Almost-optimal computational-basis-state transpositions

Steven Herbert,[*] Julien Sorci,[†] and Yao Tang[‡]
Quantinuum, Terrington House, 13-15 Hills Road, Cambridge CB2 1NL, United Kingdom

(Received 16 January 2024; revised 14 May 2024; accepted 16 May 2024; published 12 July 2024)

We give an explicit construction to perform any n-qubit computational-basis-state transposition using $\Theta(n)$ gates. This nearly coincides with the lower bound $\Omega[n/\log(nd)]$ on worst-case and average-case gate complexities to perform transpositions using a d-element gate set, which we also prove.

DOI: 10.1103/PhysRevA.110.012437

I. INTRODUCTION

Quantum circuits that permute computational basis states are widely found in quantum computing: the X, CNOT, and Toffoli gates do exactly this, and blocks of $\{X$, CNOT, Toffoli$\}$ are found, for example, every time an oracle is invoked to compute a classical function. Indeed, owing to the quantum *computational* universality of the gate set $\{H, \text{Toffoli}\}$ [1], every quantum circuit can be replaced by a functionally equivalent version represented as alternating blocks of permutations and Hadamard gates. Furthermore, it has been observed that many of the most powerful quantum circuits amount to no more than a computational-basis-state permutation conjugated by a transform, such as the Fourier or Schur transform [2]. Some notable instances featuring permutation circuits are Shor's factoring algorithm, which employs a permutation circuit for modular exponentiation, and the discrete-time quantum walk operator, which acts as a diffusion operator followed by a conditional shift, the latter of which is a permutation circuit [3,4].

Owing to the general importance of permutations in quantum circuits, we explore bounds on performing arbitrary computational-basis-state *transpositions*. Specifically, we consider an n-qubit circuit with computational basis states $\{\,|x\rangle : x \in \{0,1\}^n\}$, and we are interested in the gate complexity of the operation

$$|x\rangle \mapsto \begin{cases} |x\rangle\,, & \text{if } x \notin \{a,b\}, \\ |b\rangle\,, & \text{if } x = a, \\ |a\rangle\,, & \text{if } x = b, \end{cases} \tag{1}$$

for fixed but arbitrary $a, b \in \{0,1\}^n$.

As a concrete example of an important quantum circuit primitive where computational basis transpositions manifest, consider a quantum oracle which evaluates a classical function $f : \{0,1\}^n \to \{0,1\}^m$. This acts on the computational basis states as the mapping $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ for $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, where $\oplus$ denotes bitwise addition. Such a mapping can be compiled as a product of computational-basis-state transpositions which transpose $|x, y\rangle$ with $|x, y \oplus f(x)\rangle$, one for each $y \in \{0,1\}^m$ and $x \in \{0,1\}^n$ such that $f(x)$ is nonzero. Compiling this circuit as a product of transpositions is particularly advantageous for functions with low support.

Previous work on the compilation of permutation circuits largely focused on the complexity of compiling an arbitrary computational-basis-state permutation. The worst-case gate complexity was shown to be $\Omega[n2^n/\log(n)]$ (all logarithms used in the paper have insignificant bases) in Ref. [5], and constructions which nearly meet this worst-case lower bound were proposed in Refs. [6,7]. On the other hand, there appears to be little in the literature on the compilation of a computational-basis-state transposition. Noting that the set of transpositions generates the full group of permutations, transpositions constitute an important building block for quantum circuits in general.

The organization of this paper is as follows. In Sec. II we prove a lower bound on the worst-case gate complexity to compile a unitary from a given family of unitary matrices and show that the same asymptotic lower bound holds for the average gate complexity, independent of the number of ancilla qubits present. We specialize these results to the case of computational-basis-state transpositions. In Sec. III we give a construction for a circuit that performs any transposition with $\Theta(n)$ gates and either two or $n-1$ clean ancillas, which nearly achieves the lower bound of $\Omega[n/\log(nd)]$ for a d-element gate set proved in the preceding section. In Sec. IV we present numerical results demonstrating the performance of our proposed method of performing a computational-basis-state transposition in terms of CNOT- and T-gate counts. Last, in Sec. V we conclude the paper with some final remarks.

II. A LOWER BOUND ON THE GATE COMPLEXITY OF COMPUTATIONAL-BASIS-STATE TRANSPOSITIONS

In this section we prove a lower bound on the worst-case and average-case gate complexities of a computational-basis-state transposition for any finite gate set. We begin by proving a worst-case lower bound for an arbitrary set of operators and then specialize to transpositions. For the remainder of this

[*] Also at Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom; steven.herbert@quantinuum.com
[†] julien.sorci@quantinuum.com
[‡] yao.tang@quantinuum.com

section we will let $\mathcal{G}$ denote a finite gate set consisting of d gates with each gate acting on at most c qubits for some constant c.

Theorem 1. Let $\mathcal{G}$ be a finite gate set consisting of d gates. Then for any set of unitary matrices $\mathcal{U}$, there is an element of $\mathcal{U}$ with gate complexity

$$\Omega[\log(|\mathcal{U}|)/\log(nd)]. \tag{2}$$

Moreover, if $|\mathcal{U}| \in O(n^n)$, then this holds even if we permit an arbitrary number of ancilla qubits.

Proof. We first show that the claimed gate complexity holds if no ancillas are present. Consider an n-qubit circuit that is compiled by k gates of $\mathcal{G}$. Since each gate in $\mathcal{G}$ acts on at most c qubits, then there are at most $(n!/(n-c)!)d$ ways of applying a gate from $\mathcal{G}$ to the circuit, and therefore, there are at most $[(n!/(n-c)!)d]^k$ possible operations that can be achieved by a circuit with k gates. If every element of $\mathcal{U}$ can be compiled by such a circuit, then k must be large enough that

$$|\mathcal{U}| \leqslant [n!/(n-c)!d]^k,$$

or, equivalently,

$$k \geqslant \log(|\mathcal{U}|)/\log\,[(n!/(n-c)!)d]. \tag{3}$$

Therefore, there is some element in $\mathcal{U}$ that requires at least $\log\,(|\mathcal{U}|)/\log((n!/(n-c)!)d)$ gates of $\mathcal{G}$ to be compiled. For any positive integers n and c with $c \leqslant n$ we have the upper bound $(n!/(n-c)!) \leqslant n^c$, from which it directly follows that $\log((n!/(n-c)!)) \in \Theta[\log(n)]$. Thus, the resulting element of $\mathcal{U}$ has gate complexity $\Omega[\log(|\mathcal{U}|)/\log(nd)]$, as claimed.

