Cost-optimal single-qubit gate synthesis in the Clifford hierarchy

Gary J. Mooney[1], Charles D. Hill[1,2], and Lloyd C.L. Hollenberg[1]

[1]School of Physics, University of Melbourne, VIC, Parkville, 3010, Australia.
[2]School of Mathematics and Statistics, University of Melbourne, VIC, Parkville, 3010, Australia.

For universal quantum computation, a major challenge to overcome for practical implementation is the large amount of resources required for fault-tolerant quantum information processing. An important aspect is implementing arbitrary unitary operators built from logical gates within the quantum error correction code. A synthesis algorithm can be used to approximate any unitary gate up to arbitrary precision by assembling sequences of logical gates chosen from a small set of universal gates that are fault-tolerantly performable while encoded in a quantum error-correction code. However, current procedures do not yet support individual assignment of base gate costs and many do not support extended sets of universal base gates. We analysed cost-optimal sequences using an exhaustive search based on Dijkstra's pathfinding algorithm for the canonical Clifford+T set of base gates and compared them to when additionally including Z-rotations from higher orders of the Clifford hierarchy. Two approaches of assigning base gate costs were used. First, costs were reduced to T-counts by recursively applying a Z-rotation catalyst circuit. Second, costs were assigned as the average numbers of raw (i.e. physical level) magic states required to directly distil and implement the gates fault-tolerantly. We found that the average sequence cost decreases by up to $54 \pm 3\%$ when using the Z-rotation catalyst circuit approach and by up to $33 \pm 2\%$ when using the magic state distillation approach. In addition, we investigated observed limitations of certain assignments of base gate costs by developing an analytic model to estimate the proportion of sets of Z-rotation gates from higher orders of the Clifford hierarchy that are found within sequences approximating random target gates.

1 Introduction

Quantum computing has the potential to solve many real-world problems by using significantly fewer physical resources and computation time than the best known classical algorithms. The quantum algorithms for these problems are implemented using deep quantum circuits. Thus to reliably implement these circuits, qubits within the devices require long coherence times and high precision control. Current systems consist of physical qubits that are too noisy for large scale computation. Error-correction schemes provide the ability to overcome this hurdle by entangling clusters of physical qubits in such a way that they collectively encode the information into more robust logical qubits. In principle, when physical qubits have error-rates below the *error threshold* of the error-correction scheme, logical qubits within the code can be made arbitrarily robust using increasing numbers of qubits. A particular error-correction scheme with relatively high physical error threshold of approximately 1% is the *surface code*, which is implemented over a nearest-neighbour two-dimensional physical layout, making it one of the most realistically implementable schemes [1–4]. In this work, we analyse the resource costs for gate synthesis, which is used to fault-tolerantly implement arbitrary unitary gates in error-correction codes.

Gary J. Mooney: mooneyg@unimelb.edu.au
Charles D. Hill: cdhill@unimelb.edu.au
Lloyd C.L. Hollenberg: lloydch@unimelb.edu.au

The surface code, among other high-threshold codes, is limited to a small set of Clifford gates over logical qubits that can be performed with relative ease. A procedure called magic state distillation can be used to perform a wider range of non-Clifford gates fault-tolerantly, such as the $T := R_z(\pi/4)$ gate (up to global phase), which cannot be produced using only Clifford gates [5, 6]. Initially, raw magic states are surgically injected into the code and with the aid of state distillation procedures, a number of raw magic states are consumed to produce a smaller number of more robust magic states. In principle, the procedures can be recursively applied to obtain states with arbitrarily low noise, although requiring large amounts of physical resources. These purified magic states can then be consumed to fault-tolerantly perform corresponding gates using quantum teleportation circuits. Distillation procedures only exist for a subset of gates, in order to implement arbitrary unitary gates, the Solovay-Kitaev (SK) theorem can be used. The SK theorem states that a universal set of n-qubit gates generate a group dense in $SU(2^n)$ (Special Unitary), and the set fills $SU(2^n)$ relatively quickly. Hence single-qubit base gates that form a universal set can be multiplied in sequence to approximate any single-qubit gate to arbitrary precision [7, 8].

A frequently used set of single-qubit universal base gates for fault-tolerant quantum computation are the Clifford+T gates, where the Clifford gates are relatively cheap to apply while the T gate requires a considerable amount of resources due to the magic state distillation procedure. This set of gates and how they can be used to synthesise arbitrary single-qubit gates is a well studied topic within the quantum compilation literature. Gate synthesis algorithms, besides brute-force [9], began with the Solovay-Kitaev algorithm [8, 10]. It initially searches for a base sequence that roughly approximates a target gate and then uses a recursive strategy to append other base sequences in such a way that the new sequence approximates a gate that is closer to the target gate with distance reducing efficiently with the number of iterations. It is compatible with arbitrary single-qubit universal gate sets, provided that they include each gate's adjoint. The SK algorithm has room for optimisation with respect to lengths of resulting gate sequences since the recursive process generates strings of disjoint subsequences which are only individually optimised, rather than optimising over the entire sequence. In 2008, Matsumoto and Amano [11] developed a normal form for sequences of Clifford+T gates that produces unique elements in $SU(2)$. Shortly after, Bocharov and Svore [12] introduced their canonical form which extends the normal form by instead producing unique elements in $PSU(2)$ (Projective Special Unitary) which more concisely describes the space of all physical single-qubit gates by ignoring global phase. This normal form can be used to enumerate length optimal sequences of Clifford+T base gates which produce distinct gates, considerably reducing the size of the sequence configuration space for search algorithms (although still growing exponentially with respect to sequence length).

More recently, there has been significant progress on developing direct synthesis methods which are not based on search. For target single-qubit unitary gates that can be exactly produced by Clifford+T base gate sequences, a method was developed that optimally and efficiently finds these exact sequences directly [13]. This was later used as a subroutine in algorithms for optimal synthesis of arbitrary single-qubit Z-rotations [14, 15]. Direct Clifford+T base gate synthesis methods for Z-rotations have since been generalised to Clifford+cyclotomic (Z-rotation by π/n) sets of base gates [16] and sets derived from totally definite quaternion algebras [17]. For arbitrary single-qubit rotations (not necessarily Z-rotations) there has been a number of other approaches developed, such as a randomised algorithm that uses the distribution of primes [18], asymptotically optimal synthesis using ancilla qubits [19], and probabilistic quantum circuits with fallback [20].

It is common within the quantum compilation literature for synthesis algorithms to optimise sequences based on minimising the total number of gates that require magic state injection. This measure is well-suited to the Clifford+T set of base gates which are standard for gate synthesis algorithms, since the T gate and its adjoint are the only gates with a significantly higher cost than the Clifford gates. However, procedures exist for performing alternative gates to the T gate that vary in implementation cost. Examples of such gates are found within the Clifford hierarchy, which is an infinite discrete set of gates that are universal and can be performed on certain error-correcting codes fault-tolerantly [21]. The resource cost of implementation typically varies between orders of the hierarchy. Thus to accurately cost optimise sequences from such sets of gates, the cost of each individual base gate should be considered. We investigate two different approaches for implementing Z-rotation gates from the Clifford hierarchy and calculating their resource costs. The first approach is based on a circuit that uses a catalyst Z-rotation state

to implement two copies of its corresponding Z-rotation gate using a small number of T gates while retaining the initial Z-rotation state [22, 23]. This circuit can enable the average resource costs of implementing Z-rotation gates from the Clifford hierarchy to be expressed as T-counts. Using this approach, costs could be calculated either by assuming that output gates are applied directly to target qubits or by assuming that all output gates are first applied to $|+\rangle$ states to form intermediate magic states, which can then be consumed to implement the corresponding gates onto target qubits at any time. As an alternative to the Z-rotation catalyst circuit approach of gate implementation, the second approach is to use the average number of raw magic states required to directly distil and implement subsets of gates belonging to the Clifford hierarchy in surface codes. The distillation costs have already been calculated by Campbell and O'Gorman [24] for various levels of precision, the accumulated costs of distilling and then implementing the gates are found within their supplementary materials. Although other factors relating to physical resources are important to consider such as qubit count, circuit depth, magic state distillation methods, and details of the error-correction implementation, the number of raw magic states can serve as a rough approximation to the cost of implementing fault-tolerant logical gates on surface codes.

We introduce an algorithm, based on Dijkstra's shortest path algorithm, that generates a database of all cost-optimal sequences below a chosen maximum sequence cost where each sequence produces distinct gates in $PSU(2)$. The algorithm supports arbitrary universal sets of single-qubit base gates with individually assigned cost values. The database can then be searched to find a sequence approximating a specified target gate. We use this algorithm to compare the cost of cost-optimal gate synthesis between the canonical Clifford+T base gate set and various sets of base gates consisting of Clifford gates and Z-rotations from higher orders of the Clifford hierarchy. Each set of logical base gates is compared by calculating how the average gate sequence cost for approximating random target gates scales with respect to reaching target gate synthesis logical error rates. When including Z-rotation base gates from higher orders of the Clifford hierarchy with T-counts assigned using the Z-rotation catalyst approach, we find that the average cost-optimal sequence T-counts can potentially be reduced by over 50% when output gates are directly applied to target qubits and by over 30% when intermediate magic states are used. When using the alternative approach of assigning costs from direct magic state distillation, we find that by including Z-rotation logical base gates from the fourth order of the Clifford hierarchy, the average cost-optimal sequence costs can be reduced by 30%. These cost reductions indicate that a significant amount of resources could be saved by adapting current synthesis algorithms to include higher orders of the Clifford hierarchy and to optimise sequences with respect to individual gate costs.

