Phase Polynomials and Lempel's Algorithm

Soundarya Krishnan, Dr. Neil Julien Ross

June 2019

1 Introduction

While the structure of single-qubit Clifford+T circuits is remarkably well-understood, the structure of multi-qubit Clifford+T circuits remain elusive. For this reason, certain restricted classes of circuits received increased attention over the last few years and a particular emphasis was placed on circuits over the gate set $\{\text{CX}, \text{T}\}$. These circuits, known as CNOT+T circuits, can be conveniently represented in the formalism of phase polynomials [9] which was leveraged to define circuit optimization procedures [3], to study equality between circuits [1], and to explore connections to Reed-Muller codes 19

[4]. In recent work, an algorithm of Lempel [8] was used to reduce the T-count of CNOT+T circuits [7].

The goal of this research project is to give a detailed analysis of the methods defined in [7], which use Lempel's algorithm [8] for T-count reduction. Once the methods of [7] have been properly analyzed, the project can be extended in a variety directions. A small selection of related questions are listed below.

- For which type of circuits can the optimization algorithm of [7] be shown to produce T-optimal circuits?
- Can the H gate help lower the T-count of CNOT+T circuits? That is, are there CNOT+T operators with a Clifford+T representation of lower T-count than any of their CNOT+T representations?
- Can similar methods as the ones developed in [3, 4, 7] be applied to the study of generalized permutation circuits, which are circuits over the gate set $\{\text{X}, \text{CX}, \text{CCX}, \text{T}\}$ [2]?

2 Gate Synthesis Algorithm

Now, let's try to look at the algorithm to get the minimum number of T gates, given a particular CNOT+T circuit. We use a particular restricted case of circuits, and we will see how we can use this algorithm to generalize to more circuits in the following section.

2.1 Phase Polynomials

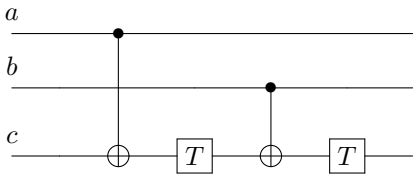
Definition 2.1. Phase polynomial P(x) is

$$P(x) = \sum_{y \in Z_2^n} a_y (y_1 x_1 \oplus y_2 x_2 \oplus y_3 x_3 \oplus \ldots \oplus y_n x_n)$$

where a_y are integers modulo 8.

In order to understand what phase polynomials are, and why they are integral to a CNOT+T circuits, let's take up an example.

We know that the action of T gate on a qubit is $|a\rangle \mapsto \omega^a |a\rangle$ (where $\omega = e^{i\pi/4}$), and the action of a CNOT gate on qubits a and b is $a \oplus b$.



1. After the action of the first CNOT gate, the last qubit becomes $a \oplus c$.
2. The 1st T gate captures a phase $\omega^{a \oplus c}$
3. The 2nd T gate captures a phase $\omega^{a \oplus c + b \oplus c}$

Thus, we can roughly see how these phase polynomials are built from CNOT+T circuits.

To understand the definition better, note that any term in the phase polynomial of an arbitrary 2 qubit circuit will be of the form

$$a_1 x_1 + a_2 x_2 + a_3 (x_1 \oplus x_2)$$

The coefficients a_1, a_2, and so on give us an idea of how many times these T gates appear in the circuit, and which bit they act on.

Comparing it to the definition, a_y for the 1st term is a_1, a_2 for the second term and so on. For the first term, $y_1 = 1, y_2$ is 0. For the second term, $y_1 = 0, y_2 = 1$, and for the third term, $y_1 = 1$ and $y_2 = 1$.

Overall, it's important to remember that the inner product is evaluated modulo 2, due to the XOR operation, and overall the function is evaluated modulo 8 as ω^8 is 1.

If this notation is confusing, all that this means is that the phase polynomial, in general, is a summation of XOR's. For example, if we had 4 qubits, a, b, c, d, then all the terms in the phase polynomial will be of the form

$$x_1 a + x_2 b + x_3 c + x_4 d + x_5 (a \oplus b) + x_6 (a \oplus c) + \ldots + x_n (a \oplus b \oplus c \oplus d)$$

where $x_1, x_2 \ldots x_n$ are integer coefficients (mod 8).

Proposition 2.2. *Every CNOT+T circuit can be represented by a phase polynomial and a reversible function.*

The phase polynomial comes from the T gates, and the reversible function is due to the diagonal CNOT circuit.

The phase polynomial can be represented in various bases. One way is to write it in the 'additive basis', using the ω, T, U, V gates. [1]

$$P(x) = a_0 + \sum_i a_i x_i + \sum_{i<j} b_{i,j} (x_i \oplus x_j) + \sum_{i<j<k} c_{i,j,k} (x_i \oplus x_j \oplus x_k)$$

We see that these can be represented using the ω, T, U and V gates. [1]

Another way is to express this in a different basis (multiplicative basis). We use the fact that $a \oplus b = a + b - 2a.b$.

Definition 2.3.

$$P(x) = \sum_i l_i x_i + 2 \sum q_{i,j} x_i x_j + 4 \sum_{i<j<k} c_{i,j,k} x_i x_j x_k$$

where $l_i, q_{i,j}, c_{i,j,k}$ are integers modulo 8.

We can prove this through the principle of mathematical induction.