Next, we consider the case where m additional ancillas are available. In particular, we ask whether the lower bound on the gate complexity can be reduced from that in (2). First, by the premise, we are concerned with only the case where the lower bound has been reduced from $\log\,(|\mathcal{U}|)/\log((n!/(n-c)!)d)$, and so as each gate operates on at most c qubits, this means that at most some

$$n' \leqslant c\frac{\log(|\mathcal{U}|)}{\log((n!/(n-c)!)d)}$$

qubits can be involved in the circuit. The assumption that $|\mathcal{U}| \in O(n^n)$ implies that $\log(|\mathcal{U}|) \in O[n\log(n)]$, and thus, $n' \in O(n)$. Therefore, even if an arbitrary number of ancillas are available, we can effectively upper bound the total number of qubits by n' (as the ancillas are identical). It follows that we can substitute n' into the denominator of the expression in (2); however, as $n' \in O(n)$, the asymptotic expression does not change. ■

We note that a similar version of Theorem 1 appeared in [5] as Lemma 8. However, our more general statement will be important for the results which follow it. We now show that the same gate complexity in Theorem 1 holds on average.

Theorem 2. Let $\mathcal{G}$ be a finite gate set consisting of d gates. Then for any set of unitary matrices $\mathcal{U}$, the average gate complexity of the elements of $\mathcal{U}$ is

$$\Omega[\log(|\mathcal{U}|)/\log(nd)].$$

Moreover, if $|\mathcal{U}| \in O(n^n)$, then this holds even if we permit an arbitrary number of ancilla qubits.

Proof. If we now adapt Theorem 1 to consider a $\tilde{k}$ that is large enough that *half* of the elements of $\mathcal{U}$ can be compiled, then we obtain

$$\tilde{k} \geqslant \log(|\mathcal{U}|/2)/\log((n!/(n-c)!)d).$$

To lower bound the average gate complexity of compiling the elements of $\mathcal{U}$ we now lower bound the following:

(1) The at most half of the elements of $\mathcal{U}$ that have been compiled within $\log(|\mathcal{U}|/2)/\log((n!/(n-c)!)d)$ operations have consumed at least zero operations in their compilation.

(2) The at least half of the elements of $\mathcal{U}$ that have not been compiled within $\log\,(|\mathcal{U}|/2)/\log((n!/(n-c)!)d)$ operations have each consumed at least $\log\,(|\mathcal{U}|/2)/\log((n!/(n-c)!)d)$ to compile.

From this we can easily obtain a lower bound on the average gate complexity:

$$k_{\text{ave}} \geqslant 0.5\times 0 + 0.5\times \log(|\mathcal{U}|/2)/\log((n!/(n-c)!)d).$$

The claim that the average complexity holds even with an arbitrary number of ancilla qubits follows by the same reasoning presented in the proof of Theorem 1. ■

We now specialize Theorems 1 and 2 to deduce the worst-case and average gate complexities of a computational-basis-state transposition.

Corollary 1. Let $\mathcal{G}$ be a finite gate set consisting of d gates. Then, for any n-bit computational basis state $|a\rangle$ there exists another n-bit computational basis state $|b\rangle$ such that the gate complexity required to compile a transposition of $|a\rangle$ and $|b\rangle$ using the gate set $\mathcal{G}$ is $\Omega[n/\log(nd)]$. In addition, the average complexity of such a transposition is $\Omega[n/\log(nd)]$. Both of these lower bounds hold even if we permit an arbitrary number of ancilla qubits.

Proof. This follows directly from Theorems 1 and 2 by taking $\mathcal{U}$ to be the set of transpositions with $|a\rangle$. This set has $2^n - 1$ elements since there are $2^n - 1$ distinct transpositions of $|a\rangle$ with another computational basis state. ■

III. ACHIEVING NEARLY OPTIMAL GATE COMPLEXITY FOR A COMPUTATIONAL-BASIS-STATE TRANSPOSITION

In this section we present a quantum circuit construction to compile an arbitrary transposition. Our construction makes use of the C^nX gate, so we first provide several statements on its decomposition into elementary gates. The main ideas behind these C^nX decompositions can be traced back to [8]. In the following, we will refer to an ancilla qubit as a *borrowed ancilla* if it can be in any initial state and its output state is unchanged. Similarly, we will refer to an ancilla qubit as a *clean ancilla* if its initial and final states are both $|0\rangle$.

Lemma 1. For all $n \geqslant 3$, a C^nX gate can be compiled using $n-2$ borrowed ancilla qubits and at most $4n-8$ Toffoli gates.

The compilation and its proof are deferred to the Appendix, but we give the general construction now. We write $\text{Tof}(i, j, k)$ to denote a Toffoli controlled on qubits i and j

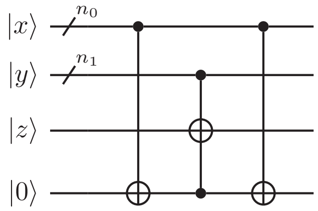
and targeted on qubit k and assume that a C^nX is controlled on qubits $x_1, \dots, x_n$, targeting qubit x_{n+1}, and $a_1, \dots, a_{n-2}$ are borrowed ancillas. The sequence of Toffoli gates which implements the desired C^nX operation is

$$\begin{aligned}&\mathrm{Tof}(a_{n-2}, x_n, x_{n+1}) \times [\mathrm{Tof}(a_{n-3}, x_{n-1}, a_{n-2})\mathrm{Tof}(a_{n-4}, x_{n-2}, a_{n-3}) \cdots \mathrm{Tof}(a_1, x_3, a_2)]\\ &\times[\mathrm{Tof}(x_1, x_2, a_1)\mathrm{Tof}(a_1, x_3, a_2) \cdots \mathrm{Tof}(a_{n-4}, x_{n-2}, a_{n-3})] \times \mathrm{Tof}(a_{n-3}, x_{n-1}, a_{n-2}),\end{aligned} \tag{4}$$

which is all repeated once more. The reader is directed to the Appendix for an explicitly worked out example of the above decomposition. The compilation of Lemma 1 uses a large number of ancilla qubits. However, this construction can be used for an alternative compilation with the same asymptotic gate complexity which, however, uses only a single clean ancilla.

Lemma 2. For all $n \geqslant 3$, a C^nX gate can be compiled using one clean ancilla qubit and at most (a) 3 Toffoli gates when $n = 3$ and (b) $6n - 18$ Toffoli gates for all $n \geqslant 4$.