In the cases when costs are assigned using the Z-rotation catalyst method via intermediate magic states or when assigned using direct magic state distillation, we observe that there is only a small improvement to the average costs of synthesis when Z-rotations of orders higher than four of the Clifford hierarchy are included as base gates. We investigate this behaviour by developing a model to estimate the proportion of Z-rotation base gates from specified orders of the Clifford hierarchy within sequences approximating random target gates, without needing to generate the database of sequences. The proportions calculated in this manner closely fit results obtained using the sequence generation algorithm to approximate uniformly distributed random target gates. The parameters of the calculation include the maximum sequence cost and separate logical base gate costs for each order of the Clifford hierarchy, which can be readily be extended to specify costs for individual logical base gates.

Results

Base Gates From The Clifford Hierarchy

The Clifford hierarchy is an infinite discrete set of gates that are universal for the purposes of quantum computation and can be fault-tolerantly performed on certain error-correcting codes. Each order of the hierarchy is defined as

$$C_l := \{U \mid UPU^\dagger \in C_{l-1},\ \forall P \in \mathcal{P}\}, \tag{1}$$

noting that $C_1 = \mathcal{P}$ is the set of Pauli gates, C_2 is the set of Clifford gates and C_3 includes, among others, the Pauli basis rotations by $\pi/4$ such as the T gate. Higher order gates typically correspond

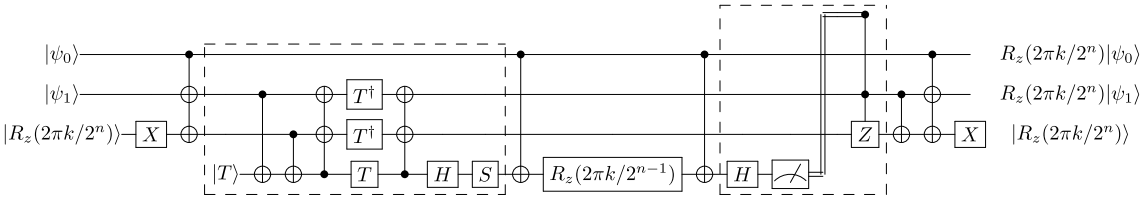


Figure 1: A Z-rotation catalyst circuit [22, 23]. The rotations $R_z(2\pi k/2^n)$ are elements of $\mathcal{T}_n$ (as shown in Eq. 2) where k is an odd integer and n is a natural number. The circuit utilises a $|\mathcal{T}_n\rangle$ state, a $|T\rangle$ state, three $\mathcal{T}_3$ gates and a $\mathcal{T}_{n-1}$ gate to perform two $\mathcal{T}_n$ gates on two separate qubits while retaining the original $|\mathcal{T}_n\rangle$ state. The output $\mathcal{T}_n$ gates can either be applied directly to target qubits or $|\psi_0\rangle$ and $|\psi_1\rangle$ states can be first set to $|+\rangle$ states, so that the application of the $\mathcal{T}_n$ gates prepare two $|\mathcal{T}_n\rangle$ states which can then be used to implement $\mathcal{T}_n$ gates at any time and on any target qubit using teleportation circuits. However, this consumes on average an additional half a $\mathcal{T}_{n-1}$ gate for the implementation of each $\mathcal{T}_n$ gate. The two sets of grouped gates (outlined by dashed lines) correspond to logical-AND computation and uncomputation circuits, which only requires a total T-count of four to implement [23]. The circuit can be recursively applied until the $R_z(2\pi k/2^{n-1})$ gate position reduces down to a $\mathcal{T}_3$ gate which has a cost of 1. All costs are calculated by assuming that all target gates at each recursive level of the circuit are used at some point (i.e. that no output gates are wasted).

to finer angle rotations.

In this work, we compare sets of single-qubit universal logical base gates consisting of Clifford gates and Z-rotation gates from higher orders of the Clifford hierarchy. Although only higher order Z-rotations are included, they can be readily converted to other gates in the same order of the Clifford hierarchy by multiplying gates from lower orders. In particular, by multiplying Clifford gates, other gates of the same order are generated for the same cost. For example $Z.R_z(\pi/4) = R_z(5\pi/4)$ and $H.R_z(\pi/4).H = R_x(\pi/4)$ up to global phase, where H is the Hadamard gate and Z is the Pauli-Z gate. These sets of logical base gates are compared with respect to the optimal resource costs resulting from gate synthesis for random target gates. Each set of Z-rotation gates from order $3 \leq l \leq 7$ of the Clifford hierarchy, denoted $\mathcal{T}_l$, can be written as

$$
\begin{aligned}
\mathcal{T}_3 &:= \left\{ R_z\left(\frac{\pi k}{4}\right) \in C_3 \;\middle|\; k \in \{-1, 1\} \right\}, \\
\mathcal{T}_4 &:= \left\{ R_z\left(\frac{\pi k}{8}\right) \in C_4 \;\middle|\; k \in \{-3, -1, 1, 3\} \right\}, \\
\mathcal{T}_5 &:= \left\{ R_z\left(\frac{\pi k}{16}\right) \in C_5 \;\middle|\; k \in \{-7, -5, \ldots, 5, 7\} \right\}, \\
\mathcal{T}_6 &:= \left\{ R_z\left(\frac{\pi k}{32}\right) \in C_6 \;\middle|\; k \in \{-15, -13, \ldots, 13, 15\} \right\}, \text{ and} \\
\mathcal{T}_7 &:= \left\{ R_z\left(\frac{\pi k}{64}\right) \in C_7 \;\middle|\; k \in \{-31, -29, \ldots, 29, 31\} \right\}.
\end{aligned} \tag{2}
$$

The five sets of logical base gates used in our analysis are then constructed as

$$
\begin{aligned}
\text{Set}_1 &:= C_1 \cup C_2 \cup \mathcal{T}_3, \\
\text{Set}_2 &:= \text{Set}_1 \cup \mathcal{T}_4, \\
\text{Set}_3 &:= \text{Set}_2 \cup \mathcal{T}_5, \\
\text{Set}_4 &:= \text{Set}_3 \cup \mathcal{T}_6, \text{ and} \\
\text{Set}_5 &:= \text{Set}_4 \cup \mathcal{T}_7.
\end{aligned} \tag{3}
$$

Calculating precise resource costs of implementing each gate fault-tolerantly is an extensive task that would need to consider a variety of factors such as qubit count, circuit depth, magic state distillation methods and details of the error-correction implementation. As an approximation for the cost of these logical gates we investigate two approaches of assigning costs to individual $\mathcal{T}_l$ gates, where gates from C_1 and C_2 are assumed to be free since they can be implemented in a relatively straightforward way. The first approach can associate the costs with the T-count, which

(a) Direct application of $\mathcal{T}_l$ method

Average T-count per base gate	
$\mathcal{T}_3$	1
$\mathcal{T}_4$	2.5
$\mathcal{T}_5$	3.25
$\mathcal{T}_6$	3.625
$\mathcal{T}_7$	3.8125

(b) Application of $\mathcal{T}_l$ via $|\mathcal{T}_l\rangle$ method

Average T-count per base gate	
$\mathcal{T}_3$	1
$\mathcal{T}_4$	3
$\mathcal{T}_5$	5
$\mathcal{T}_6$	7
$\mathcal{T}_7$	9

Table 1: The average number of T gates required to implement a single qubit Z-rotation gate from order l of the Clifford hierarchy $\mathcal{T}_l$ using the Z-rotation catalyst approach. **(a)** The average T-count required to implement $\mathcal{T}_l$ gates by directly applying them to target qubits. The T-counts are calculated using the expression $\text{Cost}[\mathcal{T}_l] = 4 - 3 \times 2^{3-l}$ as shown in Equation 5. **(b)** The average T-count required to implement $\mathcal{T}_l$ gates by applying them via intermediate $|\mathcal{T}_l\rangle$ states at every level of recursion (since the Z-rotation catalyst circuit is recursively applied). The T-counts are calculated using the expression $\text{Cost}[\mathcal{T}_l] = 1 + 2 \times (l - 3)$ as shown in Equation 7.

Average raw magic state count per base gate				
Base gate error rate μ	10^{-5}	10^{-10}	10^{-15}	10^{-20}
$\mathcal{T}_3$	5.1	36.2	70.4	120.1
$\mathcal{T}_4$	16.7	103.1	186.5	358.7
$\mathcal{T}_5$	34.8	172.7	333.2	635.8
$\mathcal{T}_6$	49.0	255.8	486.1	962.2
$\mathcal{T}_7$	64.7	344.8	671.5	1351.2

Table 2: The average raw magic state count required for distillation and implementation of corresponding logical base gates, obtained from the supplementary materials of [24]. Each column contains the cost of distilling and implementing a logical Z-rotation gate from order l of the Clifford hierarchy $\mathcal{T}_l$ to below a gate error rate μ calculated using the diamond norm. The raw magic state physical level error is assumed to be 0.1%.