Proof. From our definition of phase polynomials, let's try to construct the base case. This will be the 2 qubit case, as CNOT gates require a minimum of 2 qubits to operate on. Thus, the most basic arbitrary polynomial on a 2 qubit CNOT+T circuit will have a phase polynomial of the form

$$a_1 x_1 + a_2 x_2 + a_3 (x_1 \oplus x_2)$$

. This has already been elaborated on.

Base Case: Looking at the third term and expanding, we obtain $x_1 + x_2 - 2x_1x_2$. The total polynomial becomes

$$(a_1 + a_3)x_1 + (a_2 + a_3)x_2 - 2a_3(x_1x_2)$$

and this is of the required form, as the quadratic term comes with an even coefficient. $l_1 = a_1 + a_3$, $l_2 = a_2 + a_3$, $q_{1,2} = -a_3$ and the other terms are all 0.

Inductive Step: Assume that the phase polynomial for some m-1 qubits is already of the form required. The phase polynomial of m qubits must be a linear combination of terms of the form

$$(\sum_i l_i x_i + 2\sum q_{i,j}x_ix_j + 4\sum_{i<j<k} c_{i,j,k}x_ix_jx_k) \oplus x_m$$

Note that the phase polynomial for m-1 qubits is summed over m-1 variables.

Expanding using $a \oplus b = a + b - 2a.b$, we get

$$\sum_i l_i x_i + 2\sum q_{i,j}x_ix_j + 4\sum_{i<j<k} c_{i,j,k}x_ix_jx_k + x_m - 2\sum_i l_i x_i x_m - 4\sum q_{i,j}x_ix_jx_m - 8\sum_{i<j<k} c_{i,j,k}x_ix_jx_kx_m$$

Note that the term $-2\sum_i l_i x_i x_m$ is now quadratic instead of cubic, and has an additional factor of 2. Similar for the term $-4\sum q_{i,j}x_ix_jx_m$, which is now cubic, and has a factor of 4. The last term disappears modulo 8, and thus, we are left with

$$\sum_i l'_i x_i + 2\sum q'_{i,j}x_ix_j + 4\sum_{i<j<k} c'_{i,j,k}x_ix_jx_k$$

and the sum is over m qubits now instead of m-1, and the polynomial is of the required form. The general polynomial of m qubits is a linear combination of terms of this form, and thus remains in this form. Thus proved. □

Intuitively, we are decomposing to the multiplicative basis, where the terms are in the T, CS, CCZ basis from the CNOT+T basis. The matrices of S and Z are given along with the CS and CCZ circuits to illustrate this further.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

1. Action of CS gate can be written as

   $$|ab\rangle \mapsto \omega^{2ab}|ab\rangle$$

   , where a is the control bit, b is the target, and the phase $\omega^2 = i$ comes about due to the S gate and a has to be multiplied as it is the control bit.

2. Similarly, action of CCZ gate can be written as $|ab\rangle \mapsto \omega^{4abc}|abc\rangle$ as here the phase is $\omega^4 = -1$, and both a and b act as the control bits.

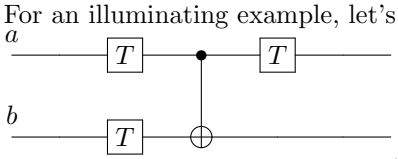
   As a note, notice that in case if there were quadratic terms with odd coefficients, we would use a Controlled-T. The T arises due to odd powers of ω and the single control comes due to the quadratic term. In case if we had a cubic term with even coefficients, for example, we would have to use a doubly controlled S gate, for the same reason. We will see more about this in a later section.

   The reason we transform to this basis will become clear in the following section when we speak about the optimization problem that we are interested in.

2.2 The optimization problem

First of all, what exactly are we attempting to do? We are attempting to reduce the count of T gates, because the T gates take up most of the cost. Some gates, for example, S gates (T^2) take up significantly lesser cost.

For an illuminating example, let's look at a concrete circuit.



The phase polynomial $P(x) = \omega^{a+b+a}$. This can be simplified to $P(x) = \omega^{2a+b}$, or, the 2 T gates can be merged to form a single S gate, which is computationally cheaper than a T gate, let alone 2 T gates. This is the type of simplification that we wish to find an algorithm for.

Lemma 2.4. *If C is a CNOT+T circuit, then there exists a diagonal circuit D and a linear circuit A such that C = DA. [1]*

Intuitively, this is equivalent to pushing all the non-diagonal operators together such that they change the basis vectors and then applying T gates to get the phase. Now, we note that we are not concerned about the non-diagonal operators, because they are 'Clifford', or in a sense, "free". The gate that takes up maximum cost is the T gate, and thus, we reduce our circuit cost minimization problem to one of minimizing the count of T gates. Thus, all the unitary operators we are working with are diagonal.

However, we don't stop here. We reduce our problem further for ease of computation.

Now, let's define our optimization problems. We'll first formally define the minimum T count, and then the extension that we will be working with.

Definition 2.5.

$$\tau[U] := min\{t | U = C_1 T_1 C_2 \dots T_t C_n; C_1, \dots C_t \in C^*\}$$

where C^* is the Clifford group which can be implemented using CNOTs and S gates. It is interesting to note that these are "CNOT+T" Cliffords. A natural generalization to this question would be what would happen if we include H too. We will deal with this question in the extensions section.

We split $U \in D_3$ (Diagonal operators in the 3rd level of the Clifford Hierarchy) [7] into unitaries V and W, where W consists entirely of CCZ gates. Note that we are looking only at diagonal unitaries due to Lemma 2.1. In other words, we are factoring out the cubic terms in the phase polynomial, and the cost of building CCZ gates is relatively lower. This eases the computation cost, and reduces a tensor factorisation problem into a matrix factorisation one.

