Efficient Methods for Multi-Qubit Circuit Optimization

Luke Heyfron, and Earl T. Campbell

October 2017

Abstract

1 Introduction

Compiling is the conversion of an algorithm into a series of hardware level commands or elementary gates. Better optimised compilers can implement the same algorithm using fewer hardware level instructions, running faster and saving other resource. Quantum compiling or gate-synthesis is the analogous task for a quantum computer and especially important given the current expense of quantum hardware. At the very dawn of quantum computing as a field, Solovay and Kitaev proposed a general purpose compiler that was compatible with any universal set of elementary gates [1, 2, 3]. Unfortunately, it was far from optimal, generating circuits using many more gates than are necessary.

After a long lull, recent years have brought a series of significant improvements in compiling methods. This new generation of compilers exploit the specific structure of the Clifford+T gate set, reducing quantum circuit depths by several order of magnitude and also often improving the classical compile time [4, 5, 6, 7]. Focus on this gate set is warranted since it naturally appears as the set of logical gates in almost every fault-tolerant computing architectures [8]. Furthermore, proposed fault-tolerant devices often use techniques, such as magic state distillation [9], where the cost per T gate is several hundreds of Clifford gates [10, 11, 12]. This suggests T count as the key metric of compiler performance.

Significant progress has been made on synthesis of single-qubit unitaries from Clifford+T gates. For purely unitary synthesis, the problem is essentially solved since we have a compiler than is optimal and efficient [4, 7], with a command line implementation freely available [13]. Further yet improvements are possible beyond unitary circuits, by making use of ancilla qubits and measurements [14, 15, 16] or adding an element of randomness to compiling [17, 18]. Whereas, the multiqubit problem is much more challenging. An algorithm is known for provably optimal (in terms of T gates) unitary synthesis of multiqubit circuits over Clifford+T but the compile runtime is exponential in the number of qubits [6]. We instead seek a compiler that runs efficiently with circuit size and outputs as close as possible to minimal T count, with some prior attempts at this goal [19].

A useful for strategy multiqubit synthesis is take an initial Clifford+T circuit and split it into partitions containing Hadamards and partitions containing CNOT, S and T gates. One can then attempt to reduce T gates just within the latter partitions. Amy and Mosca recently shows that this restricted synthesis problem is formally equivalent to error decoding on a class of Reed-Muller codes [20], which is in turn equivalent to finding the symmetric tensor rank of a 3-tensor [21]. Unfortunately, even this easier sub-problem is NP-complete to solve optimally. Nevertheless, this problem does seem more amenable to efficient solvers that offer reductions in T gates. Amy and Mosca argued that an n-qubit subcircuit (containing CNOT, S and T gates) has an optimal decomposition into $n^2/2 + O(n)$ T gates. At the time, known efficient compilers could only promise an output circuit with no more than $O(n^3)$ T gates. Later, Campbell and Howard [22] sketched a compiler that is efficient and promises an output circuit with at most $n^2/2 + O(n)$ T gates. This shows efficient compilers can in this sense be "near-optimal" with respect to worst case scaling. On the mathematical level, Campbell and Howard exploited a previously known efficient and optimal solver for a related 2-tensor problem [23] but suitably modified so that it nearly-optimally solves the required 3-tensor problem.

This paper develops several different compilers that are efficient and near-optimal in the above sense. We provide the first implementations of such compilers and provide performance comparison against: a family of random circuits; and a library of circuits taken from actual quantum algorithms. For random circuits, we find the actual performance follows the worst-case scaling for different compilers. We observe $O(n^2)$ scaling for all variants

of our compiling approach compared with $O(n^3)$ scaling for compilers based on earlier work. For actual quantum algorithms, the full Clifford+T gate set is needed and so our compiler also makes use of a gadgetisation tricks to eliminate Hadamards and convert the problem to synthesis over just the CNOT, S and T gate set. Quantum algorithms are highly structured and far from random, so the number of T gates can not be meaningfully compared with the worst case n scaling. Instead, we benchmark against the best previously known results. We find our compilers gave significant reductions in T count for almost every circuit tested and never gave worse T counts.

All of the near-optimal compilers described in this paper look for inspiration in algorithms for the related 2-tensor problem, which we call the Lempel's algorithm. We give specific details for a compiler here called TOR (Time Ordered Reduction) that comes in two different flavours (with and without feedback). The TOR compilers can be considered concrete versions of the approach outlined by Campbell and Howard [22]. Also, proposed in this paper is the TODD (Third Order Duplicate and Destroy) compiler, which is also inspired by Lempel but in a more direct and elegant way than TOR. In benchmarking, we find that TODD often achieved even lower T count than TOR.

2 Workflow Overview

The high level work-flow of the protocol begins by converting an n qubit input Clifford + T circuit, C_{in}, (see figure 1.1) to a CNOT + T circuit plus additional operations that are all Clifford (figure 1.2). The purpose of this is to isolate the non-Clifford behaviour within a subcircuit that has properties required for our optimization framework (see section 3). The main obstacle to this conversion process is the presence of Hadamard gates in the original circuit. External Hadamards, found at the beginning and end of the circuit, can be left in place. In order to remove internal Hadamards we use an approach similar to that of [24, 25] where a unitary Hadamard gate is replaced with a Hadamard gadget that involves only CNOT + T gates as well as preparation of ancillas in the $|+\rangle$ state and measurement in the Pauli-X basis (see figure 2). In doing so we are trading internal Hadamards for external Hadamards and space-like overhead.

Once the Hadamards are removed, we decompose the CNOT + T circuit into two partitions such that one part consists of CNOT gates only and the other is diagonal in the computational basis (figure 1.3). Such circuits implement unitaries in the group $\mathcal{D}_3$, the diagonal elements of the third level of the Clifford hierarchy. We then extract the *signature tensor* of the diagonal CNOT + T circuit, which is an object that uniquely characterizes the unitary U_f up to a Clifford factor.

The core layer of the algorithm, *T-optimiser*, takes as input the signature tensor of the diagonal CNOT + T circuit and outputs a *gate synthesis matrix*, A, which represents a particular circuit decomposition for the input unitary. The number of columns of A is equal to the T count of its corresponding circuit, so *T-optimiser* ideally outputs a gate synthesis matrix with minimal column number. It is believed that finding optimal solutions for this problem is NP-hard [26], so we must use efficient heuristics that yield near-minimal column number.