is used as the standard metric for measuring the costs of gate sequences within the gate synthesis literature. This can be done by using a Z-rotation catalyst circuit shown in Fig. 1, which was introduced in [23] and presented in more detail in [22]. The circuit is similar to a synthillation parity-check circuit described in [25]. It utilises a $|\mathcal{T}_l\rangle$ state and a small number of T gates to perform two $\mathcal{T}_l$ gates on two different qubits while retaining the original $|\mathcal{T}_l\rangle$ state. Costs can be calculated by recursively applying this circuit, assuming that all output gates at each recursive level are resourced (i.e. that no output gates are wasted). We calculate the costs using the Z-rotation catalyst approach in two ways. The first assumes that output $\mathcal{T}_l$ gates are directly applied to target qubits. The recurrence relation for the T-counts using this method can be obtained as

$$\text{Cost}\left[\mathcal{T}_l\right] = \frac{4 + \text{Cost}\left[\mathcal{T}_{l-1}\right]}{2}, \tag{4}$$

where $\text{Cost}[\mathcal{T}_3] = 1$. Solving this results in the average number of T gates required to implement a $\mathcal{T}_l$ gate to be expressed as

$$\text{Cost}\left[\mathcal{T}_l\right] = 4 - 3 \times 2^{3-l}, \tag{5}$$

which is enumerated in Table 1a for $3 \leq l \leq 7$. The second method of calculating the T-count using the Z-rotation catalyst approach applies the $\mathcal{T}_l$ gates to $|+\rangle$ states, creating corresponding intermediate $|\mathcal{T}_l\rangle$ states, which are then consumed to implement the gates via teleportation circuits. The recurrence relation for these costs can be obtained as

$$\text{Cost}\left[\mathcal{T}_l\right] = 2 + \text{Cost}\left[\mathcal{T}_{l-1}\right], \tag{6}$$

where $\text{Cost}[\mathcal{T}_3] = 1$, resulting in the expression

$$\text{Cost}\left[\mathcal{T}_l\right] = 1 + 2 \times (l - 3) \tag{7}$$

which is enumerated in Table 1b for $3 \leq l \leq 7$. This second method is more expensive since the teleportation circuit that consumes the $|\mathcal{T}_l\rangle$ state to implement the $\mathcal{T}_l$ gate requires a $\mathcal{T}_{l-1}$ correction

gate to be applied 50% of the time. However, this method is more flexible in implementation since the outputted $|\mathcal{T}_l\rangle$ states can be used at any time to implement $\mathcal{T}_l$ gates onto any target qubits, enabling more options when instruction scheduling. A realistic employment of the Z-rotation catalyst approach would likely benefit from a combination of both direct application of $\mathcal{T}_l$ gates and application via their intermediate $|\mathcal{T}_l\rangle$ states. For the second approach of assigning resource costs, we use the average number of raw magic states to implement fault-tolerant $\mathcal{T}_l$ gates from direct magic state distillation procedures. Resource costs have already been calculated for Y-rotation gates $R_y(2\pi/2^l)$ from the Clifford hierarchy by searching for optimal combinations of various distillation protocols with respect to target gate synthesis error rates ϵ [24]. For integer multiples $R_y(2\pi k/2^l)$, the distillation protocols can be performed identically, hence they can be assigned the same cost. To follow convention, the Y-rotation gates are converted to Z-rotation gates with the same cost using the relation $R_z(\theta) = HS^\dagger R_y(\theta)SH$, since H and $S := R_z(\pi/2)$ have zero cost due to being elements of C_2. These resource costs vary between orders of the Clifford hierarchy and are shown in Table 2.

Sequence Generation Algorithm

In this section, a sequence generation algorithm, based on Dijkstra's algorithm, is developed that generates a database of all cost-optimal single-qubit gate sequences below some maximum cost using arbitrary sets of universal base gates which have individually assigned cost values. We use this algorithm to help study the average cost of cost-optimal gate synthesis when including Z-rotation gates from higher orders of the Clifford hierarchy as base gates. Due to the flexibility of this algorithm, it could be used as a subroutine within other synthesis algorithms. For example, it could be used as the base approximation step within the SK algorithm, enabling the SK algorithm to consider individual base gate costs when synthesising target gates.

The sequence generation algorithm explores the space of sequence configurations using a tree expansion as shown in Figure 2, where each node corresponds to a gate and each path from the root node to any other node corresponds to a sequence of gates. Let B_n be an element of $PSU(2)$ corresponding to the base gate of node n in the sequence tree. A *combined gate* S_n of node n is calculated by multiplying all nodes within the branch from the root down to n, i.e. $S_n := B_{n_0} \cdot B_{n_1} \dots B_{n_k}$, where n_i is the i^{th} node from the root node such that n_0 is the root and n_k is node n. The Lie algebra generator of S_n in the Pauli basis is of the form of a vector $\alpha_n X + \beta_n Y + \gamma_n Z$ with real coefficients and can be written as $(\alpha_n, \beta_n, \gamma_n)$. Each vector represents a point in a ball of radius $\pi/2$ over the Pauli bases X, Y and Z. Thus each point within the ball is a geometrical location corresponding to a single-qubit gate.

The pseudocode for the algorithm is shown in Algorithm 1. It works by expanding nodes in a sequence tree (see Figure 2). All leaf (end) nodes of the sequence tree are stored in a minimum heap data structure which sorts the leaf nodes based on their corresponding sequence cost in increasing order. This determines the order of nodes to expand. The tree begins as a single identity gate at the root node which is added as the first element to the leaf node heap. At each iteration, the leaf node with the lowest sequence cost, i, is taken from the heap, which for the first iteration would be the identity gate node. The vector $(\alpha_i, \beta_i, \gamma_i)$ is calculated from the combined gate of the corresponding node's sequence. Before expanding a node in the sequence tree, we check whether another node with the same combined gate vector has already been expanded, using a hashset data structure. If the vector exists in the hashset, then the node is removed from the sequence tree and the algorithm proceeds to the next iteration. This repeats until a unique vector is found. When such a vector is found, it is added to the hashset for uniqueness checking in further iterations and the corresponding node in the sequence tree is expanded by generating a child node for each base gate. Each of these child nodes are added to the leaf node heap. To save computation time, adding a child node to the sequence tree and the heap can be limited to when their corresponding vectors are unique. Since vectors of sequences with lower costs are always added to the hashset before those with higher costs, the hashset must only contain vectors corresponding to sequences with the lowest cost among all sequences that produce equivalent combined gates. Thus, whenever a vector is successfully added to the hashset, the corresponding sequence must be cost-optimal. The cost-optimal vector and sequence pair can be stored in a data structure such as a k-d tree which can be used to approximate target gates by geometrically searching for nearest neighbours in the

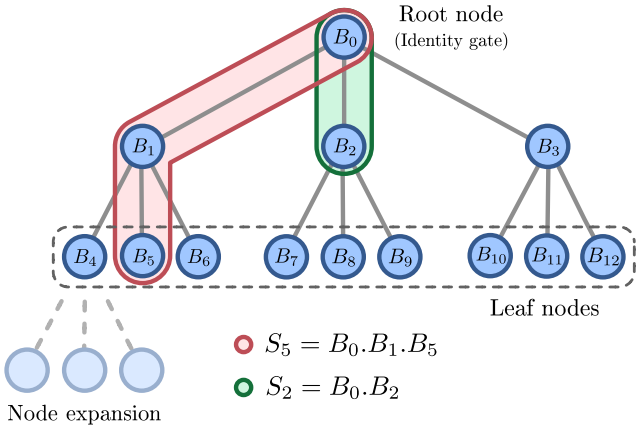
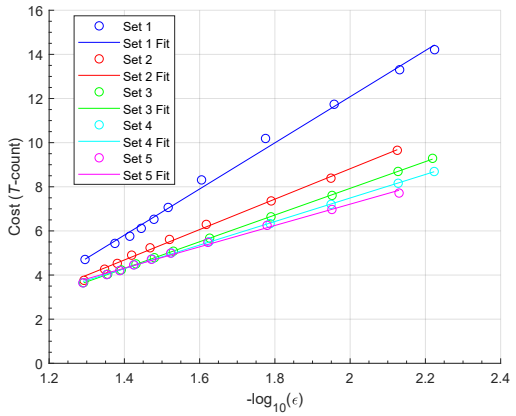


Figure 2: An example of a sequence tree used to relate logical base gates, gate sequences and combined gates for the sequence generation algorithm. A node n corresponds to a single-qubit base gate B_n and the root node corresponds to the identity gate $B_0 = I$. A gate sequence corresponding to n is the sequence of logical base gates along the path from B_0 to B_n. A combined gate S_n is calculated by multiplying all logical base gates within the gate sequence in sequence order. In this example, B_1, B_2 and B_3 are logical base gates where $B_1 = B_4 = B_7 = B_{10}$, $B_2 = B_5 = B_8 = B_{11}$ and $B_3 = B_6 = B_9 = B_{12}$. In the sequence generation algorithm, the leaf node with the lowest sequence cost is expanded by adding a child node as a new leaf node for each gate in the set of logical base gates. All non-leaf nodes of the tree correspond to cost-optimal sequences and they can be thought of as the cost-optimal sequence database generated by the algorithm. Although all leaf nodes are depicted to be at the same depth in the tree, this is not always the case. At any point during the sequence generation algorithm, a path of relatively expensive logical base gates may be much shorter than a path of relatively cheap gates.