Definition 2.6. $U = VW, W \in D_3^C, V \in D_3$, where D_3^C is the subgroup of D_3 composed of CCZ gates.

Now, let's define the optimization problem of interest to us.

Definition 2.7.

$$\mu[U] := min\{t | U = VW, t = \tau[U]\}$$

V and W are the same as the previous definition. We see that now our problem of minimizing T count has become a double optimization problem in which first, we need to find the minimal factorization of U into V and W and then need to minimize the T count in V. Later, we will see how this boils down to a single optimization problem. Also, look at section 2.5 for a concrete example on how factoring U can reduce T count further.

2.3 Reducing the problem to a tensor factorization problem

We define a signature tensor which is a way of representing this phase polynomial.

Definition 2.8. $S^{U_f}\ \ Z_2^{(n,n,n)}$ is defined to be a symmetric tensor of order 3 whose elements are constructed as

$S_{\sigma(\alpha,\alpha,\alpha)} = S_{a,a,a} = l_\alpha$
$S_{\sigma(\alpha,\beta,\beta)} = S_{(\alpha,\alpha,\beta)} = q_{\alpha,\beta}$
$S_{\sigma(\alpha,\beta,\gamma)} = c_{\alpha,\beta,\gamma}$

for all permutations of the indices, denoted σ. It follows that any two unitaries with the same signature tensor are Clifford equivalent. [6]

Now, we need a way to express the fact that it is somehow computationally cheaper to make an S gate than T gates. That is, if we have 2 T gates acting on the same state, that is equivalent to a single S gate whose costs are cheaper than T gates. This notion can be summarized as "Clifford equivalence" since S is a Clifford gate.

Definition 2.9. Unitaries U_F and $U_{F'}$ are Clifford equivalent whenever there exists an $\tilde{F}$ such that $F = F' + 2\tilde{F}$ (mod 8)

Thus, by this definition, we can say that the phase polynomial $x_1 + 2x_2 \sim_C x_1$. In other words, we can arbitrarily add or subtract any even polynomial and the circuit will have the same T count.

Before we proceed, we introduce the notion of a gate synthesis matrix, or a matrix that is characteristic of the T gates of a quantum circuit.

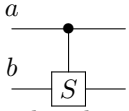
Definition 2.10. A matrix A in $Z_2^{(n,m)}$ is a gate synthesis matrix for a unitary U_f if it satisfies

$f(x) \sim_C |A^T x|$ (mod 8) $\sim_C \sum_j [\oplus A_{i,j} x_i]$ (mod 8)

Specifically, A is a matrix where the column vector u appears once if and only if $a_u = 1$ (mod 2) (Where a_u is the coefficient associated with column vector u).

The second term represents matrix multiplication with the addition replaced by an $\oplus$, and the sum over j represents the fact that we are taking the "Hamming weight" of the matrix, or the sum of all its elements.

As an example, let's take the circuit with a single CS gate.



The phase polynomial is $P(x) = 2x_1x_2$. Transforming back from the multiplicative basis, we obtain $P(x) = x_1 + x_2 - (x_1 \oplus x_2)$. Modulo 8, this is further equivalent to $x_1 + x_2 + 7(x_1 \oplus x_2)$. This is Clifford equivalent to $x_1 + x_2 + (x_1 \oplus x_2)$ (Taking $\tilde{F} = 3(x_1 \oplus x_2)$)

The vectors here are

$$x_1 : (1, 0)$$
$$x_2 : (0, 1)$$
$$x_1 \oplus x_2 : (1, 1)$$

and all of them have odd coefficients. Thus, all 3 columns will be present in A.

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

We see that $|A^T x| \sim_C f(x)$ (mod 8)

$$A^T x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 \oplus x_2 \end{bmatrix}$$

The hamming weight of the resulting matrix is Clifford equivalent to $f(x)$.

From this example, we also see that the number of columns of the gate synthesis matrix tells us about the T count.

Now, the tensor thus obtained should be "factorized" in such a way that we obtain a matrix from which we can construct back the reduced circuit.

Lemma 2.11. *The signature tensor of U_F can be determined from a gate synthesis matrix of U_F using the following relation*

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

Where m is the number of columns of A.

This is the 3 dimensional factorization required. There exists an algorithm for this, namely, the Lempel algorithm, but it does not have an optimal solution in the tensor case. If we reduce this tensor problem to a matrix problem, then we do have an efficient algorithm. To do so, we realize that CCZ gates are cheap to implement as a whole. So, we could factor out the CCZ gates, which is what we spoke about in section 2.2. In doing so, we are in a way neglecting the cubic terms, and thus, decreasing a dimension from the tensor.

Now let's define our Quadratic matrix (similar to the Signature tensor, but doesn't involve the cubic terms)

Definition 2.12. $Q^{U_F} \quad Z_2^{(n,n)}$ is defined to be a symmetric matrix whose elements are constructed as
$Q_{i,i} = l_i \pmod 2$
$Q_{i,j} = Q_{j,i} = q_{i,j} \pmod 2$

As a specialization of the tensor case, we reduce this to a matrix factorization problem as follows.

Lemma 2.13. *Let unitary U have quadratic matrix Q and unitary V have gate synthesis matrix B. It follows that $Q = BB^T$ (mod 2) if and only if $F_U \sim_\mu F_V$*
Thus, the optimization problem becomes

$$\mu[U] = min\{col(B) | Q = B.B^T\}$$

This is where Lempel's algorithm comes into picture. As mentioned earlier, Lempel's algorithm has an optimal solution for matrices, and is stated in the following section.