The output A matrix is mapped to a circuit of the same form as U_f, with which the original U_f circuit is replaced. The result is an output circuit that implements the full Clifford + T input circuit on the first n qubits and has near optimal T count. The following sections detail these steps in turn. New algorithms for the *T-Optimize* subroutine are the main technical contribution of this paper and so are the main focus.

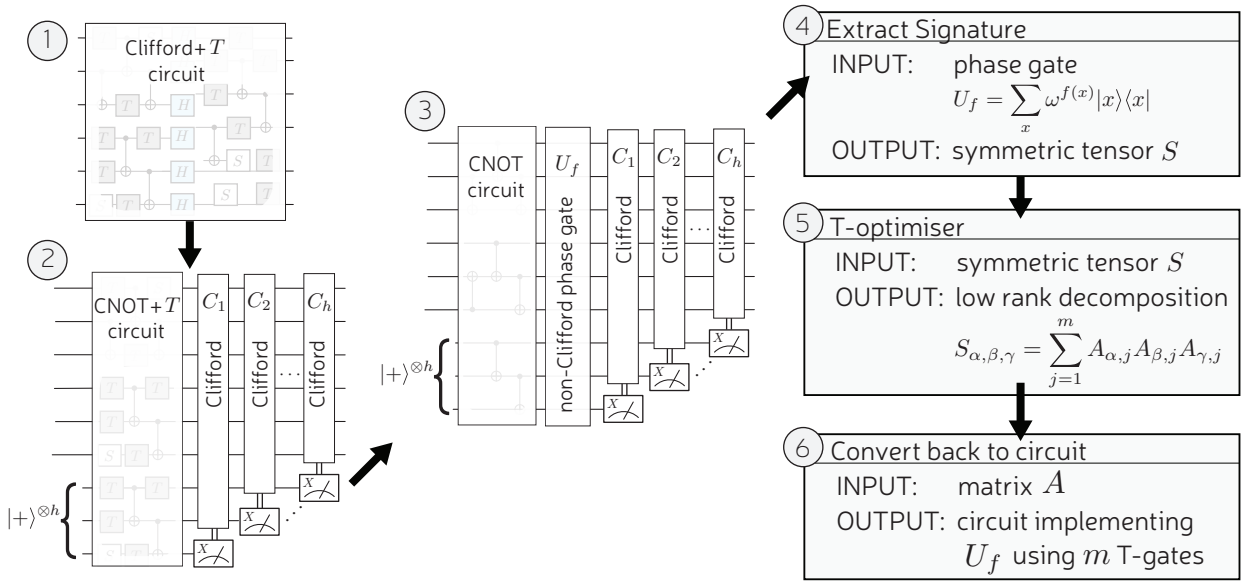

Figure 1: The high level workflow of the T gate optimization protocol is shown. A Clifford + T circuit is converted to the CNOT+T gate set by introducing ancillas and performing classically controlled Clifford gates. A non-Clifford phase gate is extracted, which maps to a signature tensor upon which the core optimization algorithm is performed. The optimized symmetric tensor decomposition is then converted back into a circuit of the form in panel 2) yielding an implementation of the original Clifford + T circuit with reduced T count.

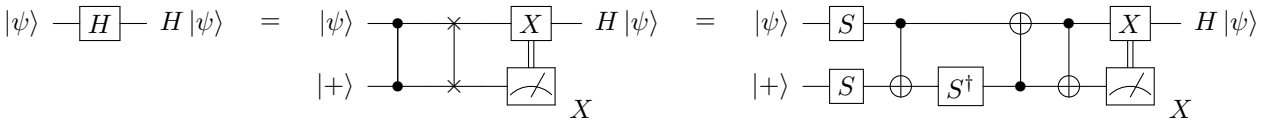

Figure 2: Internal Hadamards are removed by replacing each one according to this circuit identity. The Hadamards on the left-hand circuit are now external and act on an ancilla, therefore they no longer obstruct the *T-optimiser* algorithms. The classically controlled Pauli-X correction of each Hadamard gadget must be conjugated through to the end of the circuit in order to maximize the extent of the CNOT+T sub-circuit of figure 1.2. Because circuits generated by CNOT+T form a group within the third level of the Clifford heirarchy, the conjugation of a Pauli operator through this circuit leads to classically controlled Clifford operators $C_1, C_2, \ldots, C_h$ in figure 1.2 and 1.3.

3 Diagonal CNOT+T Framework

In section 2 we showed how we map a more general Clifford + T circuit to a diagonal CNOT+T circuit. The problem we are trying to solve is to minimize the number of T gates used in a CNOT + T circuit decomposition, as stated in the following.

Problem 3.1. ***(T-OPT)*** *Given a $U_f \in \mathcal{D}_3$, find a circuit decomposition $\mathcal{G} \in \langle CNOT, T, S \rangle$ that implements U_f with minimal uses of the T gate.*

This section describes how we map the T-OPT problem from the quantum circuit picture to an abstract mathematical picture involving binary tensors. We proceed by borrowing the observation from [27, 28] that the action of any n-qubit diagonal CNOT + T unitary U_f on a computational basis state $|\mathbf{x}\rangle$ is given by

$$U_f |\mathbf{x}\rangle = \omega^{f(\mathbf{x})} |\mathbf{x}\rangle , \tag{1}$$

where $\omega = e^{i\frac{\pi}{4}}$, and U_f is completely characterized by a polynomial $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_8$, known as the phase function of U_f. A phase function can be decomposed into a sum of linear, quadratic and cubic monomials on the Boolean variables x_i. Each monomial of order r has a coefficient in $\mathbb{Z}_8$ and is weighted by a factor 2^{r-1}, as in the following:

$$f(\mathbf{x}) = \sum_{\alpha=1}^{n} l_\alpha x_\alpha + 2 \sum_{\alpha<\beta}^{n} q_{\alpha,\beta} x_\alpha x_\beta + 4 \sum_{\alpha<\beta<\gamma}^{n} c_{\alpha,\beta,\gamma} x_\alpha x_\beta x_\gamma \pmod 8, \tag{2}$$

where $l_\alpha, q_{\alpha,\beta}, c_{\alpha,\beta,\gamma} \in \mathbb{Z}_8$. In reference [29] such decompositions of f are referred to as *weighted polynomials* and it was proven that any two unitaries with weighted polynomials whose coefficients all have the same parity are Clifford equivalent. Note the weighted polynomial can be lifted directly from the circuit definition of U_f if we work in the $\{T, CS, CCZ\}$ basis, as each kind of gate corresponds to the linear, quadratic and cubic terms, respectively.

