

UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Techniques for Improving Resource Usage in Near-Term Quantum Computations

Permalink

<https://escholarship.org/uc/item/4w37t06j>

Author

Wong, Raymond

Publication Date

2018

Peer reviewed|Thesis/dissertation

University of California
Santa Barbara

Techniques for Improving Resource Usage in Near-Term Quantum Computations

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Raymond Garwei Wong

Committee in charge:

Professor Wim van Dam, Chair
Professor Ömer Egecioglu
Professor Subhash Suri

September 2018

The Dissertation of Raymond Garwei Wong is approved.

Professor Ömer Egecioglu

Professor Subhash Suri

Professor Wim van Dam, Committee Chair

September 2018

Techniques for Improving Resource Usage in Near-Term Quantum Computations

Copyright © 2018

by

Raymond Garwei Wong

To my family.

Acknowledgements

First, I want to express my gratitude to my advisor Wim van Dam for his guidance and especially his patience. I can recall many times when I stumbled to meet my goals, but his hands-off approach meant there was ample time for me to learn from my mistakes and develop as an independent, knowing I can approach him for advice when needed.

I would also like to thank the Computer Science department for their behind the scenes support, my doctoral dissertation committee – all of whom I had the pleasure of working with as a teaching assistant – for their counsel throughout my graduate studies, and the National Science Foundation. The material presented here is based upon work supported, in part, by the National Science Foundation under Grant Nos. 0917244 and 1719118. Many thanks to all my colleagues and peers who have made my time memorable. Finally, my family deserves much credit for my success through their unconditional support.

Curriculum Vitæ

Raymond Garwei Wong

Education

- 2018 Ph.D. in Computer Science (Expected), University of California, Santa Barbara.
- 2012 B.S. in Computer Science, California Polytechnic State University, San Luis Obispo.

Publications

W. van Dam and R. Wong, *Two-qubit Stabilizer Circuits with Recovery I: Existence*, in *Proceedings of the Thirteenth Conference on the Theory of Quantum Computation, Communication and Cryptography*, TQC '18, (Dagstuhl, Germany), pp. 7:1-7:15, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

W. van Dam and R. Wong, *Two-qubit Stabilizer Circuits with Recovery II: Analysis*, in *Proceedings of the Thirteenth Conference on the Theory of Quantum Computation, Communication and Cryptography*, TQC '18, (Dagstuhl, Germany), pp. 8:1-8:21, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Abstract

Techniques for Improving Resource Usage in Near-Term Quantum Computations

by

Raymond Garwei Wong

Quantum computers have many desirable features but are physically challenging to build. They use quantum physics to solve practically motivated yet classically intractable problems, and because the experimental technology is still in its infancy, quantum mechanical devices are susceptible to errors that compromise data integrity. As a result, quantum error correction is necessary to protect important information from such undesirable influences, which inevitably increases the resource overhead to ensure a reliable quantum computation. In this thesis, we develop methods that are relevant to reducing the utilization of physical resources in a quantum computer. The core operations considered here are the so-called stabilizer operations, which have fault-tolerant constructions that are vital to achieving an error-resistant quantum computation. By applying our practices, we achieve small optimizations that have considerable value when implementing quantum algorithms in the near-term, when small quantum systems are much easier to manage than a single large quantum system. We cover two techniques to improve the efficiency of stabilizer operations applied to qubit states.

First, we introduce protocols that can probabilistically recreate an initial input qubit from the output qubit of specific quantum processes. These protocols are ideally suited for recovery purposes, and are designed with potentially many nested layers of stabilizer operations. We subsequently give a precise analysis on the effectiveness of the nested recovery. By integrating recovery at the optimal nesting depth, the resource usage of the relevant quantum processes can be reduced by up to half in expectation. Second, we

define a new special arrangement of elementary stabilizer operations for realizing certain quantum computations, which we call a binary in-tree decomposition. We show that such implementations lead to a better process for lowering resource consumption. We then propose an efficient classical algorithm to assemble stabilizer operation sequences with such binary in-tree form. Finally, we demonstrate the merits of the binary in-tree structure on several examples.

Contents

Curriculum Vitae	vi
Abstract	vii
1 Introduction	1
1.1 Benefits of Quantum Computing	1
1.2 Main Results	3
1.3 Outline	4
1.4 Quantum States	5
1.5 Quantum Operations and Measurements	10
1.6 Quantum Circuits	11
1.7 Noise	15
2 Stabilizer Quantum Computation	18
2.1 Stabilizer Formalism	18
2.2 Universal Quantum Computation	28
3 Postselected Stabilizer Circuits	37
3.1 Notation	37
3.2 Basic Definitions	38
3.3 Properties of Postselected Stabilizer Circuits	40
3.4 Summary	47
4 Two-qubit Stabilizer Circuits with Recovery	48
4.1 Notation and Conventions	49
4.2 Postselected Two-to-One Stabilizer Circuits	49
4.3 Two-qubit Recovery Circuits	53
4.4 Example Routines Featuring Recovery Circuits	62
4.5 Summary	64