2.4 A rough working of Lempel's algorithm

Let A $(A_{i,j})$ be a symmetric matrix of rank $\rho(A)$ and let

$$\delta(A) = \begin{cases} 1 & if \text{A}_{i,i} = 0 \text{ for all i} \\ 0 & otherwise \end{cases} \tag{1}$$

A matrix B is called a factor of A if $A = BB^T$, where B^T is the transpose of B. B is called a minimal factor of A if no factor of A has fewer columns than B. The number of columns of a minimal factor of A will be denoted by $\mu(A)$. The main result can now be stated as follows.

Theorem 2.14. *Every binary, symmetric matrix A has a factor over GF(2), and*

$$\mu(A) = \rho(A) + \delta(A)$$

.

Here, we describe a rough working of Lempel's algorithm, along with a brief understanding of why the steps follow. The following is a brief idea of why certain steps follow, and rigorous proofs are outside the scope of this paper.
(Go to the next section for a full example)
Step 1:
Obtain the elementary factor, E. Before that, let n be the order of the matrix A to be factorized. N= 1, 2, ... n We first define subsets N_1 and N_2 of N as

Definition 2.15.

$$N_1 := \{k \in N | \sum_{j=1}^{n} A_{k,j} = 1\}$$

$$N_2 := \{(i,j) | i,j \in N, i < j, A_{i,j} = 1\}$$

.

Thus, to obtain the elementary factor E, we take the N_1, the column numbers of columns that have an odd number of ones. N_1 will have exactly one 1 in the column number of the column with odd number of ones and 0s elsewhere. We concatenate this with N_2, the coordinates of all the ones in the upper triangle of the quadratic matrix (excluding diagonal). In N_2 also, the column vector will have exactly 2 ones (in the coordinates) and 0s elsewhere.

Take the matrix

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

We see that N_1= {1, 3} and N_2={(1, 2), (1, 3), (2, 4), (3, 4)}
Thus, we obtain E as

$$E = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The logic behind how this construction gives back Q is that

$$(EE^T)_{i,j} = \sum_k (E_{i,k} E_{j,k})$$

$$= \begin{cases} 1 & if \text{E contains an (i, j) column} \\ 0 & otherwise \end{cases} \tag{2}$$

for $i \neq j$. This holds true as when E is multiplied by its own transpose, it is as if you are multiplying 2 of its rows together. Only the term with both 1's in both the i^{th} and j^{th} rows will give 1 as a product when multiplied. Note that N_1 columns do not disrupt this as it will always have only a single 1 element, and thus cannot multiply with anything else.

For the diagonal elements, N_1 can be thought of as a correction term.

More formally, note that $(EE^T)_{kk} = \sum_j E_{kj}$

E_{kj} can be 1 if and only if one of the following 3 alternatives hold:

- The jth column of E is a k column. (From N_1)
- The jth column of E is an (i, k)-column, i¡k (1st 1 from N_2)
- The jth column of E is an (k, j)-column, i¡k (2nd 1 from N_2)

This could be summarised in the equation below.

$$(EE^T)_{kk} = \sum_j A_{kj} + \sum_{i<k} A_{ki} + \sum_{j>k} A_{kj}) = A_{kk}$$

Step 2:
Find a proper subset of columns of B whose sum is zero, call the submatrix formed by these columns G, and partition B as B [F G]. This step works due to Lemma 4 of Lempel's paper.

Lemma 2.16. *If A is nonsingular, B is a factor of A and $c(B) > \rho(A) + \delta(A)$, then B contains a proper subset of columns whose sum is zero.*

The proof of this can be illustrated as follows. If $A = BB^T$ and A is nonsingular, then $\rho(A) = \rho(B)$. With $c(B) > \rho(A)+\delta(A)$, we obtain $c(B) > \rho(B)+\delta(A)$. Hence, regardless of the value of $\delta(A)$, the columns of B are linearly dependent, and by the definition of linear dependence, must sum to 0.
Step 3:
Substitute Z for G, where Z is given as follows:

$$Z = \begin{cases} G & if\ c(G)\ is\ even \\ [\text{G } 0] & otherwise \end{cases} \tag{3}$$

where [G 0] is the matrix obtained by adjoining an all-zero column to G. Now, $B^* = [FZ]$.

This doesn't make any difference to the product, as $ZZ^T = GG^T$, and thus $BB^T = B^*(B^*)^T$. We'll see the reason for making the number of columns even shortly.

Next, Set $x = F_1 + Z_1$ and replace each column Z_j of Z by $\hat{Z}_j = Z_j + x$, to obtain $\hat{B} = [F\ \hat{Z}]$.

Lemma 2.17. *Let Z be a binary matrix such that Zu=0 and c(Z) is even, where u is a column vector of all ones. Let $\hat{Z} = Z + xu^T$, where x is an arbitrary binary vector with r(x)=r(Z). Then (3) $\hat{Z}\hat{Z}^T = ZZ^T$.*

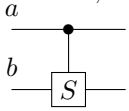
The proof is as follows: $\hat{Z}\hat{Z}^T = (Z + xu^T)(Z + xu^T)^T = (Z + xu^T)(Z^T + ux^T)$
$\hat{Z}\hat{Z}^T = ZZ^T + Zux^T + xu^TZ^T + xu^Tux^T$. Since $Zu = 0$, also $u^TZ^T = 0$. Since r(u)= c(Z) is even, $u^Tu = 0$ and, hence $\hat{Z}\hat{Z}^T = ZZ^T$. This is why we made Z with even number of columns.
Step 4:
Delete F_1 and $\hat{Z}_1$ from $\hat{B}$ to obtain the matrix $\tilde{B}$. If $c(\tilde{B}) = \mu(A)$, stop. Otherwise, set B = and go to Step 2.