Algorithm 1 Cost-optimal sequence generation

```
1: procedure GenerateSequences(baseGates, maxCost)
2:     sequenceDatabase ← new KdTree⟨Node⟩          ▷ To store the cost-optimal sequences geometrically
3:     sequenceTree ← new Tree⟨Node⟩                ▷ To relate nodes, sequences and combined gates
4:     sequenceTree.SetRoot(Identity gate)          ▷ Set the root node to the identity gate
5:     sortedLeafNodes ← new MinHeap⟨Node⟩          ▷ To order sequence tree leaf nodes based on sequence cost
6:     uniqueVectors ← new Hashset⟨Vector3⟩         ▷ To test whether sequences have the same combined gates
7:     Add sequenceTree.root to sortedLeafNodes
8:     while sortedLeafNodes not empty do
9:         i ← sortedLeafNodes.Pop()                ▷ Obtains and removes the leaf node with lowest sequence cost
10:        if sequenceTree.SequenceCost(i) > maxCost then
11:            return sequenceDatabase              ▷ Complete! Ignore i and return cost-optimal sequences
12:        end if
13:        (α_i, β_i, γ_i) ← sequenceTree.GetVector(i)
14:        if (α_i, β_i, γ_i) not in uniqueVectors then
15:            Add i to sequenceDatabase            ▷ The node i corresponds to a cost-optimal sequence
16:            Add (α_i, β_i, γ_i) to uniqueVectors
17:            childNodes ← sequenceTree.GenerateChildren(i, baseGates)
                                                    ▷ Add base gates as child nodes of i
18:            for all j in childNodes do
19:                (α_j, β_j, γ_j) ← sequenceTree.GetVector(j)
20:                if (α_j, β_j, γ_j) not in uniqueVectors then
21:                    Add j to sortedLeafNodes
22:                else
23:                    Remove j from sequenceTree   ▷ Vector corresponding to childNode j already found
24:                end if
25:            end for
26:        end if
27:    end while
28: end procedure
```

(a) Sequences using the Z-rotation catalyst approach with directly applied output gates



(b) Sequences using the Z-rotation catalyst approach with output gates applied via intermediate magic states

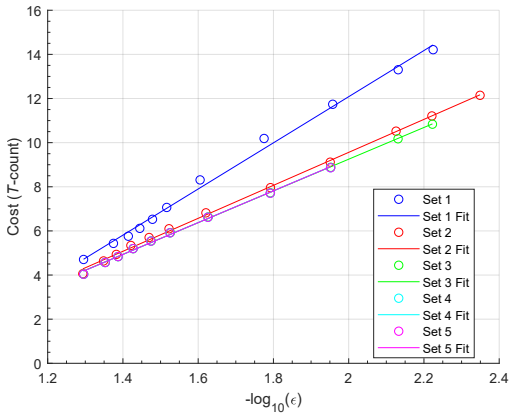


Figure 3: Cost-optimal sequence T-counts calculated using the Z-rotation catalyst approach plotted against synthesised target gate error rates of ϵ. Each point is the result of averaging the T-count for implementing 5000 random target gate. The synthesis logical errors ϵ are calculated using the trace distance (shown in Equation 8). The logical base gates for each set of base gates are specified in Equation 3. The corresponding linear best fit values for both plots are shown in Table 3. **(a)** A plot of sequence costs where base gates are assigned costs by assuming that all output gates are directly applied to target qubits. Base gate costs are calculated using Eq. 5 and enumerated in Table 1a. The reductions in scaling factors relative to Set_1 are $34 \pm 4\%$, $43 \pm 2\%$, $49 \pm 2\%$, and $54 \pm 3\%$ for Set_2, Set_3, Set_4, and Set_5 respectively, where uncertainties correspond to 95% confidence intervals. These correspond to synthesis cost savings in the limit of small ϵ. **(b)** A plot of sequence costs where base gates are assigned costs by assuming that output gates are applied to $|+\rangle$ states to form the corresponding intermediate magic states, gates are then applied by consuming the magic states via teleportation circuits. Base gate costs are calculated using Eq. 7 and enumerated in Table 1b. The reductions in scaling factors relative to Set_1 are $29 \pm 3\%$, $31 \pm 3\%$, $31 \pm 4\%$, and $31 \pm 4\%$ for Set_2, Set_3, Set_4, and Set_5 respectively, where uncertainties correspond to 95% confidence intervals. These correspond to synthesis cost savings in the limit of small ϵ.

space of vectors.

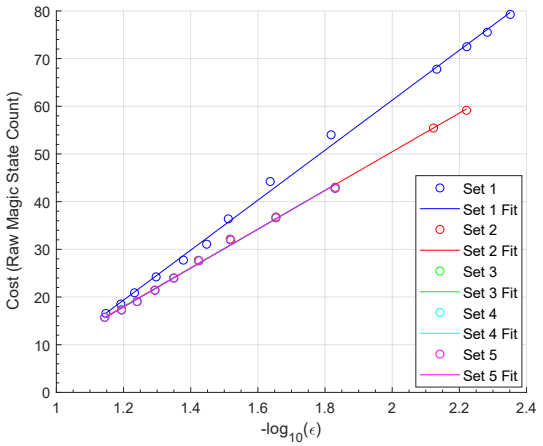
There is a notable further optimisation that could be implemented into Algorithm 1. During the procedure, all non-leaf nodes within the sequence tree correspond to cost-optimal sequences with unique combined gate vectors, that is, each path starting at the root node and ending at any non-leaf node is a shortest path to the sequence's unique combined gate. To see how this could be helpful, first assume that an existing sequence tree needs to grow to a new maximum cost, such that the leaf nodes need to expand multiple times along the same branch. Instead of searching through every combination of base gates as children for a leaf node, the sequence tree itself can be used as a sieve by iterating child nodes from the root that are known to be shortest paths. The tree already contains optimal paths up to a certain depth, so this information could be used to help avoid the tree branches expanding in directions that produce nonoptimal paths to unique combined gates.

In Algorithm 1, cost-optimal sequences and their corresponding vectors are stored in a k-d tree which uses the Euclidean distance on the vectors to organise the data. Due to the periodic nature of the vectors, there is a small chance of failure in the k-d tree when searching for nearest neighbours to points close to the boundary. With computational overhead, the k-d tree may be modified to help overcome this [26], or a more appropriate data structure such as a vantage point tree [27, 28] may be used instead. In general, further alternative data structures may be used such as the geometric nearest-neighbour access tree [29].

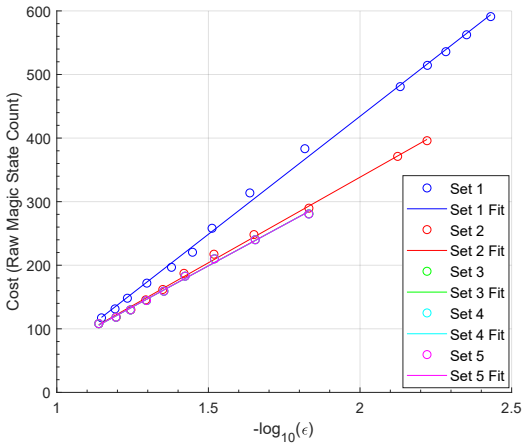
Synthesis Results

Algorithm 1 was computed using the sets of logical base gates described in Eq. 3 with the assignment of costs obtained from the two approaches of implementing base gates, where values are shown in

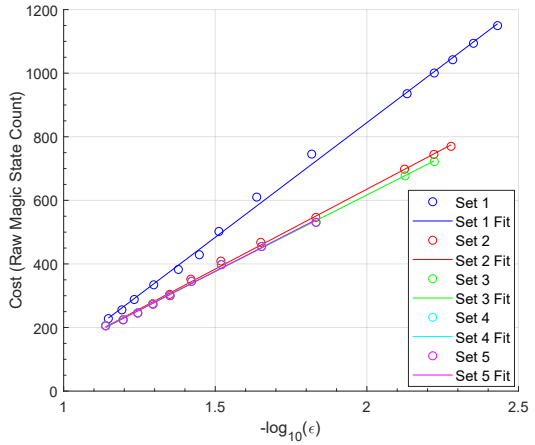
(a) Sequences with below $\mu = 10^{-5}$ logical base gate error



(b) Sequences with below $\mu = 10^{-10}$ logical base gate error



(c) Sequences with below $\mu = 10^{-15}$ logical base gate error



(d) Sequences with below $\mu = 10^{-20}$ logical base gate error

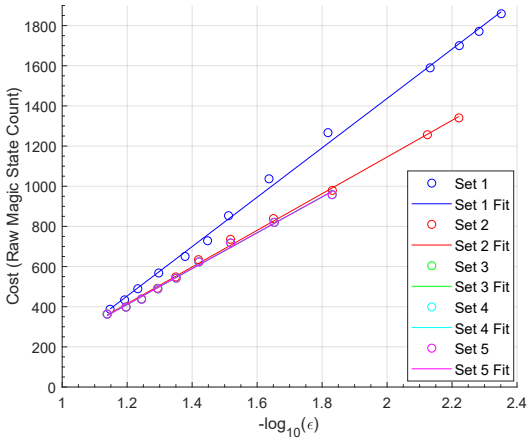


Figure 4: Cost-optimal sequence costs averaged over 5000 random target gates with respect to target gate synthesis logical error rates ϵ. The logical base gates used are specified in Eq. 3 with cost values (shown in Table 2) assigned as the average number of raw magic states required to distil and implement them to below a specified logical gate error. The synthesis logical errors ϵ are calculated using the trace distance (shown in Equation 8). Corresponding linear best fit values are shown in Table 4. The pattern of the data about the lines of best fit for each logical base gate set are similar between plots because for each of the logical base gate errors, the ratios of the base gate cost values between orders of the Clifford hierarchy are similar, hence the cost optimal sequences will be comparable. **(a)** Synthesis using logical base gate costs associated with $\mu = 10^{-5}$ logical gate error. **(b)** Synthesis using logical base gate costs associated with $\mu = 10^{-10}$ logical gate error. **(c)** Synthesis using logical base gate costs associated with $\mu = 10^{-15}$ logical gate error. **(d)** Synthesis using logical base gate costs associated with $\mu = 10^{-20}$ logical gate error.

Tables 1 and 2. A database was generated that is in the form of a k-d tree of cost-optimal sequences up to some chosen maximum sequence cost. The sequences were organised in the k-d tree with respect to the vectors corresponding to their combined gates. For a given target gate G, gate synthesis was performed by searching for the lowest cost sequence among all nearest neighbours of G up to a chosen synthesis error (distance), ϵ, between their combined gates and G. The errors were computed using the trace distance defined as

$$\text{dist}(S, G) = \sqrt{(2 - |\text{tr}(S^\dagger G)|)/2}, \tag{8}$$

where S is a combined gate and G is the target gate. If such a sequence did not exist, then the database was further generated to a higher cost and the process was repeated until a sequence was found. Incrementally generating the cost-optimal sequence database in this manner helps avoid over generation.