We define the *signature tensor*, $S \in \mathbb{Z}_2^{(n,n,n)}$, of U_f to be a symmetric tensor of order 3 whose elements are equal to the parity of the weighted polynomial coefficients of U_f according to the following relations:

$$S_{\sigma(\alpha,\alpha,\alpha)} = S_{a,a,a} = l_\alpha \pmod 2 \tag{3a}$$
$$S_{\sigma(\alpha,\beta,\beta)} = S_{\sigma(\alpha,\alpha,\beta)} = q_{\alpha,\beta} \pmod 2 \tag{3b}$$
$$S_{\sigma(\alpha,\beta,\gamma)} = c_{\alpha,\beta,\gamma} \pmod 2 \tag{3c}$$

for all $\sigma \in \mathcal{S}_3$, the symmetric group on a set with 3 elements. It follows from this definition that any two unitaries with the same signature tensor are Clifford equivalent.

We recall the definition of gate synthesis matrices from reference [29], where a matrix, A in $\mathbb{Z}_2^{(n,m)}$, is a gate synthesis matrix for a unitary U_f if it satisfies,

$$f(\mathbf{x}) = |A^T\mathbf{x}| \pmod 8 \tag{4}$$
$$= \sum_{j=1}^{m} (A_{1,j}x_1 \oplus A_{2,j}x_2 \oplus \cdots \oplus A_{n,j}x_n) \pmod 8 \tag{5}$$

where f is the phase function of U_f and $|.|$ is the Hamming weight of a binary vector. An A matrix can be efficiently extracted from a diagonal CNOT + T circuit by tracking the action of each gate on the computational basis states through the circuit. The signature tensor of U_f can be determined from an A matrix of U_f using the following relation,

$$S^{(A)}_{\alpha,\beta,\gamma} = \sum_{j=1}^{m} A_{\alpha,j}A_{\beta,j}A_{\gamma,j} \pmod 2 \tag{6}$$

In our gate synthesis protocol, we exploit a key property of A matrices described in the following lemma.

Lemma 3.1. *Given an A matrix composed of m unique, non-zero column vectors each of length n, one can efficiently generate a circuit that implements U_f with m uses of the T gate.*

Proof. First, we note from the definition of A in equation 4 that each column of A leads to a factor of $\omega^{\lambda_j(\mathbf{x})}$ appearing in the diagonal elements of U_f as written in equation 1, where λ_j is a reversible linear Boolean function given by,

$$\lambda_j(\mathbf{x}) = A_{1,j}x_1 \oplus A_{2,j}x_2 \oplus \cdots \oplus A_{n,j}x_n. \tag{7}$$

The action of a circuit generated by CNOT gates on computational basis state $|\mathbf{x}\rangle$ is to replace the value of each qubit with a reversible linear Boolean function on $x_1, x_2, \ldots, x_n$. Therefore, each λ_j can be implemented using a subcircuit B_j composed of only CNOT gates. We define B_j such that, after applying B_j to the input computational basis state $|\mathbf{x}\rangle$, the i^{th} qubit is in state $|\lambda_j(\mathbf{x})\rangle$. A T gate subsequently applied to qubit i results in the desired phase of $\omega^{\lambda_j(\mathbf{x})}$. We now uncompute B_j by reversing the order of the CNOT gates and applying the resultant $B_j^\dagger$ to the CBS, and so column j has been implemented. We then move on the next column j' until all columns of A have been implemented in this way. This method of generating U_f from A requires steps $\mathcal{O}(\text{poly}(n, m))$ and therefore is efficient. □

We now have the necessary tools to define the gate synthesis problem in the binary tensor picture.

Problem 3.2. ***(3-STR)*** *Given a symmetric tensor of order 3, $S \in \mathbb{Z}_2^{(n,n,n)}$, find a matrix $A \in \mathbb{Z}_2^{(n,m)}$ that satisfies equation 6 with minimal m.*

4 *T-optimiser*

Until now the *T-optimiser* subroutine of our protocol has been treated as a black box whose input is a signature tensor S and the output is a gate synthesis matrix S with few columns. In this section we will describe the inner workings of the various *T-optimiser*s we have implemented in this work. In reference [26], Amy and Mosca proved that the T-OPT problem is equivalent to minimum distance decoding of the punctured Reed-Muller code of order $n-4$ and length n (often written as $RM^*(n-4, n)$), which is believed to be a hard problem. This imposes a practical upper bound on the number of qubits, n_{RM}, over which circuits can be optimally synthesized with respect to T count.

4.1 Reed-Muller decoder (RM)

Although Reed-Muller decoding is believed to be hard, a brute force solver can be implemented for a small number of qubits. We implement such a brute force decoder and found its limit to be 6 qubits. To gain some intuition for the complexity of the problem, consider the following. The number of codespace generators for $RM^*(n-4,n)$ is equal to $N_G = \sum_{r=1}^{n-4} \binom{n}{r}$. Therefore, the size of the search space is $N_{\text{search}} = 2^{N_G}$. On a processor with a clock speed of 3.20GHz, assuming we can check one codeword per clock cycle, it would take over 91 years to exhaustively search this space and therefore determine optimality for a general CNOT+T circuit for $n = 7$. Performing the same back-of-the-envelope calculation for $n = 6$, it would take $\approx 7 \times 10^{-4}$ seconds. In practice, we find the brute force decoder executes in around 10 minutes for $n = 6$, so the time for $n = 7$ would be significantly worse. Clearly, we need develop heuristics for this problem.

