) implies that we can preserve at least one T-depth-optimal decomposition by pruning in this way. So it determines the efficiency. We can make a weaker conjecture with a more relaxed bound.

Conjecture 2
(Weaker version) (b) While dividing the nodes according to their sde and change in Hamming weight within any hypernode, the minimum cardinality of any set (such that its sde can be potentially reduced to 0) is bounded by poly($n^{\log n}, 2^n$). (b) Also, we get at least one T-depth-optimal decomposition.

Comparison with Conjecture 1 in ref. [23]. In ref. [23] the authors proposed some conjectures to reduce the complexity of synthesizing T-count-optimal circuits. Our algorithm has been motivated by that work but based on current knowledge it does not appear that Conjectures 1 or 2 can be derived from the conjecture used in ref. [23], with the present knowledge. The main intuition of these conjectures stems from the following observation. Suppose we multiply a unitary $\widehat{U}'$ by $\widehat{R(P_1)}$. We will notice some change in the properties (like sde, Hamming weight) in the product unitary $\widehat{W}' = \widehat{U}'\widehat{R(P_1)}$ compared to the initial $\widehat{U}'$. Now when we multiply $\widehat{W}'$ by $\widehat{R(P_1)}^{-1}$ we will see these effects reversed. But if we multiply $\widehat{W}'$ by $\widehat{R(P_i)}^{-1}$ (where $i \neq 1$) then with high probability we will observe some other effects. In ref. [23] the authors used these intuitions to design a T-count-optimal algorithm, where they iteratively tried to guess a sequence of $R(P)$s in a T-count-optimal decomposition of U, by observing these change in properties. In our present algorithm (see Fig. 1) we consider many paths with different T-counts at each level. Now the T-depth-optimal decompositions will follow some of these paths. When we select the minimum cardinality set in each hypernode (where all unitaries have the same path T-count), we expect that the distinguishing property that we explained before does not get destroyed even if we multiply an (intermediate) unitary by up to $n\widehat{R(P)}^{-1}$. We do not see how this observation follows from the conjecture in ref. [23], without some more knowledge about the underlying mathematics. So for T-depth-optimal decompositions, we have made separate conjectures.

Space and time complexity. We consider the time and space complexity of $\mathcal{A}$. From Fact 2.2 we know $|\widehat{\mathbb{V}_n}^{-1}| \leq n \cdot 2^{5.6n}$. These are the number of unitaries we always store.

In the ith iteration we have up to $ni-i+1$ children hypernodes. There are at most $n(i-1)-(i-1)+1$ parent hypernodes and within each at most poly(2^n) parent nodes are selected by Conjecture 1. Each parent node is multiplied by $|\widehat{\mathbb{V}_n}|2^{2n} \times 2^{2n}$ unitaries. Arguing in similar way space and time complexity of procedure $\mathcal{A}$ is poly$\left(d', n, 2^{5.6n}\right)$.

Since MIN T-DEPTH consists of at most dn iterations of $\mathcal{A}$, where d is the minimum T-depth of U, so space and time complexity is poly$\left(d, n, 2^{5.6n}\right)$.

If we assume the weaker Conjecture 2 then we get a space and time complexity poly$\left(d, n^{\log n}, 2^{5.6n}\right)$.

DATA AVAILABILITY

Numerical results together with instructions on how to reproduce them, are available online at https://github.com/vsoftco/t-depth.

CODE AVAILABILITY

The code is available from the corresponding author on request.

Received: 24 May 2021; Accepted: 23 August 2022;
Published online: 13 September 2022

REFERENCES

1. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys* **21**, 467–488 (1982).
2. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**, 303–332 (1999).
3. Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, 124–134 (IEEE, 1994).
4. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual Symposium on Theory of Computing*, 212–219 (ACM, 1996).
5. Aaronson, S. & Gottesman, D. Improved simulation of stabilizer circuits. *Phys. Rev. A* **70**, 052328 (2004).
6. Bravyi, S. & Kitaev, A. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Phys. Rev. A* **71**, 022316 (2005).
7. Fowler, A. G., Stephens, A. M. & Groszkowski, P. High-threshold universal quantum computation on the surface code. *Phys. Rev. A* **80**, 052312 (2009).
8. Aliferis, P., Gottesman, D. & Preskill, J. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Inf. Comput.* **6**, 97–165 (2006).
9. Britton, J. W. et al. Engineered two-dimensional Ising interactions in a trapped-ion quantum simulator with hundreds of spins. *Nature* **484**, 489 (2012).
10. Brown, K. R. et al. Single-qubit-gate error below 10^{-4} in a trapped ion. *Phys. Rev. A* **84**, 030303 (2011).
11. Chow, J. M. et al. Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits. *Phys. Rev. Lett.* **109**, 060501 (2012).
12. Rigetti, C. et al. Superconducting qubit in a waveguide cavity with a coherence time approaching 0.1 ms. *Phys. Rev. B* **86**, 100506 (2012).
13. Bombin, H., Andrist, R. S., Ohzeki, M., Katzgraber, H. G. & Martín-Delgado, M. A. Strong resilience of topological codes to depolarization. *Phys. Rev. X* **2**, 021004 (2012).