This works due to the following reason. for $\hat{Z}_1$, the first column of Z, $\hat{Z}_1 = Z_1 + F_1 + Z_1 = F_1$, as something added twice modulo 2 disappears. Thus, the overall contribution of F and Z_1 to the product $\hat{B}\hat{B}^T$ is null, and hence the matrix obtained by deleting F_1 and $\hat{Z}_1$ from $\hat{B}$ is also a factor of A.

2.5 Intuition and Example

Let's see a concrete example where factorizing U into V and W actually reduces the T count than the brute force method.

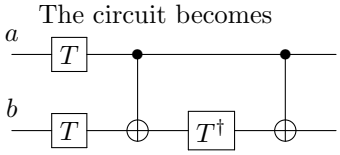
- First, let's look at controlled S, i.e.



From before, we know the action of the controlled S gate on qubits (recall section 2.1). The phase polynomial is $P(x) = 2x_1x_2$

Let's see how to decompose this in terms of T gates. Now, to see which T gates are present, we have to transform back to the XOR basis using $a \oplus b = a + b - 2ab$. Thus, our expression is $P(x) = x_1 + x_2 - x_1 \oplus x_2$. Thus, we can implement this circuit using 3 T gates, one for each term in the polynomial.

The circuit becomes



We see that the minimum count is 3 through brute force method. Now, let's apply the algorithm and see if we obtain the same answer, 3.

$$P(x) = 2x_1x_2$$

.

Step 1: Let's construct the Quadratic matrix (recall previous section). Only 1 quadratic term is present, $q_{1,2}$, and no linear terms are present. Thus, the diagonal elements are 0, and the $Q_{1,2}$, and $Q_{2,1}$ are the only terms present, due to the definition of the quadratic matrix.

$$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

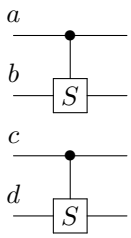
Step 2: Let's run Lempel's algorithm to get the minimal factor B. (Recall section 2.4)
Step 2.1: Obtaining the elementary factor E. We see that $N_1 = \{1, 2\}$, i.e. columns 1 and 2 have an odd number of 1's. $N_2 = \{(1, 2)\}$, i.e. only coordinate (1, 2) is a 1 in the upper triangle of the matrix Q.

$$E = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

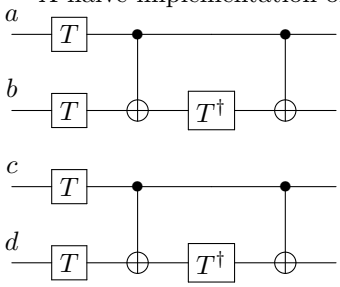
Step 2.2: We see that the number of col(B) is 3, rank(Q)=2, and Q is singular ($\delta = 1$). Thus, the stopping criterion of Lempel's algorithm, $col(B) = rank(Q) + \delta$ is satisfied. Thus, we stop here and E=B.

Now, we see that the number of columns of B, and thus, the minimum T count is 3, which agrees with our previous working.

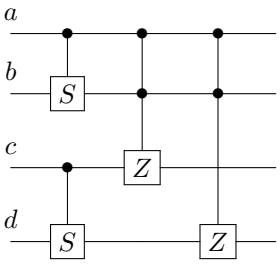
- Now, let's look at a different circuit.

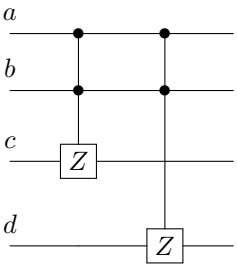


A naïve implementation of the circuit would give 6 T gates, 3 from each CS gate.



However, we can optimize this further using the U=VW decomposition mentioned earlier. By adding some CCZ gates, we are able to bring down the T count in the circuit. Again, since CCZ gates are much cheaper, we can push as many CCZ gates separately as possible.

V= 

W= 

V can be implemented with 5 T gates, and W can be implemented with 7 T gates. Thus, we see that we have brought down our minimum from 6 to 5.

Now, let's see if we obtain this number 5 from our algorithm.

$$P(x) = 2x_1x_2 + 2x_3x_4$$

.

Step 1: Let's construct the Quadratic matrix. 2 quadratic terms are present, $q_{1,2}$ and $q_{3,4}$, and no linear terms are present. Again, we obtain a singular matrix.

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 2: Let's run Lempel's algorithm to get the minimal factor B. (Recall section 2.4)
Step 2.1: Obtaining the elementary factor E. We see that $N_1 = \{1, 2, 3, 4\}$, i.e. all columns have an odd number of 1's. $N_2 = \{(1, 2), (3, 4)\}$, i.e. coordinates (1, 2) and (3, 4) have a 1 in the upper triangle of the matrix Q.

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Step 2.2: We see that the number of col(B) is 6, rank(Q)=4, and Q is singular ($\delta = 1$). Thus, the stopping criterion of Lempel's algorithm, $col(B) = rank(Q) + \delta$ is not satisfied.

Step 2.3: We need to find a subset of columns G such that Gu=0, or the rows of this subset G have an even number of ones.