5	Extending the Recovery for Two-qubit Stabilizer Circuits	65
5.1	Nested Recovery Protocol	65
5.2	Experimentation with Recovery Circuits	70
5.3	Summary	74
6	Performance Analysis of Nested Recovery	75
6.1	Prelude to Analysis	75
6.2	Expected Cost	78
6.3	Minimizing Expected Cost	80
6.4	Cost Ratio	83
6.5	Potential Improvements with Commonly Used Resource Qubits	85
6.6	Summary	86
7	Stabilizer Circuits with Binary In-tree Form: Introduction	91
7.1	Case Study: Four-qubit Quantum Circuit	91
7.2	Basic Concepts, Notation, and Review	94
7.3	Postselected and Delegated Two-Op Circuits	94
7.4	Binary In-tree Decomposition	97
7.5	Multistep Tree Execution and Expected Cost	100
8	Synthesizing Stabilizer Circuits with Binary In-tree Form	105
8.1	Basic Property of Binary In-tree Unitaries	105
8.2	Stabilizer Matrices and Stabilizer Matrix Forms	107
8.3	Synthesis of Binary In-tree Clifford Circuits	111
8.4	Partial Binary In-tree Form Circuit Synthesis	122
8.5	Examples	122
8.6	Summary	131
9	Conclusion	133
9.1	Recovery Circuits	134
9.2	Stabilizer Circuits with In-tree Form	134
A	Bounded One-Dimensional Random Walk with Difference Equation	138
A.1	Bounded One-Dimensional Random Walk	138
A.2	Random Walk with Difference Equation	140
B	Index of Terms	154
	Bibliography	157

Chapter 1

Introduction

Computers continue to evolve and influence the world in monumental ways. They are deeply integrated into many functions of society, ranging from our largest financial, energy, and health institutions to the local businesses and organizations within our communities. Equipped with rich features and capabilities, present computers enable us to engage in activities not possible or otherwise difficult in the past. However, they are not without physical limits. The diminishing size of components powering our electronic instruments means we may no longer ignore the phenomena encroaching from the quantum realm. Fortunately, we can exploit these quantum effects to our benefit in a number of meaningful applications. As hardware reliability steadily improves [50], it is only a matter of time before quantum computers become more commonplace. Even now, research and development into a quantum machine is starting to follow a trajectory similar to classical computer design, dividing into subareas that target different abstraction levels of the quantum computer system [40].

1.1 Benefits of Quantum Computing

Quantum computers have garnered significant interest in recent years, and the trend suggests that the potential gains are enough to warrant large investments by industry[1,

44] and government [2] to researching new quantum technologies. There are several notable applications for a quantum computer, and it is imperative we mention them to understand some of the forces driving this development. Many more can be found in the Quantum Algorithm Zoo [42] by Stephen Jordan, a catalog of different scenarios in which a quantum computer surpasses a classical computer in speed.

The most widely known and cited example in support of quantum computers is perhaps Shor's algorithm for integer factorization and discrete logarithm [67]. The existence of such a fast algorithm has serious implications in computer security, as confidence in elliptical curve cryptography and RSA encryption is predicated on the hardness of factoring and the discrete logarithm. This is not the only problem where we see a superpolynomial speedup over the best known classical solutions. Feynman [29] proposes that a quantum computer inherently can perform quantum simulations much faster than classical computers. Problems of physical interest include many-body systems [4, 14] and quantum chemistry [37, 41].

Elsewhere, Grover's algorithm permits an $O(\sqrt{n})$ search for an object in an unordered list of n items [35]. Clever embedding of Grover's search as a subroutine has also lead to other impressive algorithms, such as Ramesh and Vinay's $O(\sqrt{n} + \sqrt{m})$ approach to string matching, where n is the text length and m is the pattern length [60]. A recent survey by Ambainis [6] documents the quantum advantage in the language of query complexity. In particular, there exist problems involving partial functions $f(x_1, \dots, x_n)$ on n variables in which a quantum computer only needs 1 query to f to solve but $\Omega(\sqrt{n})$ classically. The gap is even larger when f is a total function.

Quantum cryptography is another area with potential. The security properties guaranteed by quantum cryptography rely on the *no-cloning theorem* [74], a fundamental law that prohibits copying of arbitrary unknown quantum states. One of the earliest protocols is BB84 by Bennett and Brassard [9] to exchange secret encryption keys between

multiple parties in a secure fashion. Several more quantum key distribution schemes have been devised ever since [28, 46, 54], and recent experiments [51, 59, 75] show promise into the viability of using quantum mechanics for secure communication.

1.2 Main Results

This thesis is broadly concerned with program optimization in the quantum setting. In the same way we want to improve the memory footprint or runtime of a classical algorithm, we can identify quantum resources that we wish to utilize more efficiently in a quantum computation. Unfortunately, devices that store quantum information are highly sensitive to influences from the environment and faulty hardware. While quantum error correction provides a measure of protection against unwanted errors, it inevitably drives up the number of quantum resources necessary to ensure a reliable calculation. The work presented herein introduces two techniques to reduce the resource demand of certain quantum processes, and they succeed by taking advantage of how we use *stabilizer operations*, a prominent subset of all quantum mechanical transformations.

1.2.1 Nested Recovery Protocol

In addition to error correction, the quantity of resources used is also affected by quantum mechanics' probabilistic nature, which implies a quantum algorithm generally does not provide the correct solution on the every execution. Thus until the desired answer is obtained, we must successively repeat the entire computation. The impact on resources is greatest when the quantum subroutine in question is “expensive”, though depending on the circumstances, there may be ways to circumvent this.

