Quantum Information and Computation, Vol. 14, No. 15&16 (2014) 1261–1276

AN ALGORITHM FOR THE T-COUNT

DAVID GOSSET
Institute for Quantum Computing and Department of Combinatorics & Optimization
University of Waterloo, Waterloo, On N2L 3G1, Canada

VADYM KLIUCHNIKOV
Institute for Quantum Computing and David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, On N2L 3G1, Canada

MICHELE MOSCA
Institute for Quantum Computing and Department of Combinatorics & Optimization
University of Waterloo, Waterloo, On N2L 3G1, Canada
Perimeter Institute for Theoretical Physics, Waterloo, On N2L 3G1, Canada

VINCENT RUSSO
Institute for Quantum Computing and David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, On N2L 3G1, Canada

Received October 19, 2013
Revised March 19, 2014

We consider quantum circuits composed of Clifford and T gates. In this context the T gate has a special status since it confers universal computation when added to the (classically simulable) Clifford gates. However it can be very expensive to implement fault-tolerantly. We therefore view this gate as a resource which should be used only when necessary. Given an n-qubit unitary U we are interested in computing a circuit that implements it using the minimum possible number of T gates (called the T-count of U). A related task is to decide if the T-count of U is less than or equal to m; we consider this problem as a function of $N = 2^n$ and m. We provide a classical algorithm which solves it using time and space both upper bounded as $\mathcal{O}(N^m \text{poly}(m, N))$. We implemented our algorithm and used it to show that any Clifford+T circuit for the Toffoli or the Fredkin gate requires at least 7 T gates. This implies that the known 7 T gate circuits for these gates are T-optimal. We also provide a simple expression for the T-count of single-qubit unitaries.

Keywords:

Communicated by: R Cleve & R de Wolf

1 Introduction

The single-qubit T gate

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix},$$

along with all gates from the Clifford group, is a universal gate set for quantum computation. The T gate is essential because circuits composed of only Clifford gates are classically simulable [7]. The T gate also plays a special role in fault-tolerant quantum computation. In contrast with Clifford gates, the T gate is not transversal for many quantum error-correcting codes,

which means that in practice it is very costly to implement fault-tolerantly. For this reason we are interested in circuits which use as few T gates as possible to implement a given unitary.

We consider unitaries which can be implemented exactly using Clifford and T gates. The set of all such unitaries is known: it was conjectured in [10] and later proven in [6] that an n-qubit unitary can be implemented using this gate set if and only if its matrix elements are in the ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$. In general this implementation requires one ancilla qubit prepared in the state $|0\rangle$ in addition to the n qubits on which the computation is to be performed. An algorithm for exactly synthesizing unitaries over the gate set $\{H, T, \mathrm{CNOT}\}$ was given in reference [6], and a superexponentially faster version of this algorithm was presented in reference [9].

In this paper we focus on the set of unitaries implementable without ancillas, that is to say, the group $\mathcal{J}_n$ generated by the n-qubit Clifford group and the T gate. An n-qubit unitary is an element of this group if and only if its matrix elements are in the ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$ and its determinant satisfies a simple condition [6] (when $n \geq 4$ the condition is that the determinant is equal to 1). Notable examples include the Toffoli and Fredkin gates which are in the group $\mathcal{J}_3$.

For $U \in \mathcal{J}_n$, the T-count of U is defined to be the minimum number of T gates in a Clifford+T circuit that implements it, and is denoted by $\mathcal{T}(U)$. In other words $\mathcal{T}(U)$ is the minimum m for which

$$e^{i\phi}U = C_m T_{(q_m)} C_{m-1} T_{(q_{m-1})} \dots T_{(q_1)} C_0 \tag{1.1}$$

where $\phi \in [0, 2\pi)$, C_i are in the n-qubit Clifford group, $q_j \in \{1, \dots, n\}$, and $T_{(r)}$ indicates the T gate acting on the rth qubit.

Note that it may be possible to implement some unitary using less than $\mathcal{T}(U)$ T gates if one uses ancilla qubits and/or measurements with classically controlled operations. For example, Jones has shown how to perform a Toffoli gate using these additional ingredients and only four T gates [8]. This does not contradict our result that $\mathcal{T}(\text{Toffoli})=7$.

We are interested in the following problem. Given $U \in \mathcal{J}_n$, compute a T-optimal n-qubit quantum circuit for it, that is to say, a circuit which implements it using $\mathcal{T}(U)$ T gates. A related, potentially easier problem, is to compute $\mathcal{T}(U)$ given U. It turns out that this latter problem is not much easier: we show in Section 3 that an algorithm which computes $\mathcal{T}(U)$ can be converted into an algorithm which outputs a T-optimal circuit for U, with overhead polynomial in $\mathcal{T}(U)$ and the dimension of U. For this reason, and for the sake of simplicity, we focus on the task of computing $\mathcal{T}(U)$.

Problem 1 (COUNT-T). Given $U \in \mathcal{J}_n$ and $m \in \mathbb{N}$, decide if $\mathcal{T}(U) \leq m$.

We consider the complexity of this problem 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 present an algorithm which solves COUNT-T using time and space both upper bounded as $\mathcal{O}\left(N^m \mathrm{poly}(m, N)\right)$. Our algorithm uses the meet-in-the-middle idea from [3] along with a representation for Clifford+T unitaries where the T gates have a special role. We implemented our algorithm in C++ and used it to prove that the T-count of Toffoli and Fredkin gates is 7. We also provide a simple expression for the T-count of single-qubit unitaries.