We can take the subset

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The remaining columns form F.

Thus, the matrix E has been partitioned into F and G.

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Step 2.4: Now, we need to transform E=[F][G] into [Z][G]. Since G has an odd number of columns, we append a 0 column to G to form Z. Thus, the matrix becomes

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} Z = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 2.5: Now, we need to transform E=[Z][G] to E=$[\hat{Z}][G]$.

$$x = F_1 + Z_1$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = X = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\hat{Z} = Z_j + X$$

for all columns of Z, Z_j

$$\hat{Z} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 2.6: Now, we delete the first columns of F and $\hat{Z}$ to obtain the matrix B

$$\tilde{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Now, we see that the number of columns of B, and thus, the minimum T count is 5, which agrees with our previous working.

Thus, we get an idea of why partitioning U into V and W could reduce the T count even further.

In the next section, we talk about how this algorithm can be used for more general cases.

3 Extensions

3.1 The usage of Hadamards

The following is an extension of what we have been looking so far. Let $\mathcal{D}_3$ and $\mathcal{B}_3$ be the following sets of operators

- $\mathcal{D}_3 = \{U \mid U|x\rangle = \alpha|x\rangle$ and there exists a CNOT+T circuit C such that $[\![C]\!] = U\}$ and
- $\mathcal{B}_3 = \{U \mid U|x\rangle = \alpha|x\rangle$ and there exists a Clifford+T circuit C such that $[\![C]\!] = U\}$.

That is, $\mathcal{D}_3$ is the set of diagonal CNOT+T operators and $\mathcal{B}_3$ is the set of diagonal Clifford+T operators. We have

$$\mathcal{D}_3 \subsetneq \mathcal{B}_3.$$

This can be established by a counting argument, see for example Remark 6.7 in [1]. This inequality can be interpreted as stating that Hadamard gates, despite being non diagonal, are required to generate the diagonal elements of the set of Clifford+T circuits.

Now let F be a collection of Clifford+T circuits and let U be a Clifford+T operator. We define the parametrized cost function τ_F by

$$\tau_F(U) = \min\{\tau(C) \mid C \in F \text{ and } [\![C]\!] = U\}$$

where $\tau(C)$ is the T-count of C. Let $U \in \mathcal{D}_3$. Then

$$\tau_{Clifford+T}(U) \leq \tau_{CNOT+T}(U).$$

We would like to know whether this equality is sometimes strict. That is, are there elements $U \in \mathcal{D}_3$ such that $\tau_{Clifford+T}(U) < \tau_{CNOT+T}(U)$? Intuitively, we are asking whether the Hadamard gate can be used "behind the scenes" to lower the T-count of CNOT+T circuits.

The thing that makes this question a little bit hard is that the Hadamard gate does not admit a nice phase polynomial representation. One way to circumvent this problem is to consider a different set of generators with the following two properties:

1. They admit a nice phase polynomial representation.
2. They generate all of $\mathcal{B}_3$.

That way, we should be able to understand the full power of diagonal Clifford+T operators while retaining the nice qualities of the phase polynomial representation.

I believe that the set of generators $\mathcal{P}$ consisting of X, CX, CCX, and T should do this. I think it follows from [5] that they generate all of $\mathcal{B}_3$ and I think that they should admit a nice phase polynomial representation.

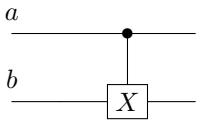
So here are some items for a To-Do list:

1. Show that $\mathcal{P}$ generates all of $\mathcal{B}_3$. The presence or the absence of ancillas should be considered carefully.
2. Characterize precisely the phase polynomial representation of circuits over $\mathcal{P}$.
3. Understand what techniques from [7] apply to circuits over $\mathcal{P}$.

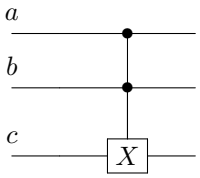
3.2 Phase polynomials from the new gate set

As we have seen in the previous section (yet to be completed), we wish to examine the diagonal operators of the Clifford+T gate set, $\langle CNOT, H, T\rangle$. As proved in the previous section, these are also generated by the gate set $\langle X, CNOT, CCX, T\rangle$. The reason we had ported to this gate set instead, is that due to the lack of the Hadamard gate, the action of these gates on a vector can be represented effectively using the phase polynomial representation that we have studied previously in the case of CNOT+T circuits. The action of the Hadamard gate complicates the phase polynomials due to the superposition states.

Now, let's look deeply at the kind of phase polynomials that this brings about.



The action of the X gate is as follows: $|a\rangle \mapsto |\bar{a}\rangle$



The action of the CCX gate is as follows: $|abc\rangle \mapsto |ab(ab \oplus c)\rangle$

3.2.1 Formation in terms of CNOT+T gate set

Firstly, let's look at X gates.

Lemma 3.1. *X gate can be exactly represented in the CNOT+T gate formulation with an ancillary qubit with the value fixed to 1.*

We can see this as, under the action of the X gate, $|a\rangle \mapsto |\bar{a}\rangle$, which is equivalent to $|a\rangle \mapsto |1 \oplus a\rangle$. Once we give this 1 to an ancilliary qubit, we can proceed to use the exact same formulation that we used earlier.