We will show that special kinds of small-scale stabilizer operations are “invertible” when provided with select types of quantum state inputs. The suggestion is that we can

recover some of the work that was recently modified and return to a slightly earlier stage of the process without restarting from the beginning; the remaining elements can be acquired with relatively little extra effort. The caveat is that these “inversions” are also probabilistic, so the recovery is not always successful. Luckily, these “inverse” activities require few resources to proceed and are “invertible” themselves, and we may try to recover from an unsuccessful recovery. This “recovery of recovery” is the basis for what we shall call as our *nested recovery protocol*. The “expensive” subroutine is rerun when the protocol is unsuccessful at some predetermined nesting limit, but by incorporating recovery, we contribute to an overall decrease in the average resources consumed.

1.2.2 Binary In-tree Implementation

Our second approach to conserving resources looks at the *binary in-tree implementation* of a stabilizer operation, named so for our interpretation of the quantum process as a directed acyclic graph (DAG) with the shape of a binary tree. The problem of producing such implementations is somewhat analogous to compiling, where the objective is to translate a computer program in human-readable code into assembly instructions. In this situation, we want to convert a stabilizer operation into a special sequence of more basic units of stabilizer operations. We give an efficient classical algorithm that does exactly that, and what we see in the ensuing decomposition is a looser coupling of subprocesses. The relative isolation enables us to restart one task within the larger procedure without repeating the whole, and hence become more economical with our resources.

1.3 Outline

For completeness, the remainder of this chapter is dedicated to a general overview on the field of quantum computation; for a more comprehensive introduction and his-

torical perspective, see Nielsen and Chuang's text [56]. Chapter 2 contains more specific background material and gives context to our problem of interest. Chapter 3 provides basic definitions and some elementary results to set up the primary outcomes found in Chapters 4 to 8. Details related to our nested recovery protocol are discussed in Chapters 4 to 6, whereas Chapters 7 and 8 examine the binary in-tree implementation of stabilizer operations. The final chapter offers some concluding words.

1.4 Quantum States

Whereas a classical computer processes strings of 0s and 1s, the basic unit of quantum information is a quantum bit, or simply *qubit*. In bra-ket notation, the quantum parallels of the classical bits are the *computational basis qubits* $|0\rangle$ and $|1\rangle$. And even though there are 2^n possibilities, the n bits of a classical computer carry only one such state at any given moment, so the dimensions of a classical state is still n . In contrast, an n -qubit quantum state lies in a 2^n -dimensional complex vector space. With $|0\rangle$ and $|1\rangle$ forming an orthonormal basis, an arbitrary single qubit $|\psi\rangle$ can be described as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (1.1)$$

where α and β are complex numbers that satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Notice that a qubit is allowed to take on a *quantum superposition* – a sum – of other qubits as seen above. Being vectors, we may alternatively express qubits in column matrix form, where the matrix versions of $|0\rangle$, $|1\rangle$, and $|\psi\rangle$ are

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad |\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \quad (1.2)$$

The Hermitian conjugate (conjugate transpose) of a *ket* $|\psi\rangle$ returns a row vector and is denoted by a *bra* $\langle\psi|$.

A geometric representation of qubits called the *Bloch sphere* is provided in Figure 1.1. In this picture, $|\psi\rangle$ is parameterized by two angles $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ and specified as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle. \quad (1.3)$$

The $|0\rangle$ and $|1\rangle$ basis qubits occupy the north and south poles of the sphere, respectively. Besides $|0\rangle$ and $|1\rangle$, other points of considerable interest are

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \quad (1.4)$$

which lie on the intersection of the x -axis and the unit sphere, and

$$|+i\rangle = \frac{|0\rangle + i|1\rangle}{\sqrt{2}}, \quad |-i\rangle = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}, \quad (1.5)$$

which fall on the intersection with the y -axis.

The tensor product “ \otimes ” is used to form larger quantum states. When dealing with two matrices A and B , of dimensions $m \times n$ and $r \times s$, the tensor product is the same as the Kronecker product:

$$A \otimes B = \begin{bmatrix} A_{1,1}B & \cdots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{m,1}B & \cdots & A_{m,n}B \end{bmatrix}. \quad (1.6)$$

yielding a new matrix $A \otimes B$ that has mr rows and ns columns. Some example two-qubit

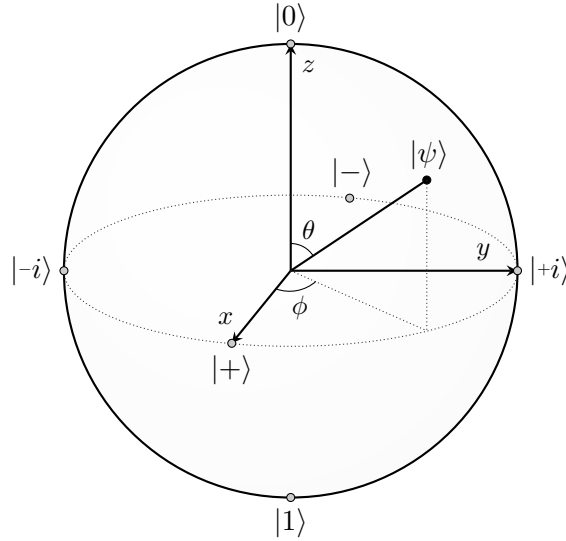


Figure 1.1: Bloch sphere representation of a qubit state. The sphere has radius 1, and the qubit $|\psi\rangle$ is a unit vector that has an angle $0 \leq \theta \leq \pi$ with respect to the z -axis and $0 \leq \phi \leq 2\pi$ with respect to the x -axis.