The problem of computing T-optimal circuits was studied in references [2,3,10]. In reference [10] an algorithm was given for synthesizing single-qubit circuits over the gate set $\{H, T\}$, and it was shown that the resulting circuits are T-optimal. In this work we provide a simple expression for the T-count of a single qubit unitary which immediately results in an alternative algorithm for synthesizing T-optimal single-qubit circuits. In addition, we describe a new canonical form for single qubit unitaries over the Clifford+T gate set; other canonical forms for single qubit unitaries over this gate set have been studied in references [4,11]. The problem of reducing the number of T-gates in circuits with $n > 1$ qubits was considered as an application of the meet-in-the-middle algorithm from reference [3], where some small examples of T-depth optimal circuits were found. The algorithm for optimizing T-depth presented in reference [3] can be used (with a small modification) to solve COUNT-T but its time complexity is $\Omega\left(4^{n^2\lfloor m/2\rfloor}\right)$ since the size of the n-qubit Clifford group is $\Omega\left(4^{n^2}\right)$[5]. It was conjectured in reference [3] that Toffoli requires 7 T gates; we prove this conjecture in this paper. In reference [2] an algorithm based on matroid partitioning is given which can be used as a heuristic for minimizing the T-count and T-depth of quantum circuits. The algorithm we present here solves COUNT-T using time $\mathcal{O}\left(2^{nm}\mathrm{poly}\left(m, 2^n\right)\right)$, which is a superexponential speedup (as a function of n) over reference [3], and in contrast with the heuristic from [2] it computes the T-count exactly.

Our work, and previous work on the T-count, takes a different but complementary perspective from that of the recent paper [12]. That paper considers quantum states (e.g., magic states) as a resource for computation using Clifford circuits, and attempts to quantify the amount of resource in a given state. Here we view the T gate as a resource and ask how much is necessary to implement a given unitary.

The rest of the paper is structured as follows. We begin in Section 2 by discussing the central objects that we study: Pauli and Clifford operators, and the group $\mathcal{J}_n$ generated by Clifford and T gates. In Section 3 we give a decomposition for unitaries from the group $\mathcal{J}_n$. Using this decomposition we show how an algorithm for the T-count can be used to generate T-optimal circuits. In Section 4 we give a simple characterization of the T-count for single-qubit unitaries, and in Section 5 we present our algorithm for COUNT-T. We conclude in Section 6 with a discussion of open problems.

2 Preliminaries

In this Section we establish notation and review facts about the Clifford+T gate set. Throughout this paper we write $N = 2^n$. We write

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

for the single-qubit Pauli matrices. We use a parenthesized subscript to indicate qubits on which an operator acts, e.g., $X_{(1)} = X \otimes \mathbb{I}^{\otimes(n-1)}$ indicates the Pauli X matrix acting on the first qubit.

2.1 Cliffords and Paulis

The single-qubit Clifford group $\mathcal{C}_1$ is generated by the Hadamard and phase gates

$$\mathcal{C}_1 = \left\langle H, T^2 \right\rangle$$

where

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad T^2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

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 $= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X$ gate (acting on any pair of qubits). The Clifford group is special because of its relationship to the set of n-qubit Pauli operators

$$\mathcal{P}_n = \{Q_1 \otimes Q_2 \otimes \ldots \otimes Q_n : \ Q_i \in \{\mathbb{I}, X, Y, Z\}\}.$$

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 $P' \in \mathcal{P}_n$. It is easy to prove that, given two Paulis, neither equal to the identity, it is always possible to (efficiently) find a Clifford which maps one to the other. For completeness we include a proof in the Appendix.

Fact 1. *For any $P, P' \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ there exists a Clifford $C \in \mathcal{C}_n$ such that $CPC^\dagger = P'$. A circuit for C over the gate set $\{H, T^2, CNOT\}$ can be computed efficiently (as a function of n).*

2.2 The group $\mathcal{J}_n$ generated by Clifford and T gates

We consider the group $\mathcal{J}_n$ generated by the n-qubit Clifford group along with the T gate. For a single qubit

$$\mathcal{J}_1 = \langle H, T \rangle,$$

and for $n > 1$ qubits

$$\mathcal{J}_n = \left\langle H_{(i)}, T_{(i)}, \mathrm{CNOT}_{(i,j)} : \ i, j \in \{1, 2, \ldots, n\} \right\rangle$$

(recall the subscript indicates qubits on which the gate acts). It is not hard to see that $\mathcal{J}_n$ is a group, since the Hadamard and CNOT gates are their own inverses and $T^{-1} = T^7$.

Noting that H, T, and CNOT all have matrix elements over the ring

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

we see that any $U \in \mathcal{J}_n$ also has matrix elements from this ring. Giles and Selinger [6] proved that this condition, along with a simple condition on the determinant, exactly characterizes $\mathcal{J}_n$.

Theorem (Corollary 2 from [6])**.**

$$\mathcal{J}_n = \left\{ \begin{array}{c} U \in U(N) : \textit{ Each entry of } U \textit{ is an element of } \mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right], \textit{ and } \det U = e^{i\frac{\pi}{8}Nr} \\ \textit{for some } \ r \in \{1, 2, \ldots, 8\} \end{array} \right\}$$

where $N = 2^n$.

Note that for $n \geq 4$ the condition on the determinant is simply $\det U = 1$.

2.3 Channel representation, and modding out global phases

Consider the action of an n-qubit unitary U on a Pauli $P_s \in \mathcal{P}_n$

$$UP_sU^\dagger. \tag{2.1}$$

The set of all such operators (with $P_s \in \mathcal{P}_n$) completely determines U up to a global phase. Since $\mathcal{P}_n$ is a basis for the space of all Hermitian $N \times N$ matrices, we can expand (2.1) as

$$UP_sU^\dagger = \sum_{P_r \in \mathcal{P}_n} \widehat{U}_{rs} P_r, \tag{2.2}$$

where

$$\widehat{U}_{rs} = \frac{1}{2^n} \mathrm{Tr}\left(P_r U P_s U^\dagger\right). \tag{2.3}$$

This defines an $N^2 \times N^2$ 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.

By taking the Hermitian conjugate of (2.2) we see that each matrix element of $\widehat{U}$ is real. It is also straightforward to check, using (2.2), that the channel representation respects matrix multiplication:

$$\widehat{UV} = \widehat{U}\widehat{V}.$$

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.

In this paper we use the channel representation for $U \in \mathcal{J}_n$. In this case, the entries of U are in the ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$, and from (2.3) we see that so are the entries of $\widehat{U}$. Since $\widehat{U}$ is also real, its entries are from the subring

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

The channel representation identifies unitaries which differ by a global phase. We write

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

for the groups in which global phases are modded out. Note that each $Q \in \widehat{\mathcal{C}_n}$ is a unitary matrix with one nonzero entry in each row and each column, equal to ± 1 (since 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 $W \in \widehat{\mathcal{C}_n}$.