Proof. Let $n_0 = \lceil n/2 \rceil$ and $n_1 = \lfloor n/2 \rfloor$ (thus, $n_0 + n_1 = n$). We show that for all $n \geqslant 3$ the circuit



acts as a C^nX gate controlled on the first two registers and targeting the third, with the fourth register being a clean ancilla (where a control on a bundle of qubits represents a control on each qubit in the bundle). We prove this by showing that it implements the mapping which sends the basis state $|x, y, z, 0\rangle$ to

$$|x, y, z \oplus (x_1 \wedge \cdots \wedge x_{n_0}) \wedge (y_1 \wedge \cdots \wedge y_{n_1}), 0\rangle$$

for all $x = (x_1, \dots, x_{n_0}) \in \{0, 1\}^{n_0}$, $y = (y_1, \dots, y_{n_1}) \in \{0, 1\}^{n_1}$, and $z \in \{0, 1\}$, where $\oplus$ denotes bitwise addition and $\wedge$ denotes the logical "and."

Considering the action of each operator in the circuit on an arbitrary initial state $|x, y, z, 0\rangle$, the basis state is mapped as

$$\begin{aligned}&\mapsto \big|x, y, z, x_1 \wedge x_2 \wedge \cdots \wedge x_{n_0}\big\rangle\\ &\mapsto \big|x, y, z \oplus \big(x_1 \wedge x_2 \wedge \cdots \wedge x_{n_0}\big) \wedge \big(y_1 \wedge \cdots \wedge y_{n_1}\big),\\ &\quad x_1 \wedge x_2 \wedge \cdots \wedge x_{n_0}\big\rangle\\ &\mapsto \big|x, y, z \oplus \big(x_1 \wedge x_2 \wedge \cdots \wedge x_{n_0}\big) \wedge \big(y_1 \wedge \cdots \wedge y_{n_1}\big), 0\big\rangle,\end{aligned}$$

which shows the circuit implements the claimed operation.

Last, we count the number of Toffoli gates used. The circuit is composed of two $C^{n_0}X$ gates and one $C^{n_1+1}X$ gate. We compute the resulting Toffoli gate count by cases.

When $n = 3$, then $n_0 = 2$, and $n_1 = 1$, so in this case we have used three Toffoli gates and only the one clean ancilla shown. This completes the proof of Lemma 2(a).

For Lemma 2(b) we first consider the case of $n = 4$, where $n_0 = 2$ and $n_1 = 2$. In this case we may apply Lemma 1 to compile the $C^{n_1+1}X$ gate using one borrowed ancilla and four Toffoli gates. There are two qubits that are neither the target nor the control of the $C^{n_1+1}X$ gate, and either may be used as a borrowed ancilla for its compilation. Therefore, in this case we have used a total of six Toffoli gates and only one clean ancilla, as claimed in Lemma 2(b), i.e., noting $6\times4-18 = 6$ Toffoli gates for $n = 4$.

Finally, when $n \geqslant 5$, both n_0 and $n_1 + 1$ are at least 3, so we may compile the $C^{n_0}X$ and $C^{n_1+1}X$ gates using Lemma 1. By Lemma 1, a $C^{n_0}X$ gate can be compiled using $n_0 - 2$ borrowed ancilla qubits. Since $n_0 - 2 \leqslant n_1 + 1$, the $n_1 + 1$ qubits that are neither the target nor control of the $C^{n_0}X$ gates may be used as borrowed ancillas for their compilation. Similarly, a $C^{n_1+1}X$ gate can be compiled using $n_1 - 1$ borrowed ancilla qubits, and since $n_1 - 1 \leqslant n_0$, the n_0 qubits that are neither the target nor control of the $C^{n_1+1}X$ gate may be used as borrowed ancillas to compile it. Therefore, no additional ancilla qubits are required. Counting Toffoli gates, we obtain a total of

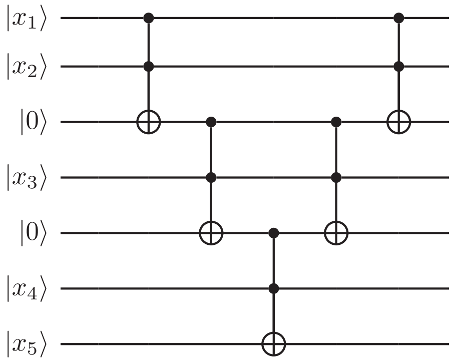
$$\begin{aligned}2(4n_0 - 8) + 4(n_1 + 1) - 8 &= 4n + 4n_0 - 20\\ &\leqslant 4n + 2n + 2 - 20\\ &= 6n - 18\end{aligned}$$

gates, where the first equality follows since $n_0 + n_1 = n$ and the inequality follows since $4n_0 \leqslant 2n + 2$ (which holds because n is an integer). Thus, we have used the claimed number of Toffoli gates in Lemma 2(b). This completes the proof in all cases. ■

The final C^nX compilation that we present uses a larger number of ancilla qubits but reduces the number of Toffoli gates by a multiplicative constant.

Lemma 3. For all $n \geqslant 3$, a C^nX gate can be compiled using $n - 2$ clean ancilla qubits and $2n - 3$ Toffoli gates.

Proof. We provide a proof for the case $n = 4$ for concreteness. The general case follows by an analogous argument on a circuit with the same pyramidlike shape that we present now. Consider the circuit



We will show that this circuit acts as a C^4X gate which is controlled on the first, second, fourth, and sixth qubits and targets the final qubit and that the remaining qubits are clean

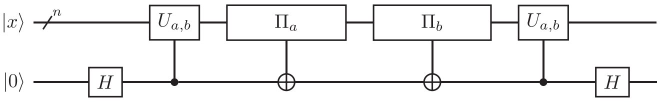


FIG. 1. The circuit of Theorem 3, which acts as a computational-basis-state transposition.

ancillas. Applying the gates one at a time to an arbitrary initial computational basis state $|x_1, x_2, 0, x_3, 0, x_4, x_5\rangle$, it is mapped as

$$\begin{aligned}
&\mapsto |x_1, x_2, x_1 \wedge x_2, x_3, 0, x_4, x_5\rangle \\
&\mapsto |x_1, x_2, x_1 \wedge x_2, x_3, x_1 \wedge x_2 \wedge x_3, x_4, x_5\rangle \\
&\mapsto |x_1, x_2, x_1 \wedge x_2, x_3, x_1 \wedge x_2 \wedge x_3, x_4, \\
&\quad x_5 \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)\rangle \\
&\mapsto |x_1, x_2, x_1 \wedge x_2, x_3, 0, x_4, x_5 \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)\rangle \\
&\mapsto |x_1, x_2, 0, x_3, 0, x_4, x_5 \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)\rangle\,,
\end{aligned}$$

which shows that the circuit implements the claimed operation. The number of Toffoli gates and clean ancillas follows by directly counting. ■