4.2 Recursive Expansion (RE)

The simplest means of efficiently obtaining an A matrix for a given signature tensor S is to make use of the modulo identity $2ab = a + b - a \oplus b$. This is applied recursively to each non-linear term in equation 2 of the weighted polynomial associated with S to yield a decomposition of the form in equation 4. Or explicitly, for each odd-parity coefficient in the weighted polynomial l_α, $q_{\alpha,\beta}$, $c_{\alpha,\beta,\gamma}$, make the following substitutions to the corresponding monomials:

$$x_\alpha \to x_\alpha, \tag{8}$$

$$2x_\alpha x_\beta \to x_\alpha + x_\beta - x_\alpha \oplus x_\beta, \tag{9}$$

$$4x_\alpha x_\beta x_\gamma \to x_\alpha + x_\beta + x_\gamma - x_\alpha \oplus x_\beta - x_\alpha \oplus x_\gamma - x_\beta \oplus x_\gamma + x_\alpha \oplus x_\beta \oplus x_\gamma, \tag{10}$$

from which the corresponding A matrix can be easily extracted. We call this the *recursive expansion* (RE) algorithm, which has been shown to yield worst-case T counts of $\mathcal{O}(n^3)$. It is straightforward to see this, as gate synthesis matrices resulting from the RE algorithm may include any column with a Hamming weight of $W \in [1,3]$. There are $\sum_{k=1}^{3} \binom{n}{k} = \mathcal{O}(n^3)$ such columns so there can be at most $\mathcal{O}(n^3)$ T gates in the corresponding circuit decomposition.

4.3 Order Reduction via Nested Polynomials (ORNePol)

An efficient heuristic that requires at most only $\mathcal{O}(n^2)$ T gates was proposed in reference [29], which involves breaking down an input CNOT + T circuit into a sequence of control-U operators where each successive operator acts on fewer qubits than the previous, eventually allowing the optimal Reed-Muller decoder to be used when $n \leq n_{RM}$ for a particular control-U. Each target U corresponds to a weighted polynomial with only up to quadratic terms, and the control qubit maps to a particular variable in the weighted polynomial that has been factorized out from all terms in which it appears. The reduction in order from a cubic to a quadratic weighted polynomial allows for an optimal decomposition U' for each target U to be found efficiently using Lempel's matrix factoring algorithm [30]. Additional steps are required to obtain a decomposition of the form of equation 4 from the control-U' sequence, which leads to sub-optimality in the general case but with T count scaling of $\mathcal{O}(n^2)$, a factor n improvement over RE. This algorithm we refer to as the *order reduction via nested polynomials* (ORNePol) algorithm efficiently yields optimal T counts for the special case where every gate in the CNOT + T circuit shares a control.

There are two versions of ORNePol: with and without feedback. This refers to whether or not extra terms that do not depend on the control qubit of the current iteration are immediately synthesized after calling Lempel's algorithm or if they are fed back into the input signature tensor to be synthesized in the subsequent step.

We now describe a single round of optimization according to the ORNePol algorithm. We begin by showing that given some input unitary $U_f \in \mathcal{D}_3$, we can decompose U_f according to panel 2 of figure 3. Recall the definition of the weighted polynomial from equation 2,

$$f(\mathbf{x}) = \sum_{\alpha=1}^{n} l_\alpha x_\alpha + 2\sum_{\alpha<\beta}^{n} q_{\alpha,\beta} x_\alpha x_\beta + 4\sum_{\alpha<\beta<\gamma}^{n} c_{\alpha,\beta,\gamma} x_\alpha x_\beta x_\gamma \pmod 8 \tag{11}$$

Now, we group terms in the following way,

$$f(\mathbf{x}) = 2x_h \tilde{f}_{x_h}(\mathbf{x} \setminus x_h) + l_h x_h + f'_{x_h}(\mathbf{x} \setminus x_h) \pmod 8, \tag{12}$$

where we have defined the following:

Definition 4.1. Nested Polynomial. Let $f(\mathbf{x})$ be a weighted polynomial on n qubits. We say $\tilde{f}_{x_h}(\mathbf{x} \setminus x_h)$ is the nested polynomial of f with respect to x_h defined such that

$$\tilde{f}_{x_h} = \frac{f_{x_h} - f'_{x_h} - l_h}{2}, \tag{13}$$

where f_{x_h} and f'_{x_h} are the positive and negative Shannon cofactors of f with respect to x_h.

The first term in equation 12 corresponds to the control-$U_{\tilde{f}_{x_h}}$ in panel 2 of figure 3, where in our example circuit $h = 1$. The second term corresponds to the T^{l_h} gate. The final term corresponds to the only operator to the right of the dotted line, $U_{f'_{x_h}}$, which acts on $n-1$ qubits. The key part of this construction is that by factoring out $2x_h$ from a portion of f, we obtain a nested polynomial, $\tilde{f}_{x_h}$, that is only defined up to quadratic terms, or explicitly:

$$\tilde{f}_h(\mathbf{x}) = \sum_{\alpha=1,\alpha\neq h}^{n} q_{\alpha,h} x_\alpha + 2 \sum_{\alpha<\beta,\beta\neq h}^{n} c_{\alpha,\beta,h} x_\alpha x_\beta \pmod 8. \tag{14}$$

Operators of the form control-$U_{\tilde{f}_{x_h}}$ are uniquely defined (up to Clifford equivalence) by a signature tensor of order 2, rather than order 3 as in the TODD algorithm (see section 4.4). For a given nested polynomial, $\tilde{f}_{x_h}$, we define the order 2 signature tensor (in a way analogous to equation 3) in terms of the coefficients of the original weighted polynomial, f,

$$S_{\alpha,\alpha} = q_{a,h} \pmod 2 \tag{15}$$

$$S_{\alpha,\beta} = S_{\beta,\alpha} = c_{\alpha,\beta,h} \pmod 2, \tag{16}$$

which can be determined from a gate synthesis matrix, $A \in \mathbb{Z}_2^{(n,m)}$, that implements $\tilde{f}_h$,

$$S_{\alpha,\beta} = \sum_{j=1}^{m} A_{\alpha,j} A_{\beta,j} \pmod 2. \tag{17}$$

The optimization step between panels 2 and 3 of figure 3 is to find an A matrix with minimal columns that satisfies equation 17.

Problem 4.1. ***(2-STR)*** *Given a symmetric tensor of order 2, $S \in \mathbb{Z}_2^{(n,n)}$, find a matrix $A \in \mathbb{Z}_2^{(n,m)}$ that satisfies equation 17 with minimal m.*

It turns out that this problem is exactly equivalent to the matrix factoring problem solved by Abraham Lempel in reference [30]. In an effort to make this paper more self-contained, and because the TODD algorithm described in section 4.4 is based on it, we provide a description of Lempel's algorithm in appendix A. For now, we treat Lempel's solver as a black box $OPT(\tilde{f})$ that outputs an optimized decomposition for the input nested polynomial $\tilde{f}$, as can be seen in panel 3 of figure 3.