Finally, note that since the definition of T-count is insensitive to global phases, it is well-defined in the channel representation; for $U \in \mathcal{J}_n$ we define $\mathcal{T}(\widehat{U}) = \mathcal{T}(U)$.

3 Decomposition for unitaries in $\mathcal{J}_n$

In this Section we present a decomposition for unitaries over the Clifford+T gate set and we use it to show that an algorithm for COUNT-T can be used to generate a T-optimal circuit for $U \in \mathcal{J}_n$ with overhead polynomial in $\mathcal{T}(U)$ and N.

By definition, any $U \in \mathcal{J}_n$ can be written as an alternating product of Cliffords and T gates as in equation (1.1).

In fact, the global phase $e^{i\phi}$ appearing in (1.1) can be chosen to be 1 without loss of generality. This is because the only numbers $z \in \mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ with $|z| = 1$ are $\{1, \omega, \omega^2, \ldots, \omega^7\}$ where $\omega = e^{i\frac{\pi}{4}}$ (for example, this is shown in Appendix A of reference [10]). Since $\omega \cdot \mathbb{I} = (HT^2)^3$ we see that any nontrivial phase $e^{i\phi} = \omega^r$ can be written as a single-qubit Clifford operator $(HT^2)^{3r}$.

Starting from equation (1.1) with $e^{i\phi} = 1$ and $m = \mathcal{T}(U)$, we reorganize so that each T gate is conjugated by a Clifford:

$$U = \left(\prod_{i=\mathcal{T}(U)}^{1} D_i T_{(q_i)} D_i^\dagger \right) D_0 \tag{3.1}$$

where $D_i = \prod_{j=\mathcal{T}(U)}^{i} C_j$ for $i \in \{0, 1, \ldots, \mathcal{T}(U)\}$. Note that

$$D_i T_{(q_i)} D_i^\dagger = \frac{1}{2}\left(1 + e^{i\frac{\pi}{4}}\right)\mathbb{I} + \frac{1}{2}\left(1 - e^{i\frac{\pi}{4}}\right) D_i Z_{(q_i)} D_i^\dagger.$$

In the second term we have a Pauli operator conjugated by a Clifford, which is equal to another Pauli (up to a possible phase of -1). In other words (3.1) can be written

$$U = \left(\prod_{i=\mathcal{T}(U)}^{1} R((-1)^{b_i} P_i) \right) D_0 \tag{3.2}$$

where $P_i \in \mathcal{P}_n \setminus \{\mathbb{I}\}$, $b_i \in \{0, 1\}$, and

$$R(\pm P) = e^{i\frac{\pi}{8}(\mathbb{I} \mp P)} = \frac{1}{2}\left(1 + e^{i\frac{\pi}{4}}\right)\mathbb{I} \pm \frac{1}{2}\left(1 - e^{i\frac{\pi}{4}}\right) P \qquad P \in \mathcal{P}_n \setminus \{\mathbb{I}\}.$$

The following Proposition shows that a decomposition of the form (3.2) exists with $b_i = 0$ for all $i \in \{1, 2, \ldots, \mathcal{T}(U)\}$.

Proposition 1. *For any $U \in \mathcal{J}_n$ there exists a Clifford $C_0 \in \mathcal{C}_n$ and Paulis $P_i \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ for $i \in \{1, 2, \ldots, \mathcal{T}(U)\}$ such that*

$$U = \left(\prod_{i=\mathcal{T}(U)}^{1} R(P_i) \right) C_0. \tag{3.3}$$

Proof. For any $Q \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ we have

$$R(-Q) = e^{i\frac{\pi}{8}(1+Q)} = R(Q) e^{i\frac{\pi}{4}Q}. \tag{3.4}$$

Note, using Fact 1, that $e^{i\frac{\pi}{4}Q} = C e^{i\frac{\pi}{4}Z_{(1)}} C^\dagger$ for some $C \in \mathcal{C}_n$, and

$$e^{i\frac{\pi}{4}Z_{(1)}} = e^{i\frac{\pi}{4}} \left(T_{(1)}\right)^6.$$

Now using the fact that T^6 is Clifford we get $e^{-i\frac{\pi}{4}} e^{i\frac{\pi}{4}Q} \in \mathcal{C}_n$.

By repeatedly using (3.4) we transform (3.2) into the form (3.3) (with a potentially different set of Paulis P_i). Start with the decomposition (3.2) and let $j \in \{1, 2, \ldots, \mathcal{T}(U)\}$ be the largest index such that $b_j = 1$. Then

$$\begin{aligned} U &= \left(\prod_{i=\mathcal{T}(U)}^{j+1} R\left(P_i\right) \right) R(-P_j) \left(\prod_{i=j-1}^{1} R\left((-1)^{b_i} P_i\right) \right) D_0 \\ &= \left(\prod_{i=\mathcal{T}(U)}^{j+1} R\left(P_i\right) \right) R(P_j) e^{i\frac{\pi}{4}P_j} \left(\prod_{i=j-1}^{1} R\left((-1)^{b_i} P_i\right) \right) D_0 \\ &= e^{i\frac{\pi}{4}} \left(\prod_{i=\mathcal{T}(U)}^{j} R\left(P_i\right) \right) \left[e^{-i\frac{\pi}{4}} e^{i\frac{\pi}{4}P_j} \left(\prod_{i=j-1}^{1} R\left((-1)^{b_i} P_i\right) \right) D_0 \right]. \end{aligned} \tag{3.5}$$

Note that all the Paulis outside of the square brackets have + signs in front of them. The unitary in square brackets has T-count $j-1$ (since $e^{-i\frac{\pi}{4}} e^{i\frac{\pi}{4}P_j} \in \mathcal{C}_n$) and can therefore be written as in equation (3.2) with $j-1$ terms in the product. We now recurse, applying the above steps to this unitary in square brackets. We repeat this procedure (at most $j-1$ more times) until all the minus signs are replaced with plus signs. At the end of this procedure we arrive at a unitary of the form (3.3) multiplied by some global phase which is a power of $e^{i\frac{\pi}{4}}$; we then absorb the overall phase into the Clifford C_0 (as discussed at the beginning of this Section). □

3.1 Computing T-optimal circuits using an algorithm for COUNT-T

Suppose that we have an algorithm $\mathcal{A}$ which solves the decision problem COUNT-T. For any $U \in \mathcal{J}_n$, we show that, with overhead polynomial in N and $\mathcal{T}(U)$, such an algorithm can also be used to generate a T-optimal circuit for U over the gate set $\{H, T, CNOT\}$. This is a simple application of the decomposition (3.3).