For each set of base gates with individual costs calculated for each approach of implementing them, gate synthesis was performed on 5000 random target gates sampled from a uniform distribution for a variety of synthesis error rates ϵ (calculated using Eq. 8 with respect to the sequences' combined gates). Cost-optimal sequence T-counts calculated using the Z-rotation catalyst circuit approach for the two methods of assigning base gate costs are plotted against synthesised target gate error rates for each set of base gates in Figure 3. The corresponding linear best fit values for each set of logical base gates and corresponding cost values are shown in Table 3. We can compare the scaling factors of the fits between different sets of logical base gates to estimate changes in average sequence costs as the synthesis error ϵ approaches zero. For the Z-rotation catalyst circuit method that assumes all output gates are directly applied to target qubits (as opposed to using intermediate magic states), we find cost savings relative to Set_1 of $34 \pm 3\%$, $42 \pm 2\%$, $49 \pm 2\%$, and $54 \pm 3\%$ for Set_2, Set_3, Set_4, and Set_5 respectively, where uncertainties correspond to 95% confidence intervals. Data for a Set_6 that includes $\mathcal{T}_8$ gates was also calculated, however no noticeable improvement was found with sequence cost values being almost identical to Set_5 resulting in a cost saving of $52 \pm 3\%$ relative to Set_1. For the Z-rotation catalyst circuit method that assumes all output gates are applied to $|+\rangle$ states forming intermediate magic states before consuming them to perform the corresponding Z-rotation gate, we find cost savings relative to Set_1 of $29 \pm 3\%$, $31 \pm 3\%$, $31 \pm 4\%$, and $31 \pm 4\%$ for Set_2, Set_3, Set_4, and Set_5 respectively. These results show that if gate synthesis includes higher order Clifford hierarchy Z-rotation gates as base gates implemented using the Z-rotation catalyst approach, then a T-count saving of over 50% could potentially be achieved. Cost-optimal sequence raw magic state counts calculated using direct base gate distillation and implementation procedures are plotted against synthesised target error rates for each combination of base gates and cost values in Figure 4. Each of the four plots correspond to different resource costs of distilling and implementing the logical base gates with corresponding logical errors $\mu = 10^{-5}$, 10^{-10}, 10^{-15} and 10^{-20} calculated using the diamond norm. The corresponding linear best fit values for each set of logical base gates are shown in Table 4 and corresponding cost values are shown in Table 2 (physical error rate assumed to be 0.1% in all calculations). The pattern of the data about their lines of best fit for each base gate set are similar between plots. This is because for each of the logical base gate errors, the ratios of the logical base gate cost values between orders of the Clifford hierarchy are similar, hence the cost optimal sequences will be comparable. For logical base gate errors $\mu = 10^{-5}$, 10^{-10}, 10^{-15} and 10^{-20}, we find that Set_2 provides $23 \pm 3\%$, $27 \pm 2\%$, $30 \pm 2\%$ and $26 \pm 3\%$ reductions in scaling factor respectively compared to Set_1. For $\mu = 10^{-10}$ and 10^{-15}, we find that Set_3 provides $30 \pm 3\%$ and $33 \pm 2\%$ reductions in scaling factor respectively compared to Set_1, which are both approximately a further 3% savings compared to Set_2. No further improvements are noticeable in our data for these assignments of cost values. These results show that for any error-correction scheme with distillation costs assigned according to Table 2, using Set_2 (which includes $\mathcal{T}_4$ as logical base gates) instead of the standard Set_1, reduces the average resource cost scaling factor with respect to the synthesis negative log-error, $\log(\epsilon^{-1})$, by up to 30%. Additionally Set_3 can provide up to a further 3% reduction when compared to Set_2. Each method of assigning individual base gate costs that were used in this work indicated that the resource requirements of synthesis algorithms may be considerably improved by including higher orders of the Clifford hierarchy as logical base gates and by optimising with respect to the individual costs of implementing them.

(a) Linear fits using Z-rotation catalyst method

Base Gates	Scaling Factor	Constant
Set_1	10.46 ± 0.43	-8.83 ± 0.73
Set_2	6.89 ± 0.22	-4.96 ± 0.36
Set_3	6.05 ± 0.03	-4.17 ± 0.06
Set_4	5.33 ± 0.06	-3.18 ± 0.11
Set_5	4.84 ± 0.21	-2.46 ± 0.34

(b) Linear fits using Z-rotation catalyst method via magic states

Base Gates	Scaling Factor	Constant
Set_1	10.46 ± 0.43	-8.83 ± 0.73
Set_2	7.47 ± 0.15	-5.39 ± 0.26
Set_3	7.19 ± 0.12	-5.13 ± 0.21
Set_4	7.21 ± 0.25	-5.16 ± 0.38
Set_5	7.21 ± 0.25	-5.15 ± 0.39

Table 3: Linear best fits with a confidence level of 95% for cost-optimal sequence costs averaged over random target logical gates with respect to the negative log-error, $\log(\epsilon^{-1})$, for target gate synthesis calculated using the trace distance (shown in Equation 8). The sequences are constructed using logical base gates with cost values assigned according to Table 1. The corresponding plots are shown in Figure 3.

(a) Linear fits for Figure 4a for below $\mu = 10^{-5}$ logical base gate error

Base Gates	Scaling Factor	Constant
Set_1	52.4 ± 1.3	-43.5 ± 2.2
Set_2	40.6 ± 0.9	-30.7 ± 1.5
Set_3	40.8 ± 1.8	-31.0 ± 2.6
Set_4	40.8 ± 1.8	-31.0 ± 2.6
Set_5	40.8 ± 1.8	-31.0 ± 2.6

(b) Linear fits for Figure 4b for below $\mu = 10^{-10}$ logical base gate error

Base Gates	Scaling Factor	Constant
Set_1	371 ± 8	-308 ± 14
Set_2	269 ± 7	-200 ± 11
Set_3	258 ± 11	-189 ± 15
Set_4	258 ± 11	-188 ± 15
Set_5	258 ± 11	-188 ± 15

(c) Linear fits for Figure 4c for below $\mu = 10^{-15}$ logical base gate error

Base Gates	Scaling Factor	Constant
Set_1	722 ± 15	-599 ± 27
Set_2	503 ± 11	-370 ± 17
Set_3	482 ± 10	-347 ± 16
Set_4	488 ± 21	-355 ± 30
Set_5	488 ± 21	-355 ± 30

(d) Linear fits for Figure 4d for below $\mu = 10^{-20}$ logical base gate error

Base Gates	Scaling Factor	Constant
Set_1	1230 ± 30	-1020 ± 50
Set_2	913 ± 24	-680 ± 39
Set_3	893 ± 41	-661 ± 59
Set_4	893 ± 41	-661 ± 59
Set_5	893 ± 41	-661 ± 59

Table 4: Linear best fits with a confidence level of 95% for cost-optimal sequence costs averaged over random target logical gates with respect to the negative log-error, $\log(\epsilon^{-1})$, for target gate synthesis calculated using the trace distance (shown in Equation 8). The sequences are constructed using logical base gates with cost values assigned according to Table 2. The corresponding plots are shown in Figure 4.

Modelling Gate Proportions

For the raw magic state approach of implementing base gates and the Z-rotation catalyst circuit method that uses intermediate magic states, the logical base gate sets Set_3, Set_4 and Set_5 (see Eq. 3) were shown to provide only marginal resource savings for gate synthesis when compared with Set_2 (see Figs. 3b and 4), even though the sets contain many more logical base gates. To investigate this behaviour we develop a model in Appendix A for determining the proportion of sets of gates among all $\mathcal{T}_l$ gates where $l \geq 3$ within cost-optimal sequences approximating random target gates with specified gate costs. The proportions can provide insight into how the average sequence cost changes with respect to which $\mathcal{T}_l$ base gates are included as logical base gates and what cost values are assigned. For logical base gates with non-zero proportion within sequences approximating target gates, we expect that by increasing their cost, their recalculated proportion will decrease and the average cost of these sequences will increase. Furthermore, for sets of logical base gates with relatively small proportions, the average sequence cost would only slightly increase if the set were to be excluded compared to sets of base gates with larger proportions.

The model estimates the average proportion, p_n, of $\mathcal{T}_n$ logical base gates among all $\mathcal{T}_l$ gates where $l \geq 3$ from within cost-optimal sequences approximating random target gates to within sufficiently small synthesis errors ϵ. The construction is based on a unique canonical form [16] for sequences of logical base gates and is defined as

$$c.t_1.H.t_2.H \ldots t_N.c', \tag{9}$$

where c and c' are Clifford gates, H is the Hadamard gate, t_m is the m^{th} positioned Z-rotation gate from order three and above of the Clifford hierarchy, and M is the total number of t_m gates in the sequence. This canonical form has the property that arbitrary gate sequences with distinct combined gates, where the sequences can consist of logical base gates from the Clifford gates and Z-rotations from orders three and above of the Clifford hierarchy, can be reduced to distinct sequences of this form. The gate proportion for $\mathcal{T}_n$, denoted p_n, can be calculated by averaging the $\mathcal{T}_n$ logical gate count over all possible sequences in this canonical form that are below a chosen maximum cost C (as detailed in Appendix A). That is,

$$p_n = \frac{\sum_{k_3=0}^{\lfloor C/c_3 \rfloor} \sum_{k_4=0}^{\lfloor (C-c_3k_3)/c_4 \rfloor} \cdots \sum_{k_L=0}^{\lfloor (C-\sum_{j=3}^{L-1} c_j k_j)/c_L \rfloor} k_n \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}}{\sum_{k_3=0}^{\lfloor C/c_3 \rfloor} \sum_{k_4=0}^{\lfloor (C-c_3k_3)/c_4 \rfloor} \cdots \sum_{k_L=0}^{\lfloor (C-\sum_{j=3}^{L-1} c_j k_j)/c_L \rfloor} \sum_{t=3}^{L} k_t \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}}, \tag{10}$$

where c_j is the logical base gate implementation cost for $\mathcal{T}_j$, k_l is the number of $\mathcal{T}_l$ gates within a particular sequence, $|\mathcal{T}_l|$ is the number of gates within $\mathcal{T}_l$, and L is the order of the Clifford hierarchy to include Z-rotation gates up to.