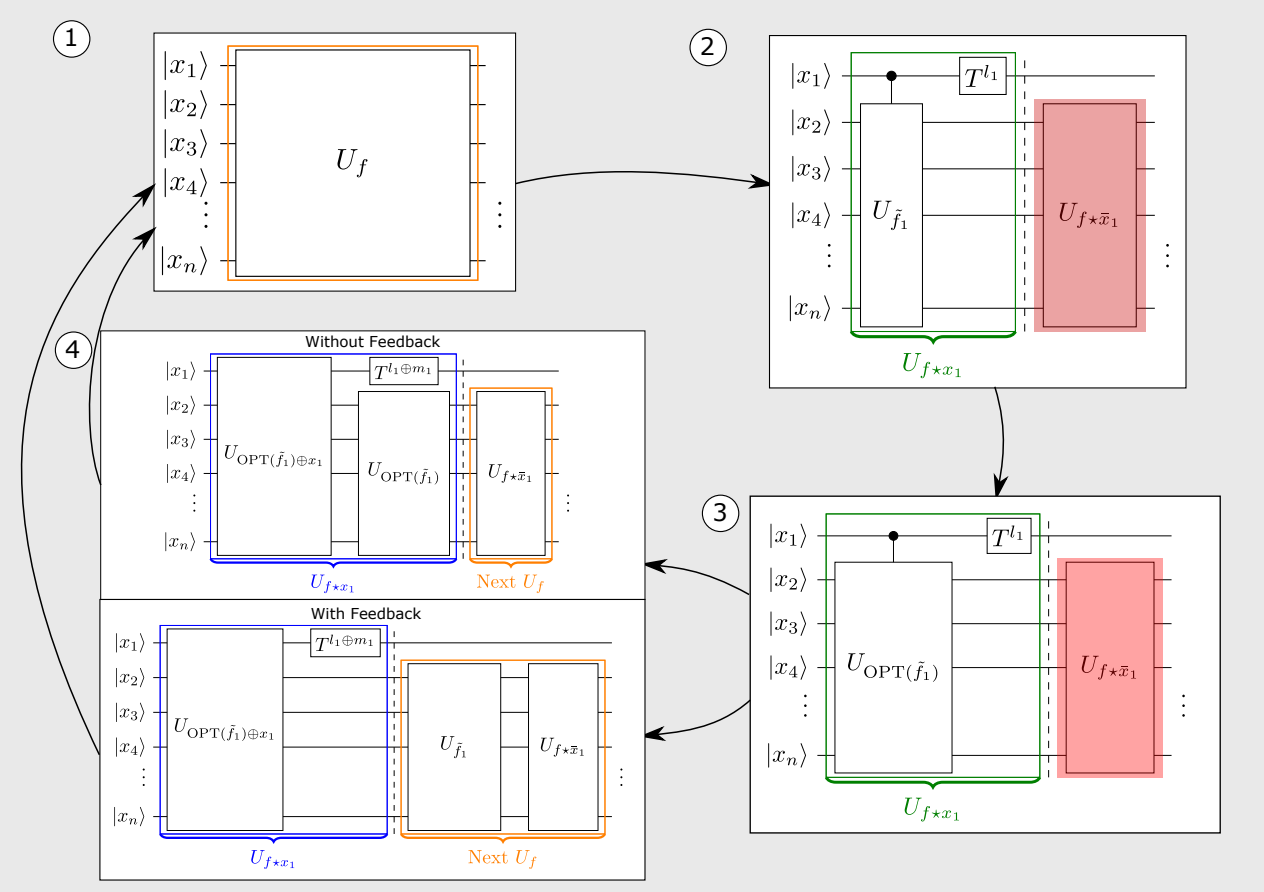

Figure 3: The structure of the TOR algorithm is shown in the quantum circuit picture. The colours of each box have the following meanings: orange border is for subcircuits yet to be pre-processed before the start of optimization and synthesis steps; green border shows subcircuits that are actively being optimized synthesized; blue border is for synthesized subcircuits i.e. they are now decomposed into a sequence of CNOT and T gates; and red fill are for subcircuits that not currently being processed.

4.4 Third Order Duplicate and Destroy (TODD)

In this section, we present an algorithm based in principle on Lempel's matrix factoring algorithm [30] that is extended to work for tensors of order 3. This algorithm requires some initial A matrix to be generated by another algorithm such as RE or TOR, then it reduces the number of columns the initial gate synthesis matrix iteratively until exit. In section 5, we present numerical evidence that it is the best efficient solver of the 3-STR problem developed so far. We call this the *Third Order Duplicate and Destroy* (TODD) algorithm because, much like the villainous Victorian barber, it shaves away at the columns of the input A matrix iteratively until the algorithm finishes execution.

We begin by introducing the key mechanism through which TODD reduces the T count of quantum circuits: by *destroying* pairs of duplicate columns of a gate synthesis matrix, a process which can be done without changing the signature tensor, as shown in the following lemma.

Lemma 4.1. *Let $A \in \mathbb{Z}^{(n,m)}$ be a gate synthesis matrix whose a^{th} and b^{th} columns are duplicates and $A' \in \mathbb{Z}^{(n,m-2)}$ be a gate synthesis matrix formed from by removing the a^{th} and b^{th} columns of A. It follows that $S^{(A)} = S^{(A')}$ for any such A and A'.*