14. Fowler, A. G., Whiteside, A. C. & Hollenberg, L. C. L. Towards practical classical processing for the surface code. *Phys. Rev. Lett.* **108**, 180501 (2012).
15. Fowler, A. G. Time-optimal quantum computation. Preprint at https://arXiv.org/quant-ph/1210.4626 (2012).
16. Amy, M., Maslov, D., Mosca, M. & Roetteler, M. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **32**, 818–830 (2013).
17. Amy, M., Maslov, D. & Mosca, M. Polynomial-time T-depth optimization of Clifford +T circuits via matroid partitioning. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **33**, 1476–1489 (2014).
18. Amy, M. et al. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In *Int. Conf. on Selected Areas in Cryptography*, 317–337 (Springer, 2016).
19. Di Matteo, O., Gheorghiu, V. & Mosca, M. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Trans. Quantum Eng.* **1**, 1–13 (2020).
20. Dawson, C. M. & Nielsen, M. A. The Solovay–Kitaev algorithm. *Quantum Inf. Comput.* **6**, 81–95 (2006).
21. Giles, B. & Selinger, P. Exact synthesis of multiqubit Clifford+T circuits. *Phys. Rev. A* **87**, 032332 (2013).
22. de Brugière, T. G., Baboulin, M., Valiron, B. & Allouche, C. Quantum circuits synthesis using Householder transformations. *Comput. Phys. Commun.* **248**, 107001 (2020).
23. Mosca, M. & Mukhopadhyay, P. A polynomial time and space heuristic algorithm for T-count. *Quantum Sci. Technol.* **7**, 015003 (2021).
24. Häner, T. & Soeken, M. Lowering the T-depth of quantum circuits by reducing the multiplicative depth of logic networks. Preprint at https://arXiv.org/quant-ph/2006.03845 (2020).
25. Kitaev, A. Y. Quantum computations: algorithms and error correction. *Russ. Math. Surv.* **52**, 1191 (1997).
26. Gosset, D., Kliuchnikov, V., Mosca, M. & Russo, V. An algorithm for the T-count. *Quantum Inf. Comput.* **14**, 1261–1276 (2014).
27. Ross, N. J. & Selinger, P. Optimal ancilla-free Clifford+T approximation of Z-rotations. *Quantum Inf. Comput.* **16**, 901–953 (2016).
28. Ozols, M. Clifford group. *Essays at University of Waterloo* (Springer, 2008).
29. Koenig, R. & Smolin, J. A. How to efficiently select an arbitrary Clifford group element. *J. Math. Phys.* **55**, 122202 (2014).
30. Calderbank, A. R., Rains, E. M., Shor, P. M. & Sloane, N. J. A. Quantum error correction via codes over GF(4). *IEEE Trans. Inf. Theory* **44**, 1369–1387 (1998).
31. Gheorghiu, V., Mosca, M. & Mukhopadhyay, P. T-count and T-depth of any multi-qubit unitary. Preprint at https://arXiv.org/quant-ph/2110.10292 (2021).
32. Di Matteo, O. & Mosca, M. Parallelizing quantum circuit synthesis. *Quantum Sci. Technol.* **1**, 015003 (2016).
33. Cowtan, A., Dilkes, S., Duncan, R., Simmons, W. & Sivarajah, S. Phase gadget synthesis for shallow circuits. In *16th Int. Conf. on Quantum Physics and Logic*, 213–228 (Open Publishing Association, 2019).
34. *The OpenMP API Specification for Parallel Programming*. https://www.openmp.org/ (2021).
35. *Eigen: A C++ Template Library for Linear Algebra*. http://eigen.tuxfamily.org (2021).

ACKNOWLEDGEMENTS

The authors wish to thank NTT Research for their financial and technical support. This work was supported in part by Canada's NSERC. IQC and the Perimeter Institute (PI) are supported in part by the Government of Canada and the Province of Ontario (PI). We thank the anonymous reviewers for their comments, that not only helped us improve the write-up significantly, but also led to a tighter bound in Fact 2.2.

AUTHOR CONTRIBUTIONS

The ideas were given by P.M. The software implementations were done by V.G. All the authors have made substantial contributions to the preparation of the manuscript.

COMPETING INTERESTS

The authors declare no competing non-financial interests but the following competing financial interests. M.M. is co-founder of softwareQ Inc. and has filed a provisional patent application for this work. P.M. is a co-inventor of this patent.

ADDITIONAL INFORMATION

Correspondence and requests for materials should be addressed to Priyanka Mukhopadhyay.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.