We can now give the main result of this section. To that end, let $|a\rangle$ and $|b\rangle$ be an arbitrary pair of n-qubit computational basis states that are to be transposed; further, let Π_a and Π_b be projectors onto these basis states. Our construction will make use of the $(n+1)$-qubit operators:

$$\Pi_a \otimes X + (I - \Pi_a) \otimes I, \tag{5}$$

$$\Pi_b \otimes X + (I - \Pi_b) \otimes I. \tag{6}$$

In circuit diagrams, these are represented as a block denoted Π_a or Π_b controlling a "$\oplus$" on the target qubit. As the projectors in question are onto computational basis states, these gates may be realized by a C^nX gate where the control is "sandwiched" between a pair of X gates when conditioned on zero for the relevant qubit. In this way, the gate "picks out" a single computational basis state which controls a bit flip on the target qubit. We also define the n-qubit operator:

$$U_{a,b} := U_1 \otimes U_2 \otimes \cdots \otimes U_n,$$

where $U_i = X$ if a and b differ in the ith bit and $U_i = I$ otherwise. Note that $U_{a,b}$ acts on $|a\rangle$ and $|b\rangle$ as $U_{a,b}|a\rangle = |b\rangle$ and $U_{a,b}|b\rangle = |a\rangle$.

Theorem 3. The circuit in Fig. 1 acts as a transposition of the computational basis states $|a\rangle$ and $|b\rangle$ for all n. For $n = 1$, $n = 2$, and $n = 3$ the circuit requires at most (i) 2 Hadamard gates, 4 X gates, 4 CNOT gates, and 1 clean ancilla; (ii) 2 Hadamard gates, 8 X gates, 4 CNOT gates, 2 Toffoli gates, and 1 clean ancilla; (iii) 2 Hadamard gates, 12 X gates, 6 CNOT gates, 6 Toffoli gates, and 2 clean ancillas, respectively, and for all $n \geqslant 4$ the circuit requires at most either (a) 2 Hadamard gates, $4n$ X gates, $2n$ CNOT gates, $12n - 36$ Toffoli gates, and 2 clean ancillas or (b) 2 Hadamard gates, $4n$ X gates, $2n$ CNOT gates, $4n - 6$ Toffoli gates, and $n - 1$ clean ancillas. Thus, in all cases the overall gate complexity is $\Theta(n)$, nearly achieving the lower bound of Corollary 1.

Proof. First, we show that the circuit acts as the mapping defined in (1) for an arbitrary input $|x\rangle|0\rangle$:

(1) For $|x\rangle|0\rangle$ with $x \notin \{a, b\}$, the first Hadamard gate maps $|x\rangle|0\rangle$ to $|x\rangle|+\rangle$. As $U_{a,b}$ is a permutation which sends $|a\rangle$ to $|b\rangle$ and $|b\rangle$ to $|a\rangle$, it follows that $U_{a,b}$ must send $|x\rangle$ to some computational basis state $|y\rangle$, where y is not equal to a or b. Therefore, the controlled $U_{a,b}$ sends $|x\rangle|+\rangle$ to $\frac{1}{\sqrt{2}}|x\rangle|0\rangle + \frac{1}{\sqrt{2}}|y\rangle|1\rangle$. Following this, none of the conditions of the next two controlled operations are met, so the state remains unchanged. The state is then mapped by the last controlled $U_{a,b}$ to $\frac{1}{\sqrt{2}}|x\rangle|0\rangle + \frac{1}{\sqrt{2}}|x\rangle|1\rangle$, and the final Hadamard gate maps this to $|x\rangle|0\rangle$. Therefore, the overall operation in this case is to map $|x\rangle|0\rangle$ to $|x\rangle|0\rangle$.

(2) Turning to the case where $x = a$, the first Hadamard gate sends $|a\rangle|0\rangle \mapsto \frac{1}{\sqrt{2}}|a\rangle|0\rangle + \frac{1}{\sqrt{2}}|a\rangle|1\rangle$; the controlled-$U_{a,b}$ operation then sends this to $\frac{1}{\sqrt{2}}|a\rangle|0\rangle + \frac{1}{\sqrt{2}}|b\rangle|1\rangle$. The third and fourth circuit blocks together send this to $\frac{1}{\sqrt{2}}|a\rangle|1\rangle + \frac{1}{\sqrt{2}}|b\rangle|0\rangle$; the second controlled-$U_{a,b}$ operation then sends this to $\frac{1}{\sqrt{2}}|b\rangle|1\rangle + \frac{1}{\sqrt{2}}|b\rangle|0\rangle$, and the remaining Hadamard gate maps this to $|b\rangle|0\rangle$. Thus, the overall operation is to send $|a\rangle|0\rangle \mapsto |b\rangle|0\rangle$. The case where $x = b$ follows by a completely analogous argument.

By the above case analysis, we have shown that the circuit performs the claimed transposition.

Now that we have shown that the circuit has the required operation, it remains to count gates and qubits. There are two uses of the controlled-$U_{a,b}$ operator. Each requires at most n CNOT gates, giving at most $2n$ CNOT gates. Continuing, there are two uses of the operators defined in (5) and (6). Each of these operators consists of a C^nX gate and at most $2n$ additional X gates. Thus, for any $n \geqslant 1$ the total number of operations required is 2 Hadamard gates, $4n$ X gates, $2n$ CNOT gates, and 2 C^nX gates. For the cases $n = 1$ and $n = 2$ this results in the bounds claimed in Theorem 3, parts (i) and (ii). For $n \geqslant 3$, we may apply either Lemma 2 or 3 to compile each C^nX (and it turns out that for the case of $n = 3$ the resources are identical). Using the compilation provided by Lemma 2, each C^nX can be compiled with a clean ancilla qubit and 3 Toffoli gates when $n = 3$ or $6n - 18$ Toffoli gates when $n \geqslant 4$. Since the required ancilla qubit is a clean ancilla, we may reuse the same one for each of these operations. Therefore, in this case we have used a total of 2 clean ancillas (one explicitly shown and one required by Lemma 2), 6 Toffoli gates for $n = 3$ and $12n - 36$ Toffoli gates for $n \geqslant 4$, $2n$ CNOT gates, 2 Hadamard gates, and at most $4n$ X gates. This completes the proof of Theorem 3, part (a). For Theorem 3, part (b), suppose we instead apply the C^nX compilation of Lemma 3. In this case each C^nX can be compiled using $n - 2$ clean ancillas and $2n - 3$ Toffoli gates. Since the ancillas are clean, they may be reused for each of these operations. Therefore, the total gate complexity in this case is $n - 1$ clean ancillas (one explicitly shown and $n - 2$ required by Lemma 3), $4n - 6$ Toffoli gates, $2n$ CNOT gates, 2 Hadamard gates, and at most $4n$ X gates, completing the proof of Theorem 3, part (b). ■