First note that $\mathcal{A}$ can be used to compute $\mathcal{T}(U)$, by starting with $m = 0$ and running $\mathcal{A}$ on each nonnegative integer in sequence until we find $m-1$ and m with $\mathcal{T}(U) \leq m$ and $\mathcal{T}(U) > m-1$. In this way we first compute $\mathcal{T}(U)$ and then we try each Pauli $Q \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ until we find one which satisfies

$$\mathcal{T}\left(R(Q)^\dagger U\right) = \mathcal{T}(U) - 1.$$

Note that (3.3) implies that at least one such Pauli Q exists. Repeating this procedure $\mathcal{T}(U) - 1$ more times we get Paulis $Q_1, \ldots, Q_{\mathcal{T}(U)}$ such that

$$\mathcal{T}\left(R(Q_1)^\dagger \ldots R(Q_{\mathcal{T}(U)-1})^\dagger R(Q_{\mathcal{T}(U)})^\dagger U\right) = 0.$$

In other words

$$U = R(Q_{\mathcal{T}(U)}) R(Q_{\mathcal{T}(U)-1}) \ldots R(Q_1) C_0 \tag{3.6}$$

where $C_0 \in \mathcal{C}_n$.

The next step is to use standard techniques to compute a circuit which implements C_0 over the gate set $\{H, T^2, \mathrm{CNOT}\}$. For example, this can be done efficiently using the procedure outlined in the proof of Theorem 8 from reference [1].

Now using Fact 1 there are Cliffords $C_1, \ldots, C_{\mathcal{T}(U)} \in \mathcal{C}_n$ which map each of the Paulis $Q_1, \ldots, Q_{\mathcal{T}(U)}$ to Z acting on the first qubit, i.e.,

$$C_i Q_i C_i^\dagger = Z_{(1)}$$

for each $i \in \{1, 2, \ldots, \mathcal{T}(U)\}$, and we can efficiently compute circuits for each of these Cliffords. Using the fact that $R(Z_{(1)}) = T_{(1)}$ and plugging into equation (3.6) gives

$$U = \left(\prod_{j=\mathcal{T}(U)}^{1} C_j^\dagger T_{(1)} C_j \right) C_0.$$

The RHS, along with the circuits for $C_0, \ldots, C_{\mathcal{T}(U)}$ discussed above, gives a T-optimal implementation of U over the gate set $\{H, T, \mathrm{CNOT}\}$.

4 T-count for single-qubit unitaries

In this Section we show how the T-count of a single-qubit unitary $U \in \mathcal{J}_1$ can be directly computed from its channel representation $\widehat{U}$.

Recall that $\widehat{U}$ has entries over the ring $\mathbb{Z}\left[\frac{1}{\sqrt{2}}\right]$. For any nonzero element $\frac{a+b\sqrt{2}}{\sqrt{2}^k}$ of this ring, the integer k can be chosen to be minimal in the following sense. If a is even and $k > 0$ then we can divide the top and bottom by $\sqrt{2}$ and reduce k by one. When a is odd or $k = 0$, k cannot be reduced any further, and we call it the *smallest denominator exponent.* Similar quantities were defined in references [10] and [6].

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

$$v = \frac{a + b\sqrt{2}}{\sqrt{2}^k}$$

with $a, b \in \mathbb{Z}$. We define $\mathrm{sde}(0) = 0$. For a $d_1 \times d_2$ matrix M with entries over this ring we define

$$\mathrm{sde}\,(M) = \max_{a \in \{1,2,\ldots,d_1\}, b \in \{1,2,\ldots,d_2\}} \mathrm{sde}(M_{ab}).$$

We use the following fact which is straightforward to prove.

Fact 2. *Let* $q, r \in \mathbb{Z}\left[\frac{1}{\sqrt{2}}\right]$ *with* $\mathrm{sde}(q) > \mathrm{sde}(r)$. *Then*

$$\mathrm{sde}\left(\frac{1}{\sqrt{2}}\,(q \pm r)\right) = \mathrm{sde}(q) + 1.$$

The T-count of a single qubit unitary is simply equal to the sde of its channel representation.

Theorem 1. *The T-count of a single-qubit unitary $U \in \mathcal{J}_1$ is*

$$\mathcal{T}(U) = \mathrm{sde}(\widehat{U}).$$

Proof. By Proposition 1, U can be decomposed as a product of $R(P_i)$ operators times C_0, with $P_i \in \{X, Y, Z\}$ for all i and $C_0 \in \mathcal{C}_1$. Note that $R(X)^2$, $R(Y)^2$, and $R(Z)^2$ are Clifford gates.

Since Proposition 1 gives a T-optimal decomposition we see that no two consecutive Paulis in this decomposition are equal, i.e., $P_i \neq P_{i+1}$ for all $i \in \{1, 2, \ldots, \mathcal{T}(U) - 1\}$. (Otherwise U could be expressed using $\mathcal{T}(U) - 2$ T gates.)

Note $\mathcal{T}(U) = \mathcal{T}(UC_0^\dagger)$ and

$$\widehat{UC_0^\dagger} = \prod_{i=\mathcal{T}(U)}^{1} \widehat{R(P_i)}. \tag{4.1}$$

The operators which appear on the RHS are

$$\widehat{R(X)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \widehat{R(Y)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \quad \widehat{R(Z)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

where the rows and columns are labeled by Paulis $\mathbb{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}$ from top to bottom and left to right.