states that we can produce with $|0\rangle$ and $|1\rangle$ are

$$|0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \quad (1.7)$$

More generally, if we have a length n bit vector $b_1 \dots b_n$ whose decimal value is $x = 2^{n-1}b_1 + \dots + 2b_{n-1} + b_n$, then the corresponding quantum state $|x\rangle = |b_1\rangle \otimes \dots \otimes |b_n\rangle$ is a size $2^n \times 1$ column matrix with a one in row $x + 1$ and a zero in the remaining $2^n - 1$ places. We can formulate any n -qubit state $|\psi\rangle$ in this computational basis as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \quad (1.8)$$

where α_i are complex values such that $|\alpha_0|^2 + \dots + |\alpha_{2^n-1}|^2 = 1$. For convenience, the

tensor product symbol is sometimes omitted, as illustrated by $|0\rangle|0\rangle$, $|0,0\rangle$, and $|00\rangle$. Another shorthand notation is the repeated n -fold tensor product that appears as a superscript, such as in $|0\rangle^{\otimes n}$.

Multiqubit states $|\psi\rangle$ that cannot be decomposed into a tensor product of smaller quantum states $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ are said to be *entangled*; otherwise we say $|\psi\rangle$ is a *product state*. The most common entangled two-qubit state is the Einstein-Podalsky-Rosen (EPR) pair

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \quad (1.9)$$

Entanglement is one of the fundamental differences that enable quantum computers to attain the speedups mentioned previously [43].

Since quantum systems are represented mathematically as objects in a vector space, naturally there will be an *inner product* between quantum states. For starters, the inner product between two computational basis vectors $|i\rangle$ and $|j\rangle$ is $\langle i|j\rangle = 1$ when $i = j$ and 0 otherwise. The inner product between arbitrary quantum states $|\psi_1\rangle = \sum_i \alpha_i |i\rangle$ and $|\psi_2\rangle = \sum_j \beta_j |j\rangle$ is then

$$\langle \psi_1 | \psi_2 \rangle = \sum_{i,j} \alpha_i^* \beta_j \langle i | j \rangle = \sum_i \alpha_i^* \beta_i. \quad (1.10)$$

Besides the inner product, there also exists an *outer product* written $|\psi_1\rangle\langle\psi_2|$ for two vectors $|\psi_1\rangle$ and $|\psi_2\rangle$. This forms a linear operator defined by

$$(|\psi_1\rangle\langle\psi_2|) |\psi_3\rangle = \langle\psi_2|\psi_3\rangle |\psi_1\rangle. \quad (1.11)$$

All the quantum states presented so far are classified as *pure states*. Such a vector is indicated by the ket $|\psi\rangle$, or by the *density matrix* $|\psi\rangle\langle\psi|$ using the outer product. An

n -qubit *mixed state* ρ is defined as a weighted distribution of n -qubit pure states $|\psi_i\rangle$:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad (1.12)$$

where p_i are probabilities (not complex amplitudes) that sum to unity. This means the system has probability p_i of being in the state $|\psi_i\rangle$. Any n -qubit quantum system, pure or mixed, has a $2^n \times 2^n$ density matrix description, and a defining quality of a density matrix ρ is the fact that its trace equals one i.e. $\text{tr}(\rho) = 1$.

There is a simple criterion to differentiate between pure and mixed states. Single qubit systems also have a nice graphical determinant. If we consider the single qubit Pauli matrices

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad (1.13)$$

we can rewrite a single qubit ρ as

$$\rho = \frac{1}{2}(I + xX + yY + zZ) \quad (1.14)$$

where the real coefficients $(x, y, z) \in \mathbb{R}^3$ form the *Bloch vector* of ρ . Throughout, we use I to stand for the identity operator; its dimensions should be clear from context. The Bloch vector allows us to visualize ρ in the Bloch sphere as a vector with coordinates (x, y, z) in the standard basis and whose norm satisfies $\sqrt{x^2 + y^2 + z^2} \leq 1$. In other words, the vector endpoint stays within the unit sphere's interior if the qubit ρ is mixed and on the surface if pure. For multiqubit density matrices ρ , the distinction between pure and mixed is given by the trace operation on ρ^2 : a pure state means $\text{tr}(\rho^2) = 1$ and $\text{tr}(\rho^2) < 1$ when mixed. Note that density matrices are positive semidefinite and have

nonnegative eigenvalues λ_i . Diagonalizing a density matrix ρ shows that $\text{tr}(\rho^2) = \sum_i \lambda_i^2$. Lastly, the scalar complex constants of quantum states called *global phases* serve no practical purpose, so we may treat, for instance, $e^{i\pi/4} |\psi\rangle$ and $|\psi\rangle$ as being the same.