Remark 1. The number of X gates can be reduced to $3n$ by noticing that for any qubit controlled on the 0 state for both the Π_a- and Π_b-controlled X gates, a pair of X gates will cancel. That is, owing to the fact that these gates occur consecutively,

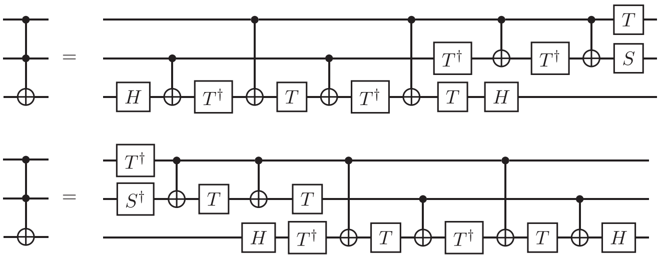


FIG. 2. The standard decomposition of the Toffoli gate (top) and its inverse (bottom) into single-qubit and CNOT gates (see Fig. 4.9 in [10]).

there will be three rather than four (partially filled) banks of X gates when the trivial simplification $XX = I$ is applied to the compilation. We also note the similarity of the circuit structure in Fig. 1 to that of the Hadamard test, which may be useful for further generalizations in future work.

IV. NUMERICAL RESULTS

We now present some numerical results to demonstrate the performance of our proposed method of transposing computational basis states. The numerical results fall into two categories. First, we compile a range of transpositions using the approach described in Theorem 3, parts (a) and (b), and compare the CNOT- and Toffoli-gate counts of the resulting circuits to the theoretical bounds described therein. Second, we compare the CNOT- and T-gate counts from our method to several state-of-the-art approaches for compiling permutational circuits, namely, the TWEEDLEDUM-based construction presented in [9] and the ToffoliBox of PYTKET. In each case, the Toffoli gates are compiled according to Fig. 2, such that the final circuits consist only of CNOT and single-qubit gates. The choice to compare CNOT- and T-gate counts is motivated by the fact that typically, CNOT gates are the most expensive gates when running circuits with physical qubits and T gates are the most expensive gates to perform fault tolerantly. All of the resulting circuits were compiled using PYTKET version 1.18.0 [11], and only mild optimization passes were used to simplify gate redundancies.

A. Comparison with theoretical bounds

For our first set of results, we compare the average CNOT- and Toffoli-gate counts across a range of random transpositions to the theoretical bounds described in Theorem 3. For each $2 \leqslant n \leqslant 20$, we generate 200 random transpositions of two n-qubit computational basis states and use the constructions proposed in Theorem 3, parts (a) and (b), to compile their corresponding circuits, resulting in circuits over the gate set $\{H, X, \text{CNOT}, \text{Toffoli}\}$. The RemoveRedundancies pass in PYTKET was applied to each of these circuits, and then the average CNOT- and Toffoli-gate counts were tabulated, as presented in Table I. The results show that the average CNOT count is typically only approximately half of the bound that we prove in this paper, whereas the Toffoli counts saturate or nearly saturate the bounds in all cases.

TABLE I. The average (Avg) CNOT- and Toffoli-gate counts across 200 random transpositions. The number n is the number of qubits required for the computational basis states that are transposed, and "Bd" is an abbreviation for the word "bound." The labels "(a)" and "(b)" refer to the two settings considered in Theorem 3; note that the bound on the number of CNOT gates is the same for both (a) and (b).

CNOT				Toffoli				
n	Avg (a)	Avg (b)	Bd	n	Avg (a)	Bd (a)	Avg (b)	Bd (b)
2	2.60	2.64	4	2	2	2	2	2
3	3.52	3.35	6	3	6	6	6	6
4	4.10	4.18	8	4	12	12	10	10
5	5.13	5.15	10	5	24	24	14	14
6	6.10	6.10	12	6	32	36	18	18
7	7.12	6.95	14	7	48	48	22	22
8	8.33	8.05	16	8	56	60	26	26
9	8.87	9.00	18	9	72	72	30	30
10	10.09	10.36	20	10	80	84	34	34
11	11.14	10.75	22	11	96	96	38	38
12	11.95	12.30	24	12	104	108	42	42
13	12.66	13.09	26	13	120	120	46	46
14	14.05	14.05	28	14	128	132	50	50
15	14.78	15.02	30	15	144	144	54	54
16	15.86	15.82	32	16	152	156	58	58
17	17.03	16.55	34	17	168	168	62	62
18	18.43	17.64	36	18	176	180	66	66
19	18.39	19.24	38	19	192	192	70	70
20	20.12	20.74	40	20	200	204	74	74

B. Comparison with other approaches

For our second set of results we compare the average CNOT- and T-gate counts of the compilation method proposed in Theorem 3 to the TWEEDLEDUM-based compilation method of Ref. [9] and the ToffoliBox of PYTKET. In each case either 100 transpositions of the same Hamming weight were randomly generated, or in the case with fewer than 100 distinct transpositions of the same Hamming weight the entire set was considered. In particular, in the following cases there were fewer than 100 possible transpositions [format is (number of qubits, Hamming distance) total transpositions]: (4, 1) 32; (4, 2) 48; (4, 3) 32; (4, 4) 8; (5, 1) 80; (5, 4) 80; (5, 5) 16; (6, 6) 32; and (7, 7) 64. For each Hamming weight the resulting circuits were compiled, and the average CNOT count and T-gate counts were computed; they are presented in Figs. 3 and 4.

TWEEDLEDUM. The first compilation method that we compare our method to is that of Ref. [9]. For each of the random transpositions the circuits were compiled and simplified using the CliffordSimp, SynthesiseTket, and RemoveRedundancies passes in PYTKET, which resulted in circuits using CNOT, TK1, and global phase gates. PYTKET version 1.18.0 does not contain functionality for T-gate synthesis, so only the CNOT-gate counts are recorded and presented in Fig. 3.