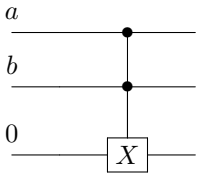
Now, looking at CCX gates,

Lemma 3.2. *The CCX gate can be exactly represented in the CNOT+T gate formulation with $\binom{N}{2}$ ancillary qubits.*

We illustrate this by noting that the term that causes the issue when including CCX gates is $(ab \oplus c)$. We see that this could be reduced to our initial CNOT+T problem if we take the composite term ab as some d, so that the term reduces to another term of the form $(d \oplus c)$, which is a problem we already know how to solve.

Now, let's see how we could prepare these composite states.

$$|a\ b\ 0\rangle \mapsto |a\ b\ (a\ b\ \oplus 0)\rangle = |a\ b\ (ab)\rangle$$



Thus, we require $\binom{N}{2}$ ancillary qubits with values fixed to 0 so that we can pre-make these into quadratic states. Note that now, the single states alongwith the quadratic states span the terms that can be obtained by using a CCX gate.

Overall, for a gate set that includes both CCX and X gates, we require $\binom{N}{2}$ +1 ancillary states if we use the CNOT+T gate set formulation.

3.2.2 Phase polynomials and the introduction of X gates

When we include X gates, the main difference in the phase polynomials is the inclusion of constant terms. More concretely, our phase polynomial becomes like this:

Definition 3.3.

$$P(x)_{\langle CNOT,T,X\rangle} = constant + \sum_i l_i x_i + 2\sum q_{i,j} x_i x_j + 4 \sum_{i<j<k} c_{i,j,k} x_i x_j x_k$$

where $l_i, q_{i,j}, c_{i,j,k}$ are integers modulo 8.

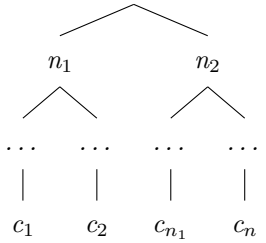
The constant terms come from the XORs with 1. ($\bar{a} = (1 \oplus a) = (1 + a - 2a)$). Now, this isn't an issue as the constant term in the phase polynomial can be simply factored out, and implemented using a scalar gate $\omega = e^{i\pi/4}$, which is clifford and doesn't change our T count.

3.2.3 Phase polynomials and the introduction of CCX gates

Now, let's look at our original gate set, $\langle X, CNOT, CCX, T\rangle$. We have already looked at what happens with the introduction of X gates. With the introduction of CCX gates, the phase polynomial gets modified.

For a better understanding, we are going to use a tree construction to denote the coming together of 2 terms of degrees x and y into a term of higher degree x+y. (Due to recursive applications of the XOR identity: $a \oplus b = a + b - 2ab$)

Lemma 3.4. *The degree of every term appearing in a phase polynomial in the multiplicative basis can be represented in the form of a strictly binary degree tree such as* $n \equiv n_1 + n_2$ *where* $c_1, c_2, \ldots c_n$



are leaves of the tree, or terms of minimum degree.

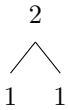
We see that:

- The tree is strictly binary. Every node has either 2 children, or is a leaf node. This is because XOR's are associative. So, we might think that a cubic term can come from 3 linear terms, and thus we may have a tree with 3 branches, i.e., from a term like $a \oplus b \oplus c$. However, since XOR's are binary operators, this is essentially $(a \oplus b) \oplus c$ or $a \oplus (b \oplus c)$. The first corresponds to a partition like 

, while the second corresponds to a partition like 

- The ordering of branches doesn't matter to the coefficients, as XORs are commutative.

Let's look at an ordinary quadratic term. We know that generally (in the CNOT+T gate set), a quadratic term can only arise from 2 monomial terms (a that the minimum degree of terms is 1) ($a \oplus b = a + b - 2ab$)



The joining of two branches represents a transformation to the multiplicative basis. We also see that every such union multiplies 2 with the coefficients of the 2 child nodes, again due to the XOR identity.

Lemma 3.5. *The coefficient of any term of degree the same as the root node is always a multiple of* $2^{parent\ nodes}$.

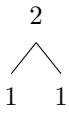
This leads us to a corollary, which functions as a recursive definition of the minimum coefficients $(MC_{gate\ set}(degree))(modulo 8)$ of terms that we are interested in.

Let the gate set $\langle X, CNOT, CCX, T\rangle$ be G1 and the gate set $\langle X, CNOT, CCX, T\rangle$ be gate set G2.

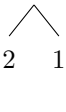
Corollary 3.6. *$MC(root)$ = Minimum over partitions $(2 * MC(leftchild) * MC(rightchild))$. $MC_{G1}(1) = MC_{G2}(1)$ of a monomial term is 1 ($a \oplus b$ contains monomials a and b with coefficients 1). For the gate set with CCX, $MC_{G2}(2)$ is also 1, as we have the polynomial term $ab \oplus c$.*

Using this framework, let's look at terms that arise from the CNOT+T (G1) gate set.

- Degree 1: By Corollary 4.3, $MC_{G1}(1) = 1$.
- Degree 2: By Lemma 4.2, $MC_{G1}(2) = 2^1$ with a single parent.



- Degree 3: By Corollary 4.3, minimum coefficient is $2 * MC_{G1}(2) * MC_{G1}(1) = 2 * 2 * 1 = 4$

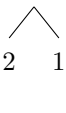


- Degree 4: Two distinct partitions of a term of degree 4 are possible.

. For the first case, $MC_{G1}(4) = 2 * 2 * 2 = 8$, and for the second case, $MC_{G1}(4) = 2 * 4 * 1 = 8$
Thus, $MC_{G1}(4) = 8$. And since these are coefficients modulo 8 (powers of ω, $MC_{G1}(4) = 0$).
Similarly, we can that higher polynomials too vanish modulo 8.