Consider the entries of (4.1). Since it is a product of the matrices (4.2), we see that the top left entry is always 1 and all other entries in the first row and column are 0. We therefore focus on the lower-right 3×3 submatrix. Looking at this submatrix for $\widehat{R(X)}$, $\widehat{R(Y)}$, and $\widehat{R(Z)}$ we see that there are two rows with sde equal to 1 and one row where it is 0 (recall from Definition 1 that the sde of a row vector is the maximum sde of one of its entries). The rows with sde 0 in the lower-right 3×3 submatrix of $\widehat{R(X)}$, $\widehat{R(Y)}$, and $\widehat{R(Z)}$ are labeled by X, Y, and Z respectively. Taking this as the base case, we prove by induction that the lower right 3×3 submatrix of (4.1) contains two rows with sde $\mathcal{T}(U)$ and one row with sde $\mathcal{T}(U) - 1$, and this latter row is the one labeled by the Pauli $P_{\mathcal{T}(U)}$. We suppose this holds for $\mathcal{T}(U) = k$ and we show this implies it holds when $\mathcal{T}(U) = k + 1$. When $\mathcal{T}(U) = k + 1$ we consider

$$M_{k+1} = \prod_{i=k+1}^{1} \widehat{R(P_i)} = \widehat{R(P_{k+1})} M_k \tag{4.3}$$

where $M_k = \prod_{i=k}^{1} \widehat{R(P_i)}$. Consider the lower-right 3×3 submatrix of M_k. Using the inductive hypothesis, the row labeled P_k has sde $k-1$, and the other two rows have sde k. Since $P_{k+1} \neq P_k$, one of these two rows is the one labeled P_{k+1} and we write $Q = \{X, Y, Z\} \setminus \{P_k, P_{k+1}\}$ for the other one. Note (looking at equation (4.2)) that left multiplying M_k by $\widehat{R(P_{k+1})}$ leaves the row P_{k+1} alone but replaces rows P_k and Q with $\frac{1}{\sqrt{2}}$ times their sum and difference (in some order). The row P_{k+1} of M_{k+1} therefore has sde equal to k, as required. Using the fact that row P_k of M_k has sde k and row Q of M_k has sde $k-1$, and using Fact 2, we see that the two row vectors obtained by taking $\frac{1}{\sqrt{2}}$ times their sum and difference both have sde $k+1$. Hence the two rows of M_{k+1} labeled P_k and Q have sde $k+1$. This completes the proof. □

5 Algorithm for the T-count

In this Section we present an algorithm which solves COUNT-T using $\mathcal{O}(N^m \mathrm{poly}(m, N))$ time and space.

A naive approach would be to exhaustively search over products of the form (1.1) which alternate Clifford group elements and T gates, stopping if we find an expression equal to $e^{i\phi}U$.

This does not work very well because the size of the search space is large. Equation (1.1) has $m+1$ Clifford group elements and m T gates (each of which may act on any out of the n qubits), so the number of products of this form is

$$|\mathcal{C}_n|^{m+1} n^m.$$

A better approach is use Proposition 1 and search over expressions of the form

$$\left(\prod_{i=m}^{1} R(P_i) \right) C_0 \tag{5.1}$$

until we find one which is equal to U. In this case the size of the search space is $N^{2m}|\mathcal{C}_n|$ (recall $N = 2^n$). However, a small modification improves this algorithm substantially. Rather than searching for expressions of the form (5.1) until we find one that is equal to U, we can search over expressions

$$U^\dagger \prod_{i=m}^{1} R(P_i)$$

until we find one that is an element of the Clifford group $\mathcal{C}_n$. This reduces the size of the search space to N^{2m}. The goal of the rest of this Section is to describe a more complicated algorithm which obtains (roughly) a square-root improvement at the expense of increasing the memory used.

We work in the channel representation, i.e., we consider the group $\widehat{\mathcal{J}_n}$. To describe our algorithm it will be convenient to have a notion of equivalence of unitaries *up to right-multiplication by a Clifford*. To be precise, we consider the left cosets of $\widehat{\mathcal{C}_n}$ in $\widehat{\mathcal{J}_n}$. We now show how to compute a (matrix-valued) function which tells you whether or not two unitaries $W, V \in \widehat{\mathcal{J}_n}$ are from the same coset, i.e., whether or not $W = VC$ for some $C \in \widehat{\mathcal{C}_n}$.

Definition 2 (Coset label). Let $W \in \widehat{\mathcal{J}_n}$. We define $W^{(c)}$ to be the matrix obtained from W by the following procedure. First rewrite W so that each entry has a common denominator, equal to $\sqrt{2}^{\mathrm{sde}(W)}$ (recall sde is defined for matrices in Definition 1). For each column of W, look at the first nonzero entry (from top to bottom) which we write as $v = \frac{a+b\sqrt{2}}{\sqrt{2}^{\mathrm{sde}(W)}}$. 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. After performing this step on all columns, permute the columns so they are ordered lexicographically from left to right.

The following Proposition shows that the function $W \to W^{(c)}$ labels cosets faithfully, so that two unitaries have the same label if and only if they are from the same coset.

Proposition 2. *Let $W, V \in \widehat{\mathcal{J}_n}$. Then $W^{(c)} = V^{(c)}$ if and only if $W = VC$ for some $C \in \widehat{\mathcal{C}_n}$.*

Proof. First suppose $W = VC$ with $C \in \widehat{\mathcal{C}_n}$. Since $C \in \widehat{\mathcal{C}_n}$ it has exactly one nonzero entry in every row and every column, equal to either 1 or -1. In other words W is obtained from V by permuting columns and multiplying some of them by -1. Given this fact, we see from Definition 2 that $W^{(c)} = V^{(c)}$.

Now suppose $W^{(c)} = V^{(c)}$. From Definition 2 we see that $W^{(c)} = WD_1\pi_1$, where D_1 is diagonal and has diagonal entries all ± 1, and π_1 is a permutation matrix. Likewise we can write $V^{(c)} = VD_2\pi_2$. Now setting these expressions equal we get

$$W = VC, \quad \text{where} \quad C = D_2\pi_2\pi_1^{-1}D_1^{-1}. \tag{5.2}$$

Note that $C = V^\dagger W \in \widehat{\mathcal{J}_n}$. Since $C = D_2 \pi_2 \pi_1^{-1} D_1^{-1}$, we see that it has a single nonzero entry in each row and in each column, equal to either $+1$ or -1. Putting these two facts together we get $C \in \widehat{\mathcal{C}_n}$ (since any unitary in $\widehat{\mathcal{J}_n}$ with nonzero entries all ± 1 is in $\widehat{\mathcal{C}_n}$). □

Our algorithm uses a sorted coset database, defined as follows.