ToffoliBox. The second compilation method is the ToffoliBox of PYTKET. The compilation can use one of two strategies, referred to as "matching" and "cycle." For the matching strategy, the resulting circuits were compiled and simplified using the CliffordSimp, SynthesiseTket, and RemoveRedundancies

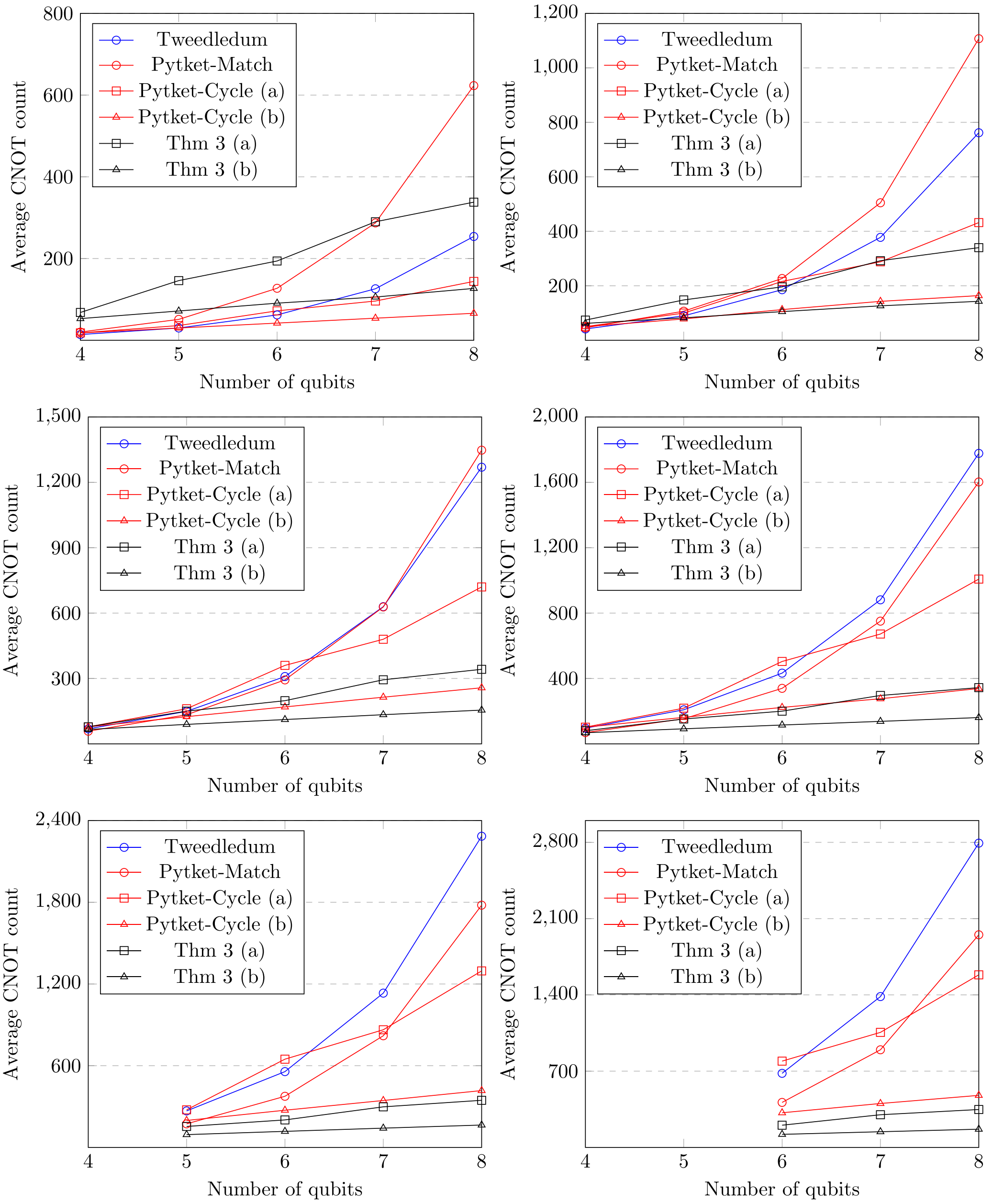


FIG. 3. The average CNOT counts across 100 randomly selected transpositions (or over all transpositions, when the total is fewer than 100) between computational basis states with a fixed Hamming distance. The plots, reading from left to right and then top to bottom, correspond to Hamming distances 1, . . . , 6, respectively. The number of qubits is the number of qubits required for the computational basis states that are transposed.

passes in PYTKET, resulting in circuits that use CNOT, TK1, and global phase gates. As noted in the TWEEDLEDUM case, PYTKET version 1.18.0 does not contain functionality for T-gate synthesis, so only the CNOT-gate counts are recorded and presented in Fig. 3 as "Pytket-Match." It is worth mentioning that compilation from a gate set including the continuously parametrized gate TK1 to a finite gate set, such as that containing the Clifford gates and T gates (or Clifford, Toffoli, and T gates), can be done only approximately, and if very high accuracy is required, the T-gate count becomes large. For this reason, fault-tolerant compilation is likely to favor techniques that require gates from a suitable finite set in the first place.

For the cycle strategy, the ToffoliBox returns a circuit consisting of X and C^nX gates. To more readily compare these circuits to those of our proposed construction, the C^nX gates were decomposed into X, CNOT, and Toffoli gates using the same C^nX decompositions used in Theorem 3, parts (a) and (b). The Toffoli gates were then decomposed into single-qubit gates and CNOT gates using the standard decomposition in Fig. 2. The RemoveRedundancies pass of PYTKET was then applied, and the CNOT- and T-gate counts were recorded. The counts are denoted by "Pytket-Cycle (a) and (b)" in Figs. 3 and 4, corresponding to the C^nX decomposition used.