Proof. We start by writing the signature tensor in terms of the elements of A according to equation 6,

$$S^{(A)}_{\alpha,\beta,\gamma} = \sum_{k=1}^{m} A_{\alpha,k} A_{\beta,k} A_{\gamma,k} \pmod 2, \tag{18}$$

and separating the terms associated with a, b from the rest of the summation,

$$S^{(A)}_{\alpha,\beta,\gamma} = \sum_{j\in\mathcal{J}} A_{\alpha,j} A_{\beta,j} A_{\gamma,j} + A_{\alpha,a} A_{\beta,a} A_{\gamma,a} + A_{\alpha,b} A_{\beta,b} A_{\gamma,b} \pmod 2, \tag{19}$$

where $\mathcal{J} = [1, m] \setminus \{a, b\}$. As stated in the lemma, the a^{th} and b^{th} columns of A are duplicates and so

$$A_{i,a} = A_{i,b} \;\forall\; i \in [1, n]\,. \tag{20}$$

Now substitute equation 20 into equation 19,

$$\begin{aligned} S^{(A)}_{\alpha,\beta,\gamma} &= \sum_{j\in\mathcal{J}} A_{\alpha,j} A_{\beta,j} A_{\gamma,j} + A_{\alpha,a} A_{\beta,a} A_{\gamma,a} + A_{\alpha,a} A_{\beta,a} A_{\gamma,a} \pmod 2 &(21)\\ &= \sum_{j\in\mathcal{J}} A_{\alpha,j} A_{\beta,j} A_{\gamma,j} \pmod 2 &(22)\\ &= S^{(A')}_{\alpha,\beta,\gamma}, &(23)\end{aligned}$$

where the penultimate step follows from modulo 2 addition and the final step follows from the definition of A' and equation 6. □

Lemma 4.1 gives us a simple means to remove columns from a gate synthesis matrix by destroying pairs of duplicates columns and thereby reducing the T count of a CNOT + T circuit by 2. However, it is often the case that the A matrix does not already contain any duplicate columns. Therefore, we wish perform some transformation on the A matrix, which results in a different gate synthesis matrix A' that: a) has duplicate columns; b) preserves the signature tensor of A; and c) does not increase the column number of A by more than 1. In the following lemma we introduce a class of transformations that *duplicate* a particular column of an A matrix such that properties a) and c) are met. We then use lemma 4.3 establish what conditions must be satisfied for the duplication transformation to have property b).

Lemma 4.2. *Let A be an $n \times m$ gate synthesis matrix where all columns are unique and non-zero. Let $A' = A \oplus \mathbf{z}\mathbf{y}^T$ where $\mathbf{z}$ and $\mathbf{y}$ are column vectors on $\mathbb{Z}_2$ of length n and m, respectively, defined such that $z_i = A_{i,a} \oplus A_{i,b}$ for some $a, b \in [1, m]$ and $y_a \oplus y_b = 1$. If follows that the a^{th} and b^{th} columns of A' are duplicates.*

Proof. We begin the proof by finding expressions for the matrix elements of A' in terms of A, $\mathbf{z}$ and $\mathbf{y}$,

$$A'_{i,j} = A_{i,j} \oplus z_i y_j, \tag{24}$$

and substitute the definition of $\mathbf{z}$,

$$A'_{i,j} = A_{i,j} \oplus (A_{i,a} \oplus A_{i,b}) y_j. \tag{25}$$

Now we can find the elements of the columns a and b of A',

$$A'_{i,a} = A_{i,a} \oplus (A_{i,a} \oplus A_{i,b}) y_a, \tag{26}$$
$$A'_{i,b} = A_{i,b} \oplus (A_{i,a} \oplus A_{i,b}) y_b. \tag{27}$$

We substitute in the condition $y_b = y_a \oplus 1$ into 27,

$$\begin{aligned} A'_{i,b} &= A_{i,b} \oplus (A_{i,a} \oplus A_{i,b})(y_a \oplus 1) \\ &= A_{i,a} \oplus (A_{i,a} \oplus A_{i,b}) y_a \\ &= A'_{i,a}. \end{aligned} \tag{28}$$

□

Lemma 4.3. *Let A and $A' = A \oplus \mathbf{z}\mathbf{y}^T$ be two gate synthesis matrices where $\mathbf{z}$, $\mathbf{y}$ are arbitrary column vectors of dimension n and m, respectively. $S^{(A)} = S^{(A')}$ as long as the following conditions hold true:*

C1: $|\mathbf{y}| = 0 \pmod 2$

C2: $A\mathbf{y} = \mathbf{0}$

C3: $\chi(A, \mathbf{z})\ \mathbf{y} = \mathbf{0}$.

where we define $\chi(A, \mathbf{z})$ as follows:

Definition 4.2. χ **Matrix.** Given some gate synthesis matrix, A, and a column vector $\mathbf{z} \in \mathbb{Z}_2^n$ let

$$\chi(A, \mathbf{z}) = \begin{pmatrix} (z_1 \mathbf{r_2} \wedge \mathbf{r_3}) \oplus (z_2 \mathbf{r_3} \wedge \mathbf{r_1}) \oplus (z_3 \mathbf{r_1} \wedge \mathbf{r_2}) \\ (z_1 \mathbf{r_2} \wedge \mathbf{r_4}) \oplus (z_2 \mathbf{r_4} \wedge \mathbf{r_1}) \oplus (z_4 \mathbf{r_1} \wedge \mathbf{r_2}) \\ \vdots \\ (z_{n-2} \mathbf{r_{n-1}} \wedge \mathbf{r_n}) \oplus (z_{n-1} \mathbf{r_n} \wedge \mathbf{r_{n-2}}) \oplus (z_n \mathbf{r_{n-2}} \wedge \mathbf{r_{n-1}}) \end{pmatrix},$$

where $\mathbf{r_i}$ is the i^{th} row of A, and $\mathbf{x} \wedge \mathbf{y}$ is the element-wise product of vectors $\mathbf{x}$ and $\mathbf{y}$.

Proof. We begin the proof by finding an expression for $S(A')$ using equation 6,

$$S^{(A')}_{\alpha,\beta,\gamma} = \sum_{j=1}^{m} (A_{\alpha,j} \oplus z_\alpha y_j)(A_{\beta,j} \oplus z_\beta y_j)(A_{\gamma,j} \oplus z_\gamma y_j) \pmod 2, \tag{29}$$

and expanding the brackets,

$$\begin{aligned} S^{(A')}_{\alpha,\beta,\gamma} = \sum_{j=1}^{m} (&A_{\alpha,j} A_{\beta,j} A_{\gamma,j} \oplus z_\alpha z_\beta z_\gamma y_j \\ &\oplus z_\alpha z_\beta A_{\gamma,j} y_j \oplus z_\beta z_\gamma A_{\alpha,j} y_j \oplus z_\gamma z_\alpha A_{\beta,j} y_j \\ &\oplus z_\alpha A_{\beta,j} A_{\gamma,j} y_j \oplus z_\beta A_{\gamma,j} A_{\alpha,j} y_j \oplus z_\gamma A_{\alpha,j} A_{\beta,j} y_j) \pmod 2. \end{aligned} \tag{30}$$