1.5 Quantum Operations and Measurements

Quantum operators describe changes in a quantum state. Without the application of quantum operators, qubits remain static. The operations are linear, and one of the primary types that we encounter are *unitary transformations*, which are reversible unlike most boolean logic functions. The usual definition states that a matrix U is unitary if and only if the Hermitian conjugate U^\dagger is also the inverse, meaning $U^\dagger = U^{-1}$. Then given an initial density matrix ρ , the state of a quantum system after some process described by a unitary matrix U is $\rho' = U\rho U^\dagger$. A property of unitary operators is that they preserve the inner product i.e. $\text{tr}(\rho) = \text{tr}(\rho') = 1$.

In addition to unitary operations, we depend on measurements that probe a quantum system to reveal information about its state. Measurements are generally *destructive* in the sense that they are, for the most part, irreversible. A special kind are projective measurements. These measurement processes are often characterized by a set of orthogonal projectors $\{P_m = |\psi_m\rangle\langle\psi_m|\}$ that fulfill the completeness relation

$$I = \sum_m P_m. \quad (1.15)$$

We immediately see from this definition that $P_m = P_m^\dagger$ and $P_m^2 = P_m$ (the latter meaning idempotent). Each basis state $|\psi_m\rangle$ is associated with an observable result m to help us identify the specific action that took place. More precisely, given a quantum state $|\psi\rangle$, the probability of projecting onto $|\psi_m\rangle$ is $\langle\psi|P_m|\psi\rangle$. If we detect m during measurement,

then the state of the quantum system after measurement is

$$|\psi'\rangle = \frac{P_m |\psi\rangle}{\sqrt{\langle\psi|P_m|\psi\rangle}}. \quad (1.16)$$

If we consider the superposition of $|\psi\rangle$ in the $\{|\psi_m\rangle\}$ basis, then the measurement causes the superposition to *collapse* onto the $|\psi_m\rangle$ state. The division is necessary for $|\psi'\rangle$ to stay normalized.

These postulates of quantum mechanics extend to a density operator ρ . The probability of observing m is $\text{tr}(P_m\rho)$, and the postmeasurement state becomes

$$\rho' = \frac{P_m\rho P_m}{\text{tr}(P_m\rho)} \quad (1.17)$$

since $\text{tr}(P_m\rho) = \text{tr}(P_m\rho P_m)$. This holds due to the invariance of the trace operation under cyclic permutations, and the idempotence of the projection operators. As usual, we can build up larger quantum operators using the tensor product, and may even construct a combination of unitaries and projectors e.g. $X \otimes |0\rangle\langle 0|$. This forms a two-qubit operator that performs a Pauli X on qubit one and projects qubit two onto $|0\rangle$.

1.6 Quantum Circuits

There exist several models for expressing quantum computations, some of which, like quantum Turing machines and quantum random access machines, are simply ported over from classical computation theory [53]. Then there are others – adiabatic quantum computing [22] and topological quantum computing [55] – that are less abstract and a little more suggestive of the physical means to carry out the computation. In this thesis, we shall follow quantum circuits, essentially the quantum analogue of boolean circuits. As such, quantum circuits describe a quantum computation as a sequence of elementary

quantum logic gates and measurements that act on an array of qubits. The qubits are usually initialized to some well-defined state. The only measurements considered here are with respect to the Pauli Z or computational basis, which involve the projectors

$$|0\rangle\langle 0| = \frac{I + Z}{2}, \quad |1\rangle\langle 1| = \frac{I - Z}{2}. \quad (1.18)$$

A Z -measurement will therefore return one classical bit b of information, and the post-measurement qubit will be $|b\rangle$.

We have already seen some quantum gates for single qubits: the Pauli spin matrices. They play a central role in several areas of quantum computing, in part because they satisfy the following relationships:

$$XY = iZ, \quad YX = -iZ, \quad (1.19)$$

$$YZ = iX, \quad ZY = -iX, \quad (1.20)$$

$$ZX = iY, \quad XZ = -iY. \quad (1.21)$$

The Pauli gates are also *Hermitian* i.e. $X = X^\dagger$, $Y = Y^\dagger$, $Z = Z^\dagger$ to yield

$$X^2 = Y^2 = Z^2 = I. \quad (1.22)$$

The Pauli X and Z unitaries also go by the name of *bit flip* and *phase flip* gates, since $X|0\rangle = |1\rangle$ and $Z|+\rangle = |-\rangle$. A few other quantum gates of particularly special interest are the Hadamard (H), Phase (P), and $\pi/8$ (T) operators:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \quad (1.23)$$

A nice way of thinking about single qubit unitary operations are as 3-dimensional rotations of the qubit vector in the Bloch sphere e.g. P and T rotate a qubit about the z -axis by $\pi/2$ and $\pi/4$, respectively. More general x , y , and z -axial spins are defined by

$$R_x(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X = \begin{bmatrix} \cos\frac{\theta}{2} & -i \sin\frac{\theta}{2} \\ -i \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \quad (1.24)$$

$$R_y(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \quad (1.25)$$

$$R_z(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}. \quad (1.26)$$

In fact, there exist four real numbers α , θ , ϕ , γ such that any 2×2 unitary matrix U is decomposable into four parts as

$$U = e^{i\alpha} R_z(\theta) R_y(\phi) R_z(\gamma). \quad (1.27)$$

Beyond single qubit operations, we require multiqubit gates. Among the most useful are controlled-operations, and the one that we will see most is the two-qubit Controlled-NOT (CNOT) gate. We can gain a firmer grasp of controlled-operations via an explanation of the Controlled-NOT. In particular, one of the two qubits is designated the *control qubit*, and the other being the *target*. The control qubit acts like a guard and dictates whether the Pauli X gate is applied to the target qubit or not. That is, a bit flip on the target qubit occurs if the control qubit is $|1\rangle$. In the computational basis, the CNOT is governed by the following action:

$$\text{CNOT } |a\rangle |b\rangle = |a\rangle |a + b \bmod 2\rangle \quad (1.28)$$

assuming the first (leftmost) qubit is the control. We can determine the matrix to this two-qubit operation on adjacent qubits using the rule above:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (1.29)$$