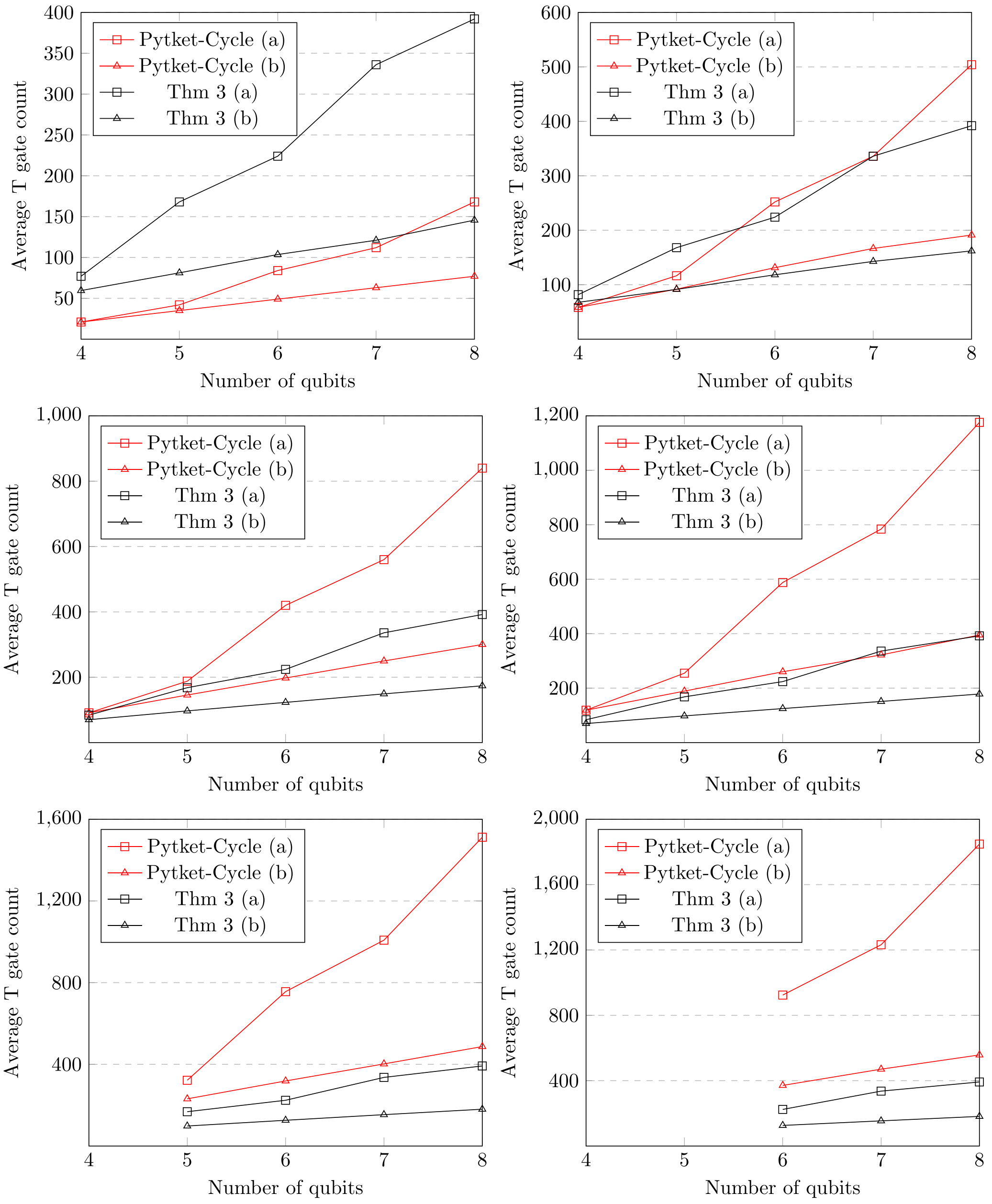


FIG. 4. The average T-gate counts across 100 randomly selected transpositions (or over all transpositions, when the total is fewer than 100) between computational basis states with a fixed Hamming distance. The plots, reading from left to right and then top to bottom, correspond to Hamming distances $1, \dots, 6$, respectively. The number of qubits is the number of qubits required for the computational basis states that are transposed.

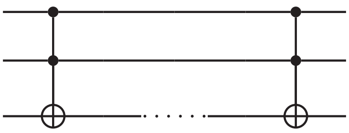
Theorem 3. For the compilation method of Theorem 3 the circuits were compiled into X, CNOT, and Toffoli gates using the constructions described in the theorem. Following this, the Toffoli gates in the circuits were then decomposed into single-qubit gates and CNOT gates using the decomposition in Fig. 2. The RemoveRedundancies pass of PYTKET was then applied, and the CNOT- and T-gate counts were recorded. The corresponding counts are denoted by "Thm 3 (a)" and "Thm 3 (b)" in Figs. 3 and 4.

We can see that the methods we propose in this paper are relatively most advantageous for large numbers of qubits and for large Hamming distances. This is to be expected, as our methods are nearly optimal in the number of qubits and have approximately the same performance for any transposition, whereas other methods, such as those that use a Gray code, suffer when the transposition is such that the Hamming distance between transposed computational basis states (written as binary strings) is large.

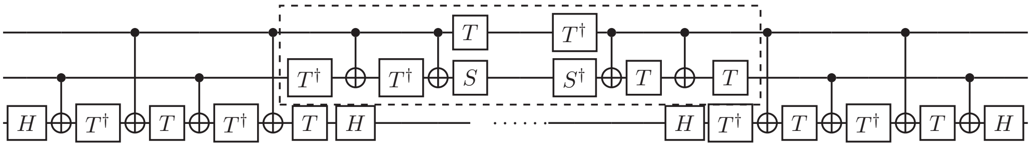
V. DISCUSSION

In this paper we have shown that on average, n-qubit computational-basis-state transpositions have a gate complexity $\Omega[n/\log(nd)]$ for any d-element gate set, even if ancillas are available. Since a general permutation can be expressed as a product of at most 2^{n-1} transpositions, this lower bound is consistent with the $\Omega[n2^n/\log(n)]$ worst-case lower bound of Ref. [5] for an arbitrary permutation. We subsequently gave an explicit construction to perform any computational-basis-state transposition with $\Theta(n)$ gates and two ancillas. Conventional wisdom is to use the Gray-code construction popularized in Nielsen and Chuang [10] to perform any two-level unitary (in the case of a transposition the unitary is the Pauli-X matrix), which requires $\Theta(n^2)$ gates in the worst case. This construction therefore represents a potentially practically useful result for any compiler that constructs arbitrary permutations from transpositions. This claim of potential for practical utility is backed up by the numerical results presented, which showed that for transpositions with large numbers of qubits and/or a large Hamming distance, our methods outperform the standard alternatives.

It is also worth noting that the transposition construction presented in Theorem 3 is amenable to several further circuit optimizations during compilation. In particular, if we consider the compilation of a Toffoli gate into single-qubit gates and CNOT gates, then the standard circuit is given by Fig. 2. However, as the Toffoli gate is equal to its inverse, we also have that the circuit reversed, with every gate replaced by its inverse, implements the Toffoli, as shown in Fig. 2. Therefore, it follows that each time a pair of Toffoli gates appears as



(where the dotted line implies that other operations occur there), then we can use the second Toffoli decomposition for the second Toffoli gate, such that the decomposed circuit is



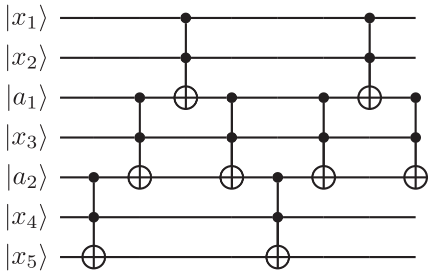
where we can readily see that the gates inside of the region enclosed by the dashed line cancel to the identity. So it follows that we have implemented the two Toffoli gates using a total of 8 CNOT gates and 12 single-qubit gates—fewer than the 12 CNOT gates and 20 single-qubit gates that are needed in general to compile two Toffoli gates. We can further see, for example, in (A1), that such structures are commonplace in our construction and hence have the potential for significant CNOT and T-gate count reductions during compilation. These savings can be readily observed in the numerical data presented in Figs. 3 and 4, which demonstrate lower CNOT- and T-gate counts for transpositions between computational basis states of large Hamming distance when compared to TWEEDLEDUM and PYTKET.