This calculation outputs values closely matching proportion results obtained using the sequence generation algorithm for random target gates, as shown in Figure 5. Figure 5a shows the summed proportions of all $\mathcal{T}_4$ gates among $\mathcal{T}_3 \cup \mathcal{T}_4$ gates over a variety of $\mathcal{T}_4$ cost values for sequences consisting of Set_2 logical base gates. Figure 5b shows the summed proportions of all $\mathcal{T}_5$ gates among $\mathcal{T}_3 \cup \mathcal{T}_4 \cup \mathcal{T}_5$ gates over a variety of $\mathcal{T}_5$ cost values for sequences consisting of Set_3 logical base gates. The other logical base gate costs are assigned values according to their distillation and implementation cost with a maximum logical base gate error of $\mu = 10^{-15}$ as shown in Table 2. These results suggest that increasing the logical base gate implementation cost of a set $\mathcal{T}_n$ drastically lowers the proportion of them found within the database of cost-optimal sequences. Thus they become less effective at reducing the average cost-optimal sequence costs since they are included within sequences less often. This is a simpler calculation compared to actually performing gate synthesis for many random target gates. The gate set proportions appears to give an indication for how useful the gate subset is among the rest of the base gates. We suspect there is potential that with some further research it could be used to help provide a quick approximation for how much the average synthesis cost reduces when including a base gate subset with specified cost values.

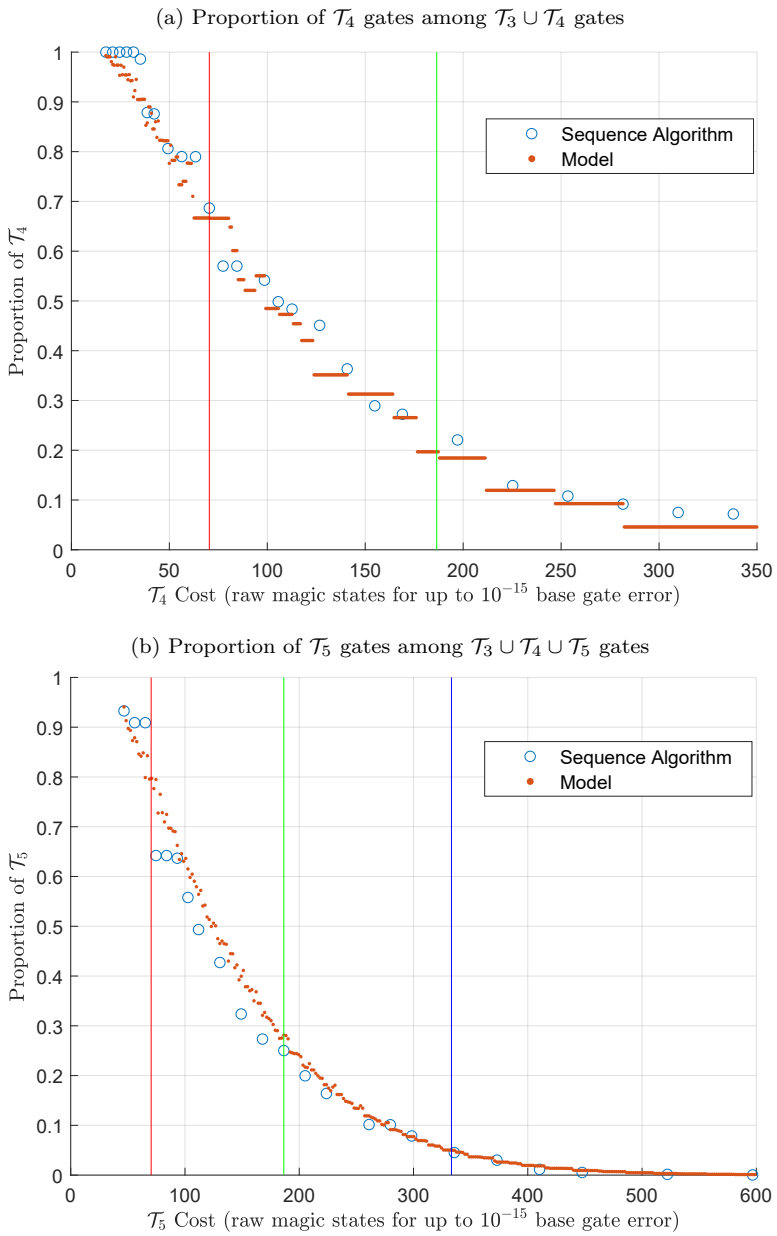


Figure 5: This figure shows the summed proportions of logical base gates from sequences resulting from the sequence generation algorithm and the proportions calculated using our model. The sequence generation algorithm outputs cost-optimal sequences approximating random target gates to within $\epsilon = 0.03$ synthesis logical gate error under the trace distance (see Eq. 8), while the model outputs the proportion of a set of logical base gates within the space of all cost-optimal sequences below a maximum cost that produce distinct combined gates. Clifford gates are ignored in the calculations since they are assumed to have zero cost. Both plots show that the model data closely fit the corresponding results from the sequence generation algorithm. The data show that increasing the logical base gate distillation and implementation cost of a particular set $\mathcal{T}_n$ drastically lowers the proportion of them found within the generated cost-optimal sequences. Thus the set $\mathcal{T}_n$ with increased costs becomes less effective at reducing the average cost-optimal sequence costs, since they are found less frequently within the sequences. Logical base gate costs are assigned according to Table 2 with a logical base gate error of $\mu = 10^{-15}$ calculated using the diamond norm. The red, green and blue vertical lines (ordered left to right) indicate the logical base gate distillation and implementation costs for $\mathcal{T}_3$, $\mathcal{T}_4$ and $\mathcal{T}_5$ respectively. **(a)** The summed proportions of $\mathcal{T}_4$ logical base gates among $\mathcal{T}_3 \cup \mathcal{T}_4$ gates for cost-optimal sequences consisting of Set_2 logical base gates. Logical base gates from $\mathcal{T}_3$ are fixed while the cost for $\mathcal{T}_4$ gates vary. **(b)** The summed proportions of $\mathcal{T}_5$ logical base gates among $\mathcal{T}_3 \cup \mathcal{T}_4 \cup \mathcal{T}_5$ gates for cost-optimal sequences consisting of Set_3 logical base gates. Logical base gates from $\mathcal{T}_3 \cup \mathcal{T}_4$ are fixed while the cost for $\mathcal{T}_5$ gates vary.

Discussion

We investigated the cost of sequences produced by cost-optimal single-qubit gate synthesis using logical base gates from a combination of Clifford gates and Z-rotation gates from higher orders of the Clifford hierarchy. An algorithm, based on Dijkstra's algorithm, was used to generate a database of cost-optimal sequences from arbitrary single-qubit universal sets of logical base gates with individually assigned costs. As base gates, combinations of Clifford gates and Z-rotation gates from various orders of the Clifford hierarchy were used with two approaches of implementing them. The first uses a recursively applied Z-rotation catalyst circuit that utilises a temporary ancilla qubit, a small number of T gates and a Z-rotation state to apply two Z-rotation gates of the same angle on two separate qubits while retaining the original Z-rotation state. We calculate average T-count costs for this approach using the following two methods: all output gates of the catalyst circuits are applied directly to target qubits; and each output gate is first applied to a $|+\rangle$ state to form an intermediate magic state, which is then consumed to implement the corresponding gate via a teleportation circuit. The second approach of implementing base gates is through magic state distillation and implementation circuits that can assign costs as the average number of raw magic states used to implement them in error-correction codes up to specified logical error rates. After assigning base gate costs using each method, gate synthesis was performed by finding nearest neighbours within the database of cost-optimal sequences in the Pauli vector space corresponding to combined gates of sequences.

Using the Z-rotation catalyst approach with directly applied output gates to assign gate costs, we found that by including the higher order Clifford hierarchy Z-rotation gates along with the standard Clifford+T set of base gates, there was a reduction in synthesis cost when compared to only using the Clifford+T base gate set. The average cost-optimal sequence T-counts reduced by $34 \pm 3\%$, $42 \pm 2\%$, $49 \pm 2\%$, and $54 \pm 3\%$ for the accumulative inclusion of the fourth, fifth, sixth, and seventh orders respectively. When using the same approach but with all output gates being applied via intermediate magic states, the average cost-optimal sequence T-counts reduced by $29 \pm 3\%$, $31 \pm 3\%$, $31 \pm 4\%$, and $31 \pm 4\%$ for the accumulative inclusion of the fourth, fifth, sixth, and seventh orders respectively. Each average T-count calculated using the catalyst circuit approach assumes that every output gate of all recursive levels of the circuit are resourced such that no output gates are wasted. The procedure also assumes that there are sufficient numbers of ancilla qubits and Z-rotation catalyst states for smooth implementation of the gate sequences resulting from synthesis. A realistic employment of the approach would likely use a combination of direct application of output gates and the use of intermediate magic states. This is because direct application is cheaper with respect to T-count, however the intermediate magic states help make the implementation more flexible since they can be consumed at any time to implement the corresponding gate onto any target qubit. Nevertheless, these results show that there is potential for the average T-count to decrease by over 50% when performing gate synthesis with higher order Clifford hierarchy Z-rotation base gates that are implemented using this approach, when compared to cost-optimal synthesis using only the Clifford+T base gate set.