The Controlled-NOT is simple, but the basic function of all controlled-operations follow the same principle. In general, there could be many control and target qubits. Suppose we have an $(n + k)$ -qubit controlled- U gate, where U is some unitary operation on the last k qubits. Then U is applied if the first n control qubits are all $|1\rangle$ s. This is easiest to see from the three-qubit Toffoli gate (denoted CCX). To be specific, we have

$$\text{CCX} |b_1\rangle |b_2\rangle |b_3\rangle = |b_1\rangle |b_2\rangle X^{b_1 \cdot b_2} |b_3\rangle, \quad (1.30)$$

so the Pauli X gate activates if $b_1 \cdot b_2 = 1$. This outcome occurs whenever both control qubits are $|1\rangle$. Not surprisingly, we may also apply quantum gates conditional on classical measurement results. For example, we can perform a phase flip Z on the second qubit if the Z -measurement on the first qubit returns a bit $b = 1$.

Other common two-qubit operations include the SWAP gate to interchange two qubits, the Controlled- Z (CZ), and the Controlled- Y (CY). The latter two work the same as Controlled-NOT, but instead apply Paulis Z and Y , respectively. To avoid confusion, we sometimes add bracketed superscripts that explicitly reference the qubits involved, such as $\text{CNOT}^{(i,j)}$ to mean control qubit i and target qubit j . Otherwise, CNOT, CZ, and CY are controlled-Pauli gates that act on neighboring qubits e.g. $\text{CNOT}^{(i,i+1)}$. The gate identity $\text{SWAP}^{(i,j)} = \text{CNOT}^{(i,j)} \text{CNOT}^{(j,i)} \text{CNOT}^{(i,j)}$ is also worth remembering.

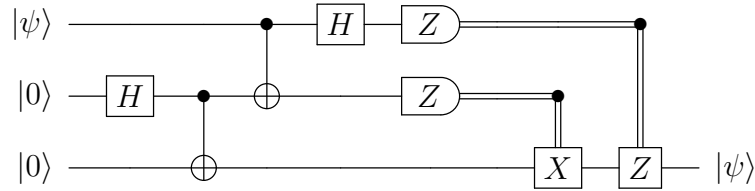


Figure 1.2: Example of a quantum circuit. The circuit displayed here implements a technique called quantum teleportation [10]. The double lines after the Z -measurements represent classical bits. The final two operations of quantum teleportation are to apply Pauli X on qubit three if the second measurement returns a bit $b_2 = 1$, and Pauli Z if the first measurement returns a bit $b_1 = 1$.

Finally, we have quantum circuit diagrams. A quantum circuit diagram, such as the one depicted in Figure 1.2, illustrates how a large and complex procedure is implemented by a series of basic quantum operations. Each wire represents a qubit, and the diagram is read left to right to indicate the passage of time. The images for several quantum circuit components discussed here are summarized in Table 1.1.

1.7 Noise

The biggest hurdle to building quantum computers is perhaps *noise*, a generic label to describe a set of error-causing processes that arise due to computer design flaws and manufacturing defects. The primary culprits of quantum error are no different than classical error: (1) faulty computer parts, and (2) the environment, because of imperfect shielding of idle qubits in data storage. This latter source of noise leads to a process called *quantum decoherence*. Since a classical computer stores bits, the main problems are isolated to bit flip errors, and bit erasure errors in a communications channel. However, the errors we see disturbing a quantum system are more diverse. In addition to the bit flip, there is also the phase flip error, which is especially relevant, for instance, when we have a qubit in the state $|+\rangle$. These two flip operations are simply seen as an application

of the Pauli X and Z gates, respectively. Because $Y = iXZ$, the Pauli Y gate represents a simultaneous bit-and-phase flip error.

Much like the classical noise model, errors that afflict individual qubits are assumed to be independent. Even if noise affect multiple qubits, the independence assumption is a good approximation [56]. Beyond bit flip and phase flip, there is another type of error unique to quantum computing that frequently appears in the literature: *depolarizing noise*. This error operation replaces a density matrix ρ with the completely mixed state $I/2$ with some probability p . The following equation describes this action on a qubit ρ :

$$\mathcal{E}_D(\rho) = \frac{pI}{2} + (1-p)\rho = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z). \quad (1.31)$$

Geometrically, the completely mixed state inhabits the center of the Bloch sphere. Another way to imagine depolarizing noise is that it causes an X , Y , or Z error with equal weighting. The second identity of \mathcal{E}_D is actually no mere coincidence, since any single qubit error can be decomposed as a linear combination of the Pauli matrices I , X , Y , and Z [33]. This allows quantum error correction methods to handle arbitrary single qubit errors by virtue of managing X and Z flips. The downside is that quantum error correction is not cheap, and generally entails a sizable price. More details on this will emerge as we progress to later chapters.