ACKNOWLEDGMENTS

The authors would like to thank S. Dilkes, A. Krajenbrink, and T. Laakkonen for carefully reviewing and providing useful feedback on an earlier draft of this article. We give special thanks to T. Laakkonen for suggesting improvements to the circuit construction given in Theorem 3 and to S. Dilkes for providing various suggestions for Sec. IV.

APPENDIX

We provide an explicit example of Lemma 1 when $n = 4$, as described by (4). The circuit in this case is

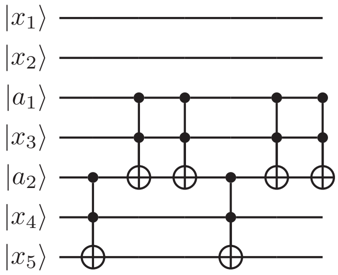


(A1)

which we show implements the desired mapping. Note that the general case follows by considering a circuit with the same pyramid structure as above, where qubit numbers $2i + 3$, with $i = 0, 1, \ldots, n - 3$, are ancilla qubits. We consider the action of the circuit in (A1) on an arbitrary input $|x_1, x_2, a_1, x_3, a_2, x_4, x_5\rangle$ and show that it is mapped to the basis state:

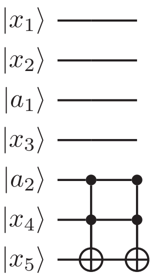
$$|x_1, x_2, a_1, x_3, a_2, x_4, x_5 \oplus (x_1 \wedge \cdots \wedge x_4)\rangle . \tag{A2}$$

Case 1. Suppose that at least one of x_1 or x_2 is equal to zero. Then the two Toffoli gates that are controlled on the first two registers will act as the identity. Consequentially, the circuit in (A1) will have the same action as the circuit:



(A3)

Since the Toffoli gate is equal to its inverse, the consecutive Toffoli gates multiply to the identity, so (A3) is equivalent to



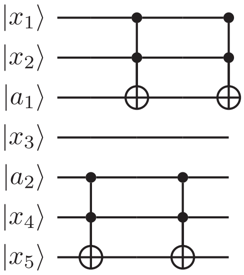
(A4)

which is then equivalent to the empty circuit for the same reason. Therefore, the overall mapping is to send

$$|x_1, x_2, a_1, x_3, a_2, x_4, x_5\rangle \mapsto |x_1, x_2, a_1, x_3, a_2, x_4, x_5\rangle \quad \text{(A5)}$$

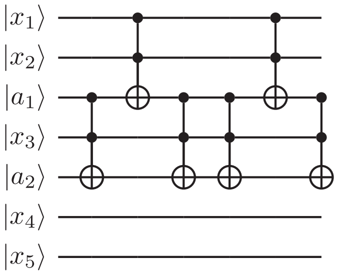
in this case.

Case 2. Suppose that x_3 is equal to zero. Then the four Toffoli gates which target the fifth qubit of (A1) act as the identity, and the circuit acts as



Again, as the Toffoli gate is equal to its inverse, the above circuit acts as the identity, and the overall mapping in this case is also (A5).

Case 3. Suppose that x_4 is equal to zero. Then the two Toffoli gates that are targeted on the final qubit have no effect, and the circuit acts as



By the same reasoning as in the previous cases, the circuit reduces to the identity.

Case 4. Last, we check the operation when x_1, x_2, x_3, and x_4 are all equal to 1. Considering the circuit (A1) one gate at a time, the basis state $|1, 1, a_1, 1, a_2, 1, x_5\rangle$ is mapped to

$$\begin{aligned}
&\mapsto |1, 1, a_1, 1, a_2, 1, x_5 \oplus a_2\rangle\\
&\mapsto |1, 1, a_1, 1, a_2 \oplus a_1, 1, x_5 \oplus a_2\rangle\\
&\mapsto |1, 1, a_1 \oplus 1, 1, a_2 \oplus a_1, 1, x_5 \oplus a_2\rangle\\
&\mapsto |1, 1, a_1 \oplus 1, 1, a_2 \oplus 1, 1, x_5 \oplus a_2\rangle\\
&\mapsto |1, 1, a_1 \oplus 1, 1, a_2 \oplus 1, 1, x_5 \oplus 1\rangle\\
&\mapsto |1, 1, a_1 \oplus 1, 1, a_2 \oplus a_1, 1, x_5 \oplus 1\rangle\\
&\mapsto |1, 1, a_1, 1, a_2 \oplus a_1, 1, x_5 \oplus 1\rangle\\
&\mapsto |1, 1, a_1, 1, a_2, 1, x_5 \oplus 1\rangle\,,
\end{aligned}$$

which indeed is the operation in (A2). Therefore, we checked all cases and have shown that the circuit in (A1) implements the mapping (A2), as claimed.

[1] D. Aharonov, A simple proof that Toffoli and Hadamard are quantum universal, arXiv:quant-ph/0301040.

[2] V. Havlíček, S. Strelchuk, and K. Temme, A classical algorithm for quantum SU(2) Schur sampling, Phys. Rev. A **99**, 062336 (2019).

[3] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE, New York, 1994), pp. 124–134.

[4] J. Kempe, Quantum random walks: An introductory overview, Contemp. Phys. **44**, 307 (2003).

[5] V. Shende, A. Prasad, I. Markov, and J. Hayes, Synthesis of reversible logic circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **22**, 710 (2003).

[6] M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian, Reversible circuit synthesis using a cycle-based approach, ACM J. Emerg. Technol. Comput. Syst. **6**, 1 (2010).

[7] L. Li and X. Wu, Asymptotically optimal synthesis of reversible circuits, arXiv:2302.06074.

[8] C. Gidney, Constructing large controlled nots (2015).

[9] M. Soeken, F. Mozafari, B. Schmitt, and G. D. Micheli, Compiling permutations for superconducting QPUs, in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, Piscataway, NJ, 2019), pp. 1349–1354.

[10] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th anniversary ed. (Cambridge University Press, Cambridge, 2010), Sec. 4.5.2.

[11] See https://cqcl.github.io/tket/pytket/api/ for PYTKET documentation.