We can see that the first term of equation 30 summed over all j is equal to $S^{(A)}$, by definition. The task is to show that the remaining terms sum to zero under the specified conditions. Next, we use the definitions of $|\mathbf{y}|$, $A\mathbf{y}$ and $\chi(A, \mathbf{z})\ \mathbf{z}$ to simplify the result,

$$S^{(A')}_{\alpha,\beta,\gamma} = S^{(A)}_{\alpha,\beta,\gamma} \oplus z_\alpha z_\beta z_\gamma |\mathbf{y}| \oplus z_\alpha z_\beta [A\mathbf{y}]_\gamma \oplus z_\beta z_\gamma [A\mathbf{y}]_\alpha \oplus z_\gamma z_\alpha [A\mathbf{y}]_\beta \oplus [\chi(A, \mathbf{z})\ \mathbf{y}]_{\iota(\alpha,\beta,\gamma)}, \tag{31}$$

where we $\iota(\alpha, \beta, \gamma)$ is the row index of χ whose elements are given by $(z_\alpha \mathbf{r}_\beta \wedge \mathbf{r}_\gamma) \oplus (z_\beta \mathbf{r}_\gamma \wedge \mathbf{r}_\alpha) \oplus (z_\gamma \mathbf{r}_\alpha \wedge \mathbf{r}_\beta)$. By applying condition *C1*, the second term is eliminated; by applying condition *C2*, the next three terms are eliminated, and by applying condition *C3*, the final term is eliminated. □

Now we have shown how to duplicate and destroy columns of a gate synthesis matrix, we are ready to describe the TODD algorithm, presented as pseudo-code in algorithm 1. Given an input gate synthesis matrix A with signature tensor S, we begin by iterating through all column pairs of A given by indices a, b. We construct the vector $\mathbf{z} = \mathbf{c}_a \oplus \mathbf{c}_b$ where $\mathbf{c}_j$ is the j^{th} column of A, as in lemma 4.2. We check to see if the conditions in lemma 4.3 are satisfied for $\mathbf{z}$ by forming the matrix,

$$\tilde{A} = \begin{pmatrix} A \\ \chi(A, \mathbf{z}) \end{pmatrix}. \tag{32}$$

Any vector, $\mathbf{y}$, in the null space of $\tilde{A}$ satisfies *C2* and *C3* of lemma 4.3. We scan through the null space basis until we find a $\mathbf{y}$ such that $y_a \oplus y_b = 1$. At this stage we know that we can remove at least one column from A. If $|\mathbf{y}| = 0 \pmod 2$ then condition *C1* is satisfied and we can perform the duplication transformation from lemma 4.3. Otherwise, we force *C1* to be satisfied by appending a 1 to $\mathbf{y}$ and an all-zero column to A before applying the duplication transformation. Finally, we duplicate columns by adding the value of $\mathbf{z}$ to every column j for which $y_j = 1$, then destroy the a^{th} and b^{th} columns of A. This reduces the number of columns of A and therefore the T count of U_f. We now start again from the beginning, iterating over columns of the new A matrix. The algorithm terminates if every column pair has been exhausted without success.

Algorithm 1 Third Order Duplicate-then-Destroy (TODD) Algorithm

Input: Gate synthesis matrix $A \in \mathbb{Z}_2^{(n,m)}$.

Output: Gate synthesis matrix $A' \in \mathbb{Z}_2^{(n,m')}$ such that $m' \leq m$ and $S^{(A')} = S^{(A)}$.

- Let $\text{col}_j(A)$ be a function that returns the j^{th} column of A.
- Let $\text{cols}(A)$ be a function that returns the number of columns of A.
- Let $\text{nullspace}(A)$ be a function that returns a matrix whose columns generate the right nullspace of A.
- Let simplify(A) be a function that removes every pair of identical columns and every all-zero column from A.

procedure TODD
  Initialize $A' \leftarrow A$
start:
  for all $1 \leq a < b \leq \text{cols}(A')$ **do**
    $\mathbf{z} \leftarrow \text{col}_a(A') + \text{col}_b(A')$
    $\tilde{A} \leftarrow \begin{pmatrix} A' \\ \chi(A', \mathbf{z}) \end{pmatrix}$
    $N \leftarrow \text{nullspace}(\tilde{A})$
    for all $1 \leq k \leq \text{cols}(N)$ **do**
      $\mathbf{y} \leftarrow \text{col}_k(N)$
      if $y_a \oplus y_b = 1$ **then**
        if $|\mathbf{y}| = 1 \pmod 2$ **then**
          $A' \leftarrow \begin{pmatrix} A' & \mathbf{0} \end{pmatrix}$
          $\mathbf{y} \leftarrow \begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix}$
        $A' \leftarrow A' + \mathbf{z}\mathbf{y}^T$
        simplify(A′)
        goto *start*

5 Results

Our T gate optimization protocol was implemented in C++ in order to achieve two goals: first benchmark test our different *T-Optimise* algorithms against one another using random circuits (see figure 4); and second to evaluate the performance of the full Clifford + T protocol against the best known circuit decompositions for useful quantum algorithms, the results of which are listed in table 1. We see from both results that TODD produces quantum circuits with smallest T count.

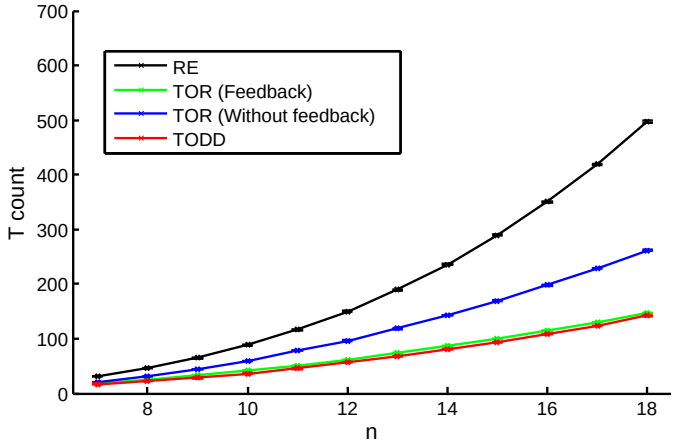

Figure 4: Circuits generated by the CNOT and T gate were randomly generated for varying number of qubits n then optimized by our implementations of RE, TOR and TODD. The average T-count for each n over many random circuits are shown on the vertical axis. TODD produces circuit decompositions with the smallest T-counts on average but scales the same as the next best algorithm, TOR (Feedback). Both of these algorithms are better than RE by a factor n. The difference between the T-counts for TODD and TOR (Feedback) converge on a constant 5.5 ± 0.7 for large n.