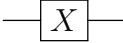
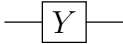
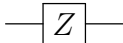

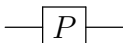
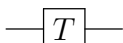
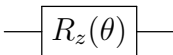
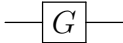
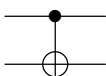

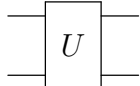
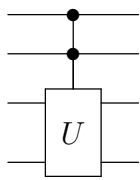
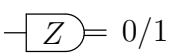
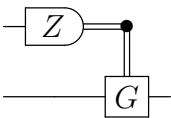
Quantum Circuit Component	Circuit Diagram
Pauli X gate	
Pauli Y gate	
Pauli Z gate	
Hadamard gate H	
Phase gate P	
$\pi/8$ or T gate	
z -axis rotation gate $R_z(\theta)$	
Single qubit gate G	
$\text{CNOT}^{(i,i+1)}$	
$\text{SWAP}^{(i,i+1)}$	
Two-qubit gate U	
Controlled- U gate: two control, two target qubits	
Measurement in Pauli Z basis that returns 0 or 1	
Classically conditioned single qubit gate G	

Table 1.1: Diagrams for various quantum circuit components.

Chapter 2

Stabilizer Quantum Computation

In this chapter, we cover the fundamentals of stabilizer quantum computation. We start with the *stabilizer formalism*, a mathematical device for understanding a large class of quantum states and operations, otherwise known as stabilizer states and stabilizer operations. The stabilizer operations in particular come highly recommended due to their ability to suppress the spread of quantum errors during the execution of quantum algorithms. For this reason, the *magic state* approach is a well established method for achieving universal quantum computation (UQC). Under this model, we draw from an elementary set of stabilizer and non-stabilizer operations to implement more complicated procedures, and associate with each non-stabilizer operation a resource “cost” that reflects the degree of difficulty to carry out the operation. The magic state model relates this resource cost of non-stabilizer operations to non-stabilizer quantum states.

2.1 Stabilizer Formalism

Quantum computers are still in the early stages of development, and many researchers around the world are actively seeking various methods to engineer these devices [50]. In an ideal world, every module of the computer would operate in perfect condition, and we would have complete control over every aspect of the quantum system. While the reality

is that defects are an unavoidable part of life, we can at least prepare solutions to help us handle a few unwelcome surprises.

In this section, we review the stabilizer theory to combating errors. The additional software layer brought in by quantum error correcting and fault-tolerant procedures is vital as the gate failure rates for all physical implementations are high enough to require their assistance. Otherwise we have no assurances to the quality of results from long quantum computations. Moreover, the notion of a quantum resource described later in this chapter is easier to comprehend given an understanding of quantum error correction and the associated overhead. The extra labor necessary to sustain a stable quantum computer draws further significance to the techniques of this thesis.

2.1.1 Pauli Operators

The foundation of the stabilizer formalism is built on the anticommuting properties of the Pauli matrices

$$\{A, B\} = AB + BA = 0 \tag{2.1}$$

for all $A, B \in \{X, Y, Z\}$ and $A \neq B$. The equation $\{A, B\} = AB + BA$ is known as the anticommutator of two operators A and B , and it is a trivial task to arrive at the anticommutation relation above using the identity $XY = iZ$.

The commutator $[A, B] = AB - BA$ is a closely related idea for which we say A and B commute if and only if

$$[A, B] = AB - BA = 0. \tag{2.2}$$

Generally speaking, two unitaries A and B may neither commute nor anticommute, but

is always one of the two cases whenever A and B are Pauli operators. For example,

$$[X \otimes X, Z \otimes Z] = (XZ \otimes XZ) - (ZX \otimes ZX) \quad (2.3)$$

$$= (-Y \otimes Y) + (Y \otimes Y) = 0. \quad (2.4)$$

Although X and Z anticommute, the presence of two anticommuting positions contribute to commuting Pauli operators overall. This observation holds for larger systems n : if g and h are n -qubit Pauli matrices, then g and h commute if and only if the number of anticommuting positions is even. The stabilizer formalism depends on this crucial property of Pauli operators for designing quantum error-correcting codes.

2.1.2 Groups of Pauli Operators

When we look past individual Pauli operators, we find that the stabilizer formalism has a strong connection to group theory. If we collect all n -fold tensor products of the single qubit Pauli matrices, and allow overall factors ± 1 and $\pm i$, then the set over these operators form the *Pauli group* on n qubits

$$\mathcal{P}(n) = \{i^k g_1 \otimes \cdots \otimes g_n \mid g_j \in \{I, X, Y, Z\}, k \in \{0, 1, 2, 3\}\}. \quad (2.5)$$

The matrices in $\mathcal{P}(n)$ are closed under multiplication, and it is fairly obvious that this set has $|\mathcal{P}(n)| = 4^{n+1}$ elements. Note that for any n -qubit density matrix ρ , we can always expand it as a linear combination

$$\rho = \frac{1}{2^n} \sum cg \quad (2.6)$$

where g is a Pauli operator with overall factor $+1$ and $c = \text{tr}(g\rho)$ is a real number.