Similarly, for the new gate set, $\langle CNOT, T, X, CCX \rangle$ (G2),

- Degree 1: By Corollary 4.3, $MC_{G2}(1) = 1$.
- Degree 2: By Corollary 4.3, $MC_{G2}(2) = 1$.
- Degree 3: By Corollary 4.3, minimum coefficient is $2 * MC_{G2}(2) * MC_{G2}(1) = 2 * 1 * 1 = 2$

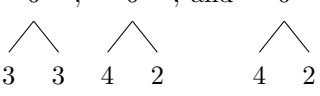


- Degree 4: Two distinct partitions of a term of degree 4 are possible.

. For the first case, $MC_{G2}(4) = 2 * 1 * 1 = 2$, and for the second case, $MC_{G2}(4) = 2 * 2 * 1 = 4$
Thus, $MC_{G2}(4) = min(4, 2) = 2$

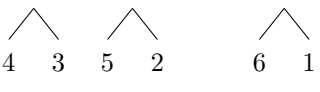
- Degree 5: Two distinct partitions of a term of degree 5 are possible.

. For the first case, $MC_{G2}(5) = 2 * 2 * 1 = 4$, and for the second case, $MC_{G2}(5) = 2 * 2 * 1 = 4$.
Thus, $MC_{G2}(5) = 4$.

- Degree 6: Three distinct partitions of a term of degree 6 are possible.  .

  For the first case, $MC_{G2}(6) = 2 * 2 * 2 = 8$, for the second case, $MC_{G2}(6) = 2 * 2 * 1 = 4$, and for the third case, $MC_{G2}(6) = 2 * 2 * 1 = 4$.

  Thus, $MC_{G2}(6) = min(4, 8) = 4$

- Degree 7: Three distinct partitions of a term of degree 7 are possible.  .

  For the first case, $MC_{G2}(7) = 2 * 2 * 2 = 8$, for the second case, $MC_{G2}(7) = 2 * 4 * 1 = 8$, and for the third case, $MC_{G2}(7) = 2 * 4 * 1 = 8$.

  Thus, $MC_{G2}(7) = 8$, or $MC_{G2}(7) = 0$ modulo 8.

- For higher terms, by a similar argument, we can argue that their coefficients vanish modulo 8.

Let's summarize these findings with the help of a table. Minimum Coefficients with different gate sets as given as follows. The gates in the brackets denote the basis gates that are required in order to form the polynomial in the multiplicative basis. In a way, they represent the basis vectors in the multiplicative basis.

Degree	CNOT+T	CNOT+T+X	CNOT+T+X+CCX
0	0	1	1
1	1 (T)	1 (T)	1 (T)
2	2 (CS)	2 (CS)	1 (CT)
3	4 (CCZ)	4 (CCZ)	2 (CCS)
4	0	0	2 (CCCS)
5	0	0	4 (CCCCZ)
6	0	0	4 (CCCCCZ)
7	0	0	0

Definition 3.7.

$$P(x)_{\langle CNOT,T,X,CCX\rangle}$$
$$= constant + \sum_i l_i x_i + \sum q_{i,j} x_i x_j + 2 \sum_{i<j<k} c_{i,j,k} x_i x_j x_k + 2 \sum_{i<j<k<l} t_{i,j,k,l} x_i x_j x_k x_l$$
$$+4 \sum_{i<j<k<l<m} p_{i,j,k,l,m} x_i x_j x_k x_l x_m + 4 \sum_{i<j<k<l<m<n} h_{i,j,k,l,m,n} x_i x_j x_k x_l x_m x_n$$

where $l_i, q_{i,j}, c_{i,j,k}, t_{i,j,k,l}, p_{i,j,k,l,m}, h_{i,j,k,l,m,n}$ are integers modulo 8.

References

[1] M. Amy, J. Chen, and N. J. Ross. A Finite Presentation of CNOT-Dihedral Operators. In *Proceedings 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen, The Netherlands, 3-7 July 2017.*, pages 84–97, 2017. Available from `arXiv:1701.00140`.

[2] M. Amy, A. N. Glaudell, and N. J. Ross. A Characterization of Integral, Real, and Gaussian Clifford+T Operators. In preparation, May 2019.

[3] 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*, 33(10):1476–1489, Oct 2014. Also available from `arXiv:1303.2042v2`.

[4] M. Amy and M. Mosca. T-Count Optimization and Reed-Muller Codes. *IEEE Transactions on Information Theory*, pages 1–1, 2019. Preprint available from `arXiv:1601.07363`.

[5] B. Giles and P. Selinger. Exact Synthesis of Multiqubit Clifford+T Circuits. *Physical Review A*, 87:032332, 2013. Preprint available from `arXiv:1212.0506`.

[6] L. E. Heyfron and E. T. Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, sep 2018.

[7] M. Howard and E. T. Campbell. A Unified Framework for Magic State Distillation and Multi-Qubit Gate-Synthesis with Reduced Resource Cost. Preprint available from `arXiv:1606.01904`, June 2016.

[8] A. Lempel. Matrix Factorization over $GF(2)$ and Trace-Orthogonal Bases of $GF(2^n)$. *SIAM Journal on Computing*, 4(2):175–186, 1975.

[9] P. Selinger. Quantum Circuits of T-Depth One. *Phys. Rev. A*, 87:042302, Apr 2013. Preprint available from `arXiv:1210.0974`.