Table 1: T-counts for various universal Clifford + T benchmark circuits as synthesized by the TODD and RE algorithm are shown. T_{original} are the best known results produced by the works cited in the *Circuit* column, and T_{TODD} is the result for TODD. $n_{\textbf{original}}$ is the number of qubits of the original circuit and $n_{\textbf{out}}$ is the total number of qubits of the output circuit including ancillas used to implement multiply controlled Toffoli gates as well as Hadamards using path variables [24]. The total execution time in seconds for TODD run on an *Intel i7* 2.40Gz processor is given in column *Time*.

Circuit	$T_{\textbf{original}}$	$T_{\textbf{TODD}}$	$T_{\textbf{RE}}$	$n_{\textbf{original}}$	$n_{\textbf{out}}$	**Time (s)**
hwb6_47_107 [26]	71	55	102	6	43	91.713
hwb6-42-150 [26]	71	46	140	6	43	160.198
nth_prime6_inc_55_667 [26]	400	263	354	6	39	228.025
ham15-109-214 [26]	97	28	65	15	47	27.756
ham15-70 [26]	230	103	148	15	47	100.899
ham15tc1 [26]	1019	258	359	15	50	270.862
#*0117* [31]	79	13	63	6	116	118.64
#*017F* [31]	80	23	59	6	59	20.618
#*0001* [31]	40	21	46	6	39	5.2
#*001F* [31]	43	22	55	6	39	4.382
#*0007* [31]	47	13	31	6	64	10.141
#*007F* [31]	40	19	46	6	27	1.267

6 Discussion

7 Conclusion

8 Acknowledgements

References

[1] A. Y. Kitaev, A. Shen, and M. N. Vyalyi, *Classical and quantum computation.* American Mathematical Society Providence, 2002, vol. 47.

[2] C. M. Dawson and M. A. Nielsen, "The solovay-kitaev algorithm," *arXiv preprint quant-ph/0505030*, 2005.

[3] A. G. Fowler, "Constructing arbitrary steane code single logical qubit fault-tolerant gates," *Quantum Information & Computation*, vol. 11, no. 9-10, pp. 867–873, 2011.

[4] V. Kliuchnikov, D. Maslov, and M. Mosca, "Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits," *Physical review letters*, vol. 110, no. 19, p. 190502, 2013.

[5] P. Selinger, "Quantum circuits of t-depth one," *Physical Review A*, vol. 87, no. 4, p. 042302, 2013.

[6] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, "An algorithm for the t-count," *Quantum Information & Computation*, vol. 14, no. 15-16, pp. 1261–1276, 2014.

[7] N. J. Ross and P. Selinger, "Optimal ancilla-free clifford+ t approximation of z-rotations," *Quant. Inf. and Comp.*, vol. 16, p. 901, 2016.

[8] E. T. Campbell, B. M. Terhal, and C. Vuillot, "The steep road towards robust and universal quantum computation," *arXiv preprint arXiv:1612.07330*, 2016.

[9] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, p. 022316, 2005.

[10] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New Journal of Physics*, vol. 9, p. 199, 2007.

[11] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, p. 032324, Sep 2012. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.86.032324

[12] J. O'Gorman and E. T. Campbell, "Quantum computation with realistic magic-state factories," *Physical Review A*, vol. 95, no. 3, p. 032338, 2017.

[13] "Gridsynth command line tool," http://www.mathstat.dal.ca/ selinger/newsynth/.

[14] A. Paetznick and K. M. Svore, "Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries," *Quantum Information & Computation*, vol. 14, no. 15-16, pp. 1277–1301, 2014.

[15] A. Bocharov, M. Roetteler, and K. M. Svore, "Efficient synthesis of probabilistic quantum circuits with fallback," *Physical Review A*, vol. 91, no. 5, p. 052317, 2015.

[16] ——, "Efficient synthesis of universal repeat-until-success quantum circuits," *Phys. Rev. Lett.*, vol. 114, p. 080502, Feb 2015. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.114.080502

[17] E. Campbell, "Shorter gate sequences for quantum computing by mixing unitaries," *Physical Review A*, vol. 95, no. 4, p. 042306, 2017.

[18] M. B. Hastings, "Turning gate synthesis errors into incoherent errors," *arXiv preprint arXiv:1612.01011*, 2016.

[19] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013.

[20] M. Amy and M. Mosca, "T-count optimization and reed-muller codes," *arXiv preprint arXiv:1601.07363*, 2016.

[21] G. Seroussi and A. Lempel, "Factorization of symmetric matrices and trace-orthogonal bases in finite fields," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 758–767, 1980.

[22] E. T. Campbell and M. Howard, "Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost," *Phys. Rev. A*, vol. 95, p. 022316, Feb 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.95.022316

[23] A. Lempel, "Matrix factorization over gf(2) and trace-orthogonal bases of gf(2ˆn)," *SIAM Journal on Computing*, vol. 4, no. 2, pp. 175–186, 1975.

[24] A. Montanaro, "Quantum circuits and low-degree polynomials over $\mathbb{F}_2$," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 8, p. 084002, 2017. [Online]. Available: http://stacks.iop.org/1751-8121/50/i=8/a=084002

[25] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2010. [Online]. Available: http://rspa.royalsocietypublishing.org/content/early/2010/08/05/rspa.2010.0301

[26] M. Amy and M. Mosca, "T-count optimization and Reed-Muller codes," *arXiv*, vol. arXiv:1601.07363v1 [quant-ph], 2016.

[27] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, June 2013.

[28] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, Oct 2014.

[29] E. T. Campbell and M. Howard, "Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost," *Phys. Rev. A*, vol. 95, no. 022316, 2017.

[30] A. Lempel, "Matrix factorization over GF(2) and trace-orthogonal bases of GF(2)," *SIAM J. Comput.*, vol. 4, no. 2, 1975.

[31] M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, "Logic synthesis for quantum computing," *arXiv preprint*, vol. arXiv:1706.02721, 2017.

A Lempel's Factoring Algorithm