Definition 3 (Sorted coset database $\mathcal{D}_k^n$). For any $k \in \mathbb{N}$, a sorted coset database $\mathcal{D}_k^n$ is a list of unitaries $W \in \widehat{\mathcal{J}_n}$ with the following three properties:

(a) *Every unitary in the database has T-count k*, i.e., every $W \in \mathcal{D}_k^n$ satisfies $\mathcal{T}(W) = k$.

(b) *For any unitary with T-count k, there is a unique unitary in the database with the same coset label*, i.e., for any $V \in \widehat{\mathcal{J}_n}$ with $\mathcal{T}(V) = k$, there exists a unique $W \in \mathcal{D}_k^n$ such that $W^{(c)} = V^{(c)}$.

(c) *The database is sorted according to the coset labels*, i.e., if $W, V \in \mathcal{D}_k^n$ and $W^{(c)} < V^{(c)}$ (using lexicographic ordering on the matrices) then W appears before V.

A sorted coset database can be viewed mathematically as a list of unitaries, but in practice when we implement such a database on a computer it makes sense to store each unitary along with its coset label as a pair $\left(W, W^{(c)}\right)$. This ensures that coset labels do not need to be computed on the fly during the course of our algorithm.

5.1 Algorithm

We are given as input a unitary $U \in \mathcal{J}_n$ and a nonnegative integer m. The algorithm determines if $\mathcal{T}(U) \leq m$, and if it is, also computes $\mathcal{T}(U)$.

1. **Precompute sorted coset databases $\mathcal{D}_0^n, \mathcal{D}_1^n, \ldots, D^n_{\lceil \frac{m}{2} \rceil}$.** To generate these databases, we start with $\mathcal{D}_0^n$ which contains only the $N^2 \times N^2$ identity matrix. We then construct $\mathcal{D}_1^n$, then $\mathcal{D}_2^n$, up to $D^n_{\lceil \frac{m}{2} \rceil}$ as follows. To construct $\mathcal{D}_k^n$ we consider all unitaries of the form
$$W = R(P)M, \tag{5.3}$$
where $M \in \mathcal{D}_{k-1}^n$ and $P \in \mathcal{P}_n \setminus \{\mathbb{I}\}$, one at a time. We insert W into $\mathcal{D}_k^n$ (maintaining the ordering according to the coset labels) if and only if its coset label is new. That is to say, if and only if no unitary V in one of the databases $\mathcal{D}_0^n, \mathcal{D}_1^n, \ldots, D_{k-1}^n$ or in the partially constructed database $\mathcal{D}_k^n$ satisfies $W^{(c)} = V^{(c)}$.

2. **Check if $\mathcal{T}(U) \leq \lceil \frac{m}{2} \rceil$.** Use binary search to check if there exists $W \in \mathcal{D}_j^n$ for some $j \in \{0, 1, \ldots, \lceil \frac{m}{2} \rceil\}$, such that $\widehat{U}^{(c)} = W^{(c)}$. If so, output $\mathcal{T}(U) = j$ and stop. If not, proceed to step 3.

3. **Meet-in-the-middle search.** Start with $r = \lceil \frac{m}{2} \rceil + 1$ and do the following. For each $W \in \mathcal{D}^n_{r - \lceil \frac{m}{2} \rceil}$, use binary search to find $V \in \mathcal{D}^n_{\lceil \frac{m}{2} \rceil}$ satisfying $\left(W^\dagger \widehat{U}\right)^{(c)} = V^{(c)}$, if it exists. If we find W and V satisfying this condition, we conclude $\mathcal{T}(U) = r$ and stop. If no such pair of unitaries is found, and if $r \leq m - 1$, we increase r by one and repeat this step; if $r = m$ we conclude $\mathcal{T}(U) > m$ and stop.

Let us now consider the time and memory resources required by this algorithm. First consider step 1. To compute the sorted coset databases we loop over all unitaries of the form (5.3), with $k \in \{0, \ldots, \lceil \frac{m}{2} \rceil\}$. There are $\mathcal{O}\left(N^{2\lceil \frac{m}{2} \rceil}\right)$ such unitaries since $|\mathcal{P}_n| = N^2$. For each unitary we need to compute the coset label and search to find it in the databases generated so far; since the databases are sorted, the search can be done using $\mathcal{O}(\log(N^{2\lceil \frac{m}{2} \rceil}))$ comparisons. The objects we are comparing (the unitaries and their coset labels) are themselves of size $N^2 \times N^2$, so step 1. takes time $\mathcal{O}(N^m \text{poly}(m, N))$. To store the databases likewise requires space $\mathcal{O}(N^m \text{poly}(m, N))$. Step 2. takes time $\mathcal{O}(\text{poly}(m, N))$ since the binary search can be performed quickly. In step 3. we loop over all elements of the databases until we reach the stopping condition; the total time required in this step is $\mathcal{O}(N^m \text{poly}(m, N))$.

5.2 Correctness of the algorithm

To prove that the output of our algorithm is correct, we first show that step 1. of the algorithm correctly generates sorted coset databases $\mathcal{D}_0^n, \mathcal{D}_1^n, \ldots, D_{\lceil \frac{m}{2} \rceil}^n$. Given this fact, it is clear that if $0 \leq \mathcal{T}(U) \leq \lceil \frac{m}{2} \rceil$ then step 2. of the algorithm will correctly compute $\mathcal{T}(U)$. In the following we show that if $\lceil \frac{m}{2} \rceil < \mathcal{T}(U) \leq m$ then step 3. computes it (and otherwise outputs $\mathcal{T}(U) > m$).

First consider step 1. It is not hard to see that $\mathcal{D}_0^n$, which contains only the identity matrix, is a sorted coset database (since all unitaries with T-count 0 are Cliffords, and every Clifford has the same coset label as the identity matrix).