By instead using the magic state distillation approach with base gate costs assigned as the number of raw magic states, we found that including the fourth order Z-rotation gates from the Clifford hierarchy along with the standard Clifford+T gate set decreased the average cost-optimal sequence costs by up to $30 \pm 2\%$. We observe a reduction of up to $33 \pm 2\%$ when additionally including the Z-rotation gates from the fifth order. No noticeable improvement is observed when additionally including higher order Z-rotation base gates up to the seventh order. Although these savings are not quite as large as what may be possible with the Z-rotation catalyst approach, the magic state distillation approach does not require an accessible collection of Z-rotation catalyst states to be stored throughout the computation. The implementation circuit for the distilled Z-rotation magic state does require the application of a double angled Z-rotation gate as a correction 50% of the time. However, this correction gate can ideally be generated as it is required, so that every possible angled rotation does not need to be stored in advance. Also, the number of raw magic states is only a rough approximation for the actual resource costs of implementation. A precise calculation would be an extensive task that considers a variety of factors such as qubits count, circuit depth, magic state distillation cost and details of the error-correcting implementation.

We investigated the lack of further improvement found when including Z-rotation gates from

higher than the fourth order of the Clifford hierarchy when using the direct magic state distillation approach and the Z-rotation catalyst circuit approach with output gates being applied via intermediate magic states. A model was developed that estimates the proportion of logical base gates within sequences approximating random target gates. This model assumes that each Z-rotation gate from orders three and above of the Clifford hierarchy have equal proportions when assigned equal cost values, that is, the gate operations have equal usefulness for approximating random target gates for the purposes of gate synthesis. The proportion estimations were shown to closely fit the data obtained using the sequence generation algorithm on random target gates. This suggests that the lack of observed cost reduction when using higher order logical base gates is due to there being far less numbers of them at their assigned costs within all cost-optimal sequences generated up to the chosen maximum sequence cost. Thus the frequency of the base gates being used for synthesis of random target gates is low, leading to a low level of influence over the average resource costs overall. The model provides a simple method, without needing to generate the full database of sequences, for estimating these gate proportions with each order of the Clifford hierarchy being assigned individual cost values.

Acknowledgements

This work was supported by the University of Melbourne through the establishment of an IBM Q Network Hub at the University. CDH is supported by a research grant from the Laby Foundation. We would like to thank Earl Campbell and Kae Nemoto for valuable discussions. We would also like to thank the referee for suggesting the Z-rotation catalyst circuit approach for estimating gate costs.

References

[1] Sergey B Bravyi and A Yu Kitaev. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052*, 1998.

[2] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002. DOI: 10.1063/1.1499754.

[3] Robert Raussendorf, Jim Harrington, and Kovid Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9(6):199, 2007. DOI: 10.1088/1367-2630/9/6/199.

[4] David S Wang, Austin G Fowler, and Lloyd CL Hollenberg. Surface code quantum computing with error rates over 1%. *Physical Review A*, 83(2):020302, 2011. DOI: 10.1103/PhysRevA.83.020302.

[5] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*, 102(11):110502, 2009. DOI: 10.1103/PhysRevLett.102.110502.

[6] Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. Methodology for quantum logic gate construction. *Physical Review A*, 62(5):052316, 2000. DOI: 10.1103/PhysRevA.62.052316.

[7] Michael A. Nielsen and Isaac L. Chuang. *The Solovay–Kitaev theorem*, page 617–624. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.019.

[8] A Yu Kitaev, AH Shen, and MN Vyalyi. Classical and Quantum Computation (Graduate Studies in Mathematics vol 47)(Providence, RI: American Mathematical Society). 2002. DOI: 10.1090/GSM/047.

[9] Austin G Fowler. Constructing arbitrary Steane code single logical qubit fault-tolerant gates. *Quantum Information & Computation*, 11(9-10):867–873, 2011.

[10] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006. ISSN 1533-7146.

[11] Ken Matsumoto and Kazuyuki Amano. Representation of quantum circuits with Clifford and $\pi/8$ gates. *arXiv preprint arXiv:0806.3834*, 2008.

[12] Alex Bocharov and Krysta M Svore. Resource-optimal single-qubit quantum circuits. *Physical Review Letters*, 109(19):190501, 2012. DOI: 10.1103/PhysRevLett.109.190501.

[13] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates. *Quantum Information & Computation*, 13(7–8):607–630, 2013. ISSN 1533-7146.
[14] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits. *IEEE Transactions on Computers*, 65(1):161–172, 2016. DOI: 10.1109/TC.2015.2409842.
[15] Neil J Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations. *Quantum Information & Computation*, 16(11-12):901–953, 2016.
[16] Simon Forest, David Gosset, Vadym Kliuchnikov, and David McKinnon. Exact synthesis of single-qubit unitaries over Clifford-cyclotomic gate sets. *Journal of Mathematical Physics*, 56 (8):082201, 2015. DOI: 10.1063/1.4927100.
[17] Vadym Kliuchnikov, Alex Bocharov, Martin Roetteler, and Jon Yard. A framework for approximating qubit unitaries. *arXiv preprint arXiv:1510.03888*, 2015.
[18] Peter Selinger. Efficient Clifford+T approximation of single-qubit operators. *Quantum Information & Computation*, 15(1–2):159–180, 2015. ISSN 1533-7146.
[19] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits. *Physical Review Letters*, 110(19):190502, 2013. DOI: 10.1103/PhysRevLett.110.190502.
[20] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Efficient synthesis of probabilistic quantum circuits with fallback. *Physical Review A*, 91(5):052317, 2015. DOI: 10.1103/PhysRevA.91.052317.
[21] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390, 1999. DOI: 10.1038/46503.
[22] Craig Gidney and Austin G Fowler. Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation. *Quantum*, 3:135, 2019. DOI: 10.22331/Q-2019-04-30-135.
[23] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, 2018. DOI: 10.22331/Q-2018-06-18-74.
[24] Earl T Campbell and Joe O'Gorman. An efficient magic state approach to small angle rotations. *Quantum Science and Technology*, 1(1):015007, 2016. DOI: 10.1088/2058-9565/1/1/015007.
[25] Earl T Campbell and Mark Howard. Magic state parity-checker with pre-distilled components. *Quantum*, 2:56, 2018. DOI: 10.22331/Q-2018-03-14-56.
[26] Joshua M Brown, Terry Bossomaier, and Lionel Barnett. Review of data structures for computationally efficient nearest-neighbour entropy estimators for large systems with periodic boundary conditions. *Journal of Computational Science*, 23:109–117, 2017. DOI: 10.1016/J.JOCS.2017.10.019.
[27] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991. DOI: 10.1016/0020-0190(91)90074-r.
[28] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Soda*, volume 93, pages 311–21, 1993.
[29] Tien Trung Pham, Rodney Van Meter, and Clare Horsman. Optimization of the Solovay-Kitaev algorithm. *Physical Review A*, 87(5):052332, 2013. DOI: 10.1103/PhysRevA.87.052332.

Appendix

A Model for Gate Proportions

Here we develop the theory for estimating the average proportion p_n of logical base gates among all $\mathcal{T}_l$ gates (where $l \geq 3$) with specified costs within cost-optimal sequences approximating random target gates synthesised to within an error threshold of ϵ. We begin by assuming that each logical base gate in $\mathcal{T}_3 \cup \mathcal{T}_4 \ldots \mathcal{T}_L$ for $L \geq 3$ has equal proportions if they were to have equal costs, that is, the gate operations are equally effective for the purposes of gate synthesis. This can be justified

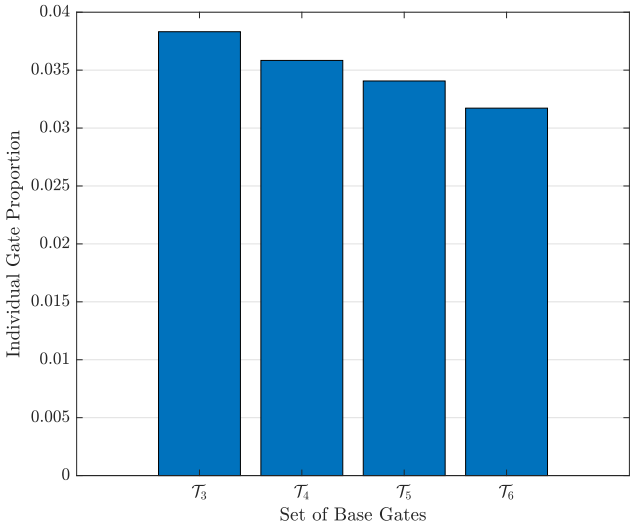


Figure A.1: The proportions of individual logical base gates with equally assigned costs (synthesis logical level error 0.03 using the trace distance). The number of gates within each set doubles for increasing order where $\mathcal{T}_3$ contains two gates (see Equation 2). This plot indicates that the logical base gates are almost equivalently useful in approximating random target gates using cost-optimal gate synthesis.

by the data in Figure A.1. The figure shows that when each logical base gate is given equal costs, the sequence generation algorithm generates a database of gate sequences with each gate having approximately the same proportions, where the proportions slowly decrease for increasing order. We do not expect these proportions to significantly change for larger sequence costs (or smaller synthesis error thresholds ϵ) since the logical base gate proportions are approximately constant for sufficiently large maximum sequence costs. This can be seen in Fig. A.2 for the case of $\mathcal{T}_5$ logical base gates from within Set_3 generated by the sequence generation algorithm for random target gates.