We now use induction; we assume that $\mathcal{D}_0^n, \mathcal{D}_1^n, \ldots, D_{k-1}^n$ are sorted coset databases and show this implies that $\mathcal{D}_k^n$, as generated by the algorithm, is a sorted coset database. We confirm properties (a), (b), and (c) from Definition 3. First suppose W is added to the list $\mathcal{D}_k^n$ by the algorithm. This implies it has the form

$$W = \prod_{i=k}^{1} \widehat{R(P_i)} \tag{5.4}$$

for some $\{P_i\} \in \mathcal{P}_n \setminus \{\mathbb{I}\}$, and that its coset label is not equal to that of some other unitary $V \in \mathcal{D}_j^n$ with $j < k$. From (5.4) we see that $\mathcal{T}(W) \leq k$. Using the inductive hypothesis we see that $\mathcal{T}(W) > k - 1$, since otherwise there would exist $V \in \mathcal{D}_{\mathcal{T}(W)}^n$ with $W^{(c)} = V^{(c)}$. Hence $\mathcal{T}(W) = k$ and $\mathcal{D}_k^n$ satisfies property (a). Now suppose that $Q \in \widehat{\mathcal{J}_n}$ has $\mathcal{T}(Q) = k$. Using the fact that $\mathcal{D}_{k-1}^n$ is a sorted coset database, there exists $M \in \mathcal{D}_{k-1}^n$ and $P \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ such that $Q^{(c)} = (R(P)M)^{(c)}$. Now looking at the method by which $\mathcal{D}_k^n$ is generated we see that there exists a unique $W \in \mathcal{D}_k^n$ such that $W^{(c)} = (R(P)M)^{(c)}$. We have thus confirmed property (b): for any $Q \in \widehat{\mathcal{J}_n}$ with $\mathcal{T}(Q) = k$ there exists a unique $W \in \mathcal{D}_k^n$ such that $W^{(c)} = Q^{(c)}$. The ordering of the database $\mathcal{D}_k^n$ (property (c)) is maintained throughout the algorithm. Hence $\mathcal{D}_k^n$ is a sorted coset database.

The following Theorem shows that the output computed in step 3. of the algorithm is correct.

Theorem 2. *Let $U \in \mathcal{J}_n$ and $m \in \mathbb{N}$ with $\lceil \frac{m}{2} \rceil < \mathcal{T}(U) \leq m$, and let $\mathcal{D}_0^n, \ldots, \mathcal{D}_{\lceil \frac{m}{2} \rceil}^n$ be sorted coset databases. Then $r = \mathcal{T}(U)$ is the smallest integer in $\{\lceil \frac{m}{2} \rceil + 1, \lceil \frac{m}{2} \rceil + 2, \ldots, m\}$ for*

which

$$\left(W^\dagger \widehat{U}\right)^{(c)} = V^{(c)} \tag{5.5}$$

with $W \in \mathcal{D}^n_{r-\lceil \frac{m}{2} \rceil}$ *and* $V \in \mathcal{D}^n_{\lceil \frac{m}{2} \rceil}$.

Proof. Using Proposition 2 we see that (5.5) implies

$$\widehat{U} = WVC$$

for some $C \in \widehat{\mathcal{C}_n}$. Hence, whenever (5.5) holds we have

$$\mathcal{T}(U) \leq \mathcal{T}(W) + \mathcal{T}(V) = r - \left\lceil \frac{m}{2} \right\rceil + \left\lceil \frac{m}{2} \right\rceil = r.$$

To complete the proof, we show that (5.5) holds with $r = \mathcal{T}(U)$. From Proposition 1 we can write

$$\widehat{U} = W_0 V_0 C_0 \tag{5.6}$$

where $C_0 \in \widehat{\mathcal{C}_n}$ and

$$W_0 = \prod_{i=\mathcal{T}(U)}^{\lceil \frac{m}{2} \rceil + 1} \widehat{R(P_i)} \qquad V_0 = \prod_{i=\lceil \frac{m}{2} \rceil}^{1} \widehat{R(P_i)}$$

for some Paulis $P_i \in \mathcal{P}_n \setminus \{\mathbb{I}\}$. Note that $\mathcal{T}(W_0) = \mathcal{T}(U) - \lceil \frac{m}{2} \rceil$ and $\mathcal{T}(V_0) = \lceil \frac{m}{2} \rceil$.

Using the fact that $\mathcal{T}(W_0) = \mathcal{T}(U) - \lceil \frac{m}{2} \rceil$ and property (b) from Definition 3, there exists $W \in \mathcal{D}^n_{\mathcal{T}(U)-\lceil \frac{m}{2} \rceil}$ satisfying $W^{(c)} = W_0^{(c)}$, which implies

$$WC_1 = W_0$$

for some $C_1 \in \widehat{\mathcal{C}_n}$ by Proposition 2. Hence

$$\widehat{U} = WC_1 V_0 C_0.$$

Now $\mathcal{T}(C_1 V_0 C_0) = \mathcal{T}(V_0) = \lceil \frac{m}{2} \rceil$ so (using the same reasoning as above) there exists $V \in \mathcal{D}^n_{\lceil \frac{m}{2} \rceil}$ satisfying

$$C_1 V_0 C_0 = VC_2$$

for some $C_2 \in \widehat{\mathcal{C}_n}$. Hence

$$\widehat{U} = WVC$$

where $C = C_2 C_0$, or equivalently $W^\dagger \widehat{U} = VC$. Applying Proposition 2 gives $\left(W^\dagger \widehat{U}\right)^{(c)} = V^{(c)}$. □

5.3 The T-count of Toffoli and Fredkin is 7

We implemented our algorithm in C++. For two qubits we were are able to generate coset databases $\mathcal{D}^2_0, \ldots, \mathcal{D}^2_6$ (which used 3.96 GB[a] of space in total), and for three qubits we generated $\mathcal{D}^3_0, \ldots, \mathcal{D}^3_3$ (size in memory 4.60 GB); this allows us to run the two-qubit algorithm with

[a] 1 GB=10^9 bytes

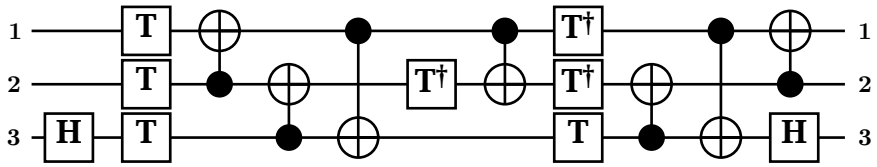

Fig. 5.1. Toffoli circuit with 7 T gates [3]. Our computer search shows that this circuit is T-optimal: Toffoli cannot be implemented using 3 qubits with less than 7 T gates.