Assume we have a database of cost-optimal gate sequences that have been generated up to a chosen maximum cost with individually assigned implementation costs for each set of logical base gates $\mathcal{T}_l$ where $l \geq 3$. We will calculate the proportion of $\mathcal{T}_n$ gates among all sequences within the database. For simplicity, let logical gates from any set $\mathcal{T}_l$ for $l \geq 3$ be called t gates. Using a unique canonical form [16] for sequences consisting of the Clifford gates and combinations of $\mathcal{T}_l$ gates, arbitrary gate sequences can be reduced to the form

$$c.t_1.H.t_2.H \ldots t_M.c', \tag{11}$$

where c and c' are Clifford gates, t_m is the m^{th} positioned t gate in the sequence, and M is the t-count. For a particular sequence, let the number of t gates from $\mathcal{T}_l$ be denoted by k_l. It follows that each sequence consisting of gates from up to order L of the Clifford hierarchy satisfies (noting that $c_0 = c_1 = 0$)

$$\sum_{l=3}^{L} c_l k_l \leq C, \tag{12}$$

where c_l is the cost assigned to logical gates from $\mathcal{T}_l$ and C is the maximum cost of the database of gate sequences. It will be useful to denote the number of t gates from order l to L of the Clifford hierarchy as

$$K_l := \sum_{i=l}^{L} k_i, \tag{13}$$

noting that K_3 is the t-count, M that appears in Eq. 11.

The aim is to calculate the proportion of $\mathcal{T}_n$ gates among all gates in sequences within the database. We begin by counting the total number of possible sequences that can be formed given

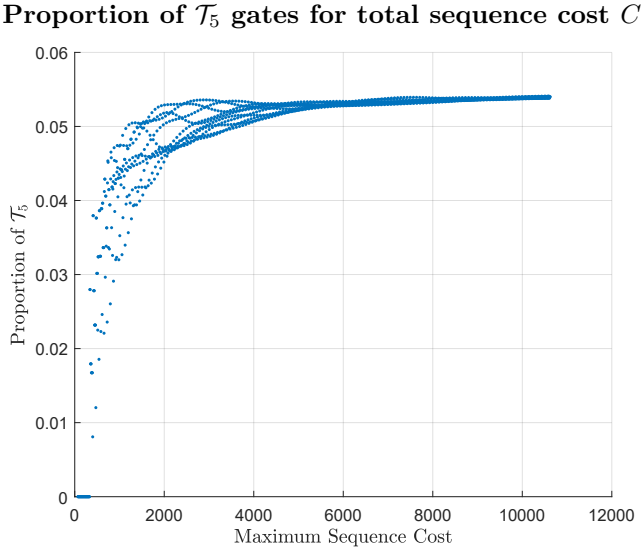


Figure A.2: The proportion of $\mathcal{T}_5$ logical base gates among $\mathcal{T}_3 \cup \mathcal{T}_4 \cup \mathcal{T}_5$ gates calculated using the combinatorial model for all cost-optimal sequences below a maximum sequence cost that produce distinct combined gates. The logical base gate cost values are assigned according to Table 2 for a logical base gate error threshold of $\mu = 10^{-15}$ under the diamond norm. This plot shows that the proportion of $\mathcal{T}_5$ gates becomes approximately constant for sufficiently large maximum sequence costs.

a set of t gate counts $\{k_l\}|_3^L$. Then the total number of possible sequences can be summed by iterating through every combination of possible sets $\{k_l\}|_3^L$ that satisfy Eq. 12 with their assigned base gate costs. Once this expression is determined, it can be extended to calculate the number of $\mathcal{T}_n$ gates and the total number of gates, which can then be used to calculate the proportions. For sequences of t-count K_3, the number of permutations of k_l gates within K_3 gate locations is

$$(\#\text{Permutations}(k_l, K_3)) := \binom{K_3}{k_l} = \frac{K_3!}{(K_3 - k_l)!k_l!}. \tag{14}$$

Let $|\mathcal{T}_l|$ be the number of distinct Z-rotation gates within order l of the Clifford hierarchy, for example, $|\mathcal{T}_3| = 2$ since $\mathcal{T}_3 = \{T, T^\dagger\}$ (up to global phase). Then for each permutation, there are $|\mathcal{T}_l|^{k_l}$ unique combinations of assigned $\mathcal{T}_l$ logical base gates within the permutation. Thus, the total number of configurations for k_l number of gate locations with $|\mathcal{T}_l|$ variations in a sequence of t gate count K_3 is

$$\gamma(k_l, |\mathcal{T}_l|, K_3) := |\mathcal{T}_l|^{k_l} \frac{K_3!}{(K_3 - k_l)!k_l!}. \tag{15}$$

After assigning gates to k_l locations, there are $K_3 - k_l$ locations remaining within the sequence. The strategy from here is to iteratively count the total number of configurations from $l = 3$ to L by updating the number of remaining locations at each step, which now updates as $K_{l+1} = K_l - k_l$. So for the second iteration, the number of configurations of k_{l+1} gates with $|\mathcal{T}_{l+1}|$ variations within remaining locations K_{l+1} of a given configuration from the assigned k_l number of $\mathcal{T}_l$ gates is $\gamma(k_{l+1}, |\mathcal{T}_{l+1}|, K_{l+1})$, leading to a total of $\gamma(k_l, |\mathcal{T}_l|, K_l)\gamma(k_{l+1}, |\mathcal{T}_{l+1}|, K_{l+1})$ configurations for k_l and k_{l+1} numbers of $\mathcal{T}_l$ and $\mathcal{T}_{l+1}$ gates respectively in sequences of t-count K_l. Thus the total number of configurations for a set of t gate counts $\boldsymbol{k} = \{k_3, k_4, \ldots, k_L\}$ in sequences of t-count K_3

(containing t gates up to order L of the Clifford hierarchy) is

$$\Gamma(\boldsymbol{k}) := \prod_{l=3}^{L} \gamma(k_l, |\mathcal{T}_l|, K_l) = \prod_{l=3}^{L} |\mathcal{T}_l|^{k_l} \frac{K_l!}{(K_l - k_l)! k_l!} \tag{16}$$

$$= \frac{K_3! K_4! \dots K_L!}{K_4! \dots K_L! (K_L - k_L)!} \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!} = K_3! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!} \tag{17}$$

$$= \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}. \tag{18}$$

To count the total number of sequences, we sum over all configurations for each assignment of $\boldsymbol{k}$ satisfying Equation 12. We begin by determining the maximum allowable values for each k_l with respect to already specified lower order t gate counts $\{k_j\}_3^{l-1}$. The maximum possible value for k_3 is $\lfloor C/c_3 \rfloor$. Given a specified k_3, the maximum value for k_4 is $\lfloor (C - c_3 k_3)/c_4 \rfloor$. By continuing this pattern, given a set of t gate counts $\{k_3, k_4, \dots, k_{l-1}\}$, the maximum value for k_l is

$$\max(k_l) = \lfloor (C - \sum_{j=3}^{l-1} c_j k_j)/c_i \rfloor. \tag{19}$$

So now the total number of sequence configurations with logical base gate costs $\boldsymbol{c}$ and maximum sequence cost C can be calculated as

$$\zeta(\boldsymbol{c}, C) := \sum_{\{\boldsymbol{k} \mid \boldsymbol{c}\cdot\boldsymbol{k} \leq C\}} \Gamma(\boldsymbol{k})$$

$$= \sum_{k_3=0}^{\lfloor C/c_3 \rfloor} \sum_{k_4=0}^{\lfloor (C - c_3 k_3)/c_4 \rfloor} \cdots \sum_{k_L=0}^{\lfloor (C - \sum_{j=3}^{L-1} c_j k_j)/c_L \rfloor} \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}. \tag{20}$$

Since the number of $\mathcal{T}_l$ logical gates within a particular sequence is k_l, the total number of $\mathcal{T}_l$ gates within all possible sequences below the maximum cost C is calculated by multiplying k_l to each term in the summation, the total number of gates can be calculated in a similar way. Thus, the proportion of $\mathcal{T}_n$ gates can be calculated as the weighted sum

$$p_n = \frac{\sum_{\{\boldsymbol{k} \mid \boldsymbol{c}\cdot\boldsymbol{k} \leq C\}} k_n \Gamma(\boldsymbol{k})}{\sum_{\{\boldsymbol{k} \mid \boldsymbol{c}\cdot\boldsymbol{k} \leq C\}} \sum_{t=3}^{L} k_t \Gamma(\boldsymbol{k})} \tag{21}$$

$$= \frac{\sum_{k_3=0}^{\lfloor C/c_3 \rfloor} \sum_{k_4=0}^{\lfloor (C - c_3 k_3)/c_4 \rfloor} \cdots \sum_{k_L=0}^{\lfloor (C - \sum_{j=3}^{L-1} c_j k_j)/c_L \rfloor} k_n \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}}{\sum_{k_3=0}^{\lfloor C/c_3 \rfloor} \sum_{k_4=0}^{\lfloor (C - c_3 k_3)/c_4 \rfloor} \cdots \sum_{k_L=0}^{\lfloor (C - \sum_{j=3}^{L-1} c_j k_j)/c_L \rfloor} \sum_{t=3}^{L} k_t \left(\sum_{i=3}^{L} k_i \right)! \prod_{l=3}^{L} \frac{|\mathcal{T}_l|^{k_l}}{k_l!}}. \tag{22}$$