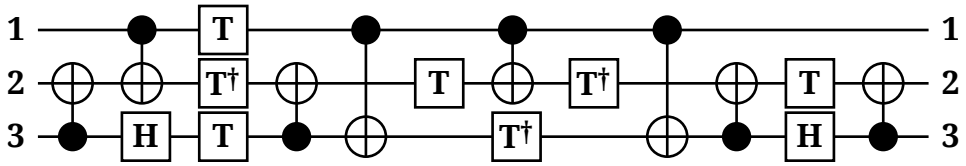

Fig. 5.2. Fredkin circuit with 7 T gates [3]. Our computer search shows that this circuit is T-optimal.

$m = 12$ or the three-qubit algorithm with $m = 6$. We ran the three-qubit algorithm on Toffoli and Fredkin gates with $m = 6$, and we found that the T-count of both of these unitaries is ≥ 7. It was already known that both of these gates can be implemented using circuits with 7 T gates [3]. Together with our results, this shows that the T-count of Toffoli and Fredkin are both 7. In Figures (5.1) and (5.2) we reproduce the circuits for these gates from reference [3], which are now known to be T-optimal. We emphasize that our definition of T-count does not permit ancilla qubits; it may be possible to do better using ancillae and measurement along with classically controlled operations (e.g., for the Toffoli gate [8]).

6 Conclusions and open problems

Our algorithm for COUNT-T can be viewed as a method of performing exhaustive search. In contrast, in the single-qubit case we can characterize the T-count as a simple property of the given unitary: the sde of its channel representation. This characterization does not appear to generalize to $n > 1$ qubits; for example, the sde of the channel representation of the Toffoli gate is 2 but its T-count is 7.

We conclude by stating the most obvious question, which remains open: does there exist a polynomial time (as a function of N and m) algorithm for COUNT-T? Alternatively, is this problem computationally difficult? We do not know the answer to this question even for the special case of two-qubit unitaries.

Acknowledgments

We thank Matthew Amy for discussions and we thank the anonymous referees for their helpful suggestions. David Gosset is supported by NSERC. Michele Mosca is supported by Canada's NSERC, MPrime, CIFAR, and CFI. IQC and Perimeter Institute are supported in part by the Government of Canada and the Province of Ontario.

References

[1] Scott Aaronson and Daniel Gottesman, *Improved simulation of stabilizer circuits*, Physical Review A **70** (2004), no. 5, 052328, available at 0406196.

[2] M. Amy, D. Maslov, and M. Mosca, *Polynomial-time T-depth Optimization of Clifford+T circuits via Matroid Partitioning*, e-print arXiv: 1303.2042 (March 2013).

[3] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler, *A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **32** (2013), 818–830, available at 1206.0758.

[4] Alex Bocharov and Krysta M. Svore, *Resource-Optimal Single-Qubit Quantum Circuits*, Physical Review Letters **109** (November 2012), no. 19, 190501.

[5] A. R. Calderbank, E. M Rains, P. W. Shor, and N. J. A. Sloane, *Quantum Error Correction via Codes over GF(4)*, eprint arXiv:quant-ph (August 1996), available at arXiv:quant-ph/9608006.

[6] Brett Giles and Peter Selinger, *Exact synthesis of multiqubit Clifford+ T circuits*, Physical Review A **87** (2013), no. 3, 032332.

[7] D. Gottesman, *The Heisenberg Representation of Quantum Computers*, eprint arXiv:quant-ph (July 1998), available at arXiv:quant-ph/9807006.

[8] C. Jones, *Low-overhead constructions for the fault-tolerant Toffoli gate*, Physical Review A **87** (February 2013), no. 2, 022328, available at 1212.5069.

[9] Vadym Kliuchnikov, *Synthesis of unitaries with Clifford+T circuits*, eprint arXiv: 1306.3200 (2013).

[10] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca, *Fast and Efficient Exact Synthesis of Single-qubit Unitaries Generated by Clifford and T Gates*, Quantum Info. Comput. **13** (July 2013), no. 7-8, 607–630.

[11] Ken Matsumoto and Kazuyuki Amano, *Representation of Quantum Circuits with Clifford and $\pi/8$ Gates*, eprint arXiv: 0806.3834 (2008June).

[12] V. Veitch, S. A. Hamed Mousavian, D. Gottesman, and J. Emerson, *The resource theory of stabilizer quantum computation*, New Journal of Physics **16** (January 2014), no. 1, 013009.

Appendix Proof of Fact 1

Proof. We first show that it is sufficient to prove the result with $P = Z_{(1)}$. To see this, consider $P_A, P_B \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ and suppose we can efficiently compute circuits for Cliffords $C_A, C_B \in \mathcal{C}_n$ which satisfy $C_A Z_{(1)} C_A^\dagger = P_A$ and $C_B Z_{(1)} C_B^\dagger = P_B$. Then $C_B C_A^\dagger P_A C_A C_B^\dagger = P_B$, and we can efficiently compute the circuit for $C_B C_A^\dagger$.

By conjugating with $\mathcal{O}(n)$ SWAP gates and CNOT gates (both are Clifford) we can map $P = Z_{(1)}$ into any operator of the form

$$\bigotimes_{i=1}^{n} Z_{(i)}^{y_i} \tag{6.1}$$

with $\vec{y} \neq \vec{0}$ an n-bit string. This can be seen using the fact that

$$\mathrm{CNOT}\,(\mathbb{I} \otimes Z)\,\mathrm{CNOT} = Z \otimes Z$$

and

$$\mathrm{SWAP}\,(\mathbb{I} \otimes Z)\,\mathrm{SWAP} = Z \otimes \mathbb{I}.$$

Finally, note that the single-qubit Pauli Z matrix can be mapped to either X or Y by single-qubit Cliffords $H \in \mathcal{C}_1$ and $T^2 H \in \mathcal{C}_1$

$$HZH = X \qquad \left(T^2 H\right) Z \left(T^2 H\right)^\dagger = Y.$$

Using these facts it is not hard to see that $P = Z_{(1)}$ can be transformed into $P' \in \mathcal{P}_n \setminus \{\mathbb{I}\}$ by first mapping to an operator of the form A.1 by conjugating with $\mathcal{O}(n)$ CNOTs and SWAPs, and then conjugating by a tensor product of (at most n) single-qubit Cliffords. □