Inside the Pauli group are abelian subgroups of varying sizes, or *stabilizer groups*. A property of commuting matrices is that they have common eigenspaces, and hence eigenvectors. Pauli operators only have eigenvalues ± 1 but generally many more eigenvectors – quantum states – to each eigenvalue. At the smallest level $n = 1$, the eigenvectors and eigenvalues of the single qubit Pauli matrices X , Y , and Z are

$$X |+\rangle = |+\rangle, \quad X |-\rangle = -|-\rangle, \quad (2.7)$$

$$Y |+i\rangle = |+i\rangle, \quad Y |-i\rangle = -|-i\rangle, \quad (2.8)$$

$$Z |0\rangle = |0\rangle, \quad Z |1\rangle = -|1\rangle. \quad (2.9)$$

These six qubits should look familiar: they are precisely the intersection points between the Bloch sphere and the x , y , z axes (see Figure 1.1). But instead of listing every element one by one, we can describe a stabilizer group $\mathcal{S} \subseteq \mathcal{P}(n)$ more compactly as k mutually commuting Pauli operators $\{s_1, \dots, s_k\}$. In group theory terminology, the s_i matrices are the *generators*, and the group is denoted with angle brackets $\mathcal{S} = \langle s_1, \dots, s_k \rangle$. When dealing with stabilizer groups, we prefer the generating set to be as small as possible. This requires the k member set to be *independent*: a Pauli operator s_1 is independent of $\{s_2, \dots, s_k\}$ if and only if $s_1 \notin \langle s_2, \dots, s_k \rangle$. Therefore a set of generators $\{s_1, \dots, s_k\}$ is independent if and only if each s_i is independent of the other $k - 1$ operators. This also means that every subgroup \mathcal{S} generated by k independent and commuting Pauli matrices has exactly $|\mathcal{S}| = 2^k$ elements.

The main feature of every stabilizer group \mathcal{S} is a special vector space $V_{\mathcal{S}}$ of n -qubit states. To be exact, $V_{\mathcal{S}}$ contains all quantum states $|\psi\rangle$ such that $g|\psi\rangle = |\psi\rangle$ for all $g \in \mathcal{S}$. The number of elements in \mathcal{S} control the dimensions of $V_{\mathcal{S}}$. Specifically, if $|\mathcal{S}| = 2^k$, then $V_{\mathcal{S}}$ has dimensionality 2^{n-k} [56]. The responsibility of $V_{\mathcal{S}}$ will become clear when we look at the stabilizer formalism applications to quantum error correction.

2.1.3 Stabilizer States

Informally, stabilizer states are the eigenvectors to sets of commuting Pauli operators. A more precise definition is the following.

Definition 1 (Stabilizer State) *Let \mathcal{S} be a stabilizer group with 2^n elements. Then an n -qubit quantum state $|\psi\rangle$ is a stabilizer state of \mathcal{S} if $g|\psi\rangle = |\psi\rangle$ for all $g \in \mathcal{S}$.*

The converse has been proven to hold: if $|\psi\rangle$ is an n -qubit stabilizer state, then there is a stabilizer group \mathcal{S} with 2^n elements such that $g|\psi\rangle = |\psi\rangle$ for all $g \in \mathcal{S}$ [3]. The association between \mathcal{S} and $|\psi\rangle$ is furthermore unique (up to a global phase), thus forming a bijection between the set of size 2^n stabilizer groups and set of stabilizer states. It should also be clear from the definition that no set \mathcal{S} will ever contain $-I$.

We may also define certain mixtures of stabilizer states solely with stabilizer group generators. If $\mathcal{S} = \langle s_1, \dots, s_k \rangle$ for independent s_i , then we say the n -qubit quantum state

$$\rho = \frac{1}{2^k} \prod_{i=1}^k (I + s_i) \quad (2.10)$$

is a *pure stabilizer state* $|\psi\rangle\langle\psi| = \rho$ when $k = n$ and a *stabilizer mixed state* when $k < n$. The latter kind should not be confused with *mixed stabilizer states*, which are more generic mixtures of stabilizer states that may not follow Equation 2.10. The nice quality is that both pure and stabilizer mixed states are efficiently representable by the k generators as a simple rectangular array. If we wish to count the number of pure stabilizer states on n qubits, we recommend the following proposition.

Proposition 1 (Aaronson and Gottesman [3], Prop. 2) *The number of pure n -qubit stabilizer states is given by the expression*

$$2^n \prod_{k=0}^{n-1} (2^{n-k} + 1) = 2^{(1/2+o(1))n^2}. \quad (2.11)$$

$$\left[\begin{array}{c|cccccc} s_1 & I & I & I & X & X & X & X \\ s_2 & I & X & X & I & I & X & X \\ s_3 & I & I & I & I & X & I & X \\ s_4 & I & I & I & Z & Z & Z & Z \\ s_5 & I & Z & Z & I & I & Z & Z \\ s_6 & I & I & I & I & Z & I & Z \end{array} \right]$$

Figure 2.1: An array storing six independent and commuting Pauli operators s_i on seven qubits. The tensor product symbols between I , X , and Z are omitted for brevity. The six Pauli operators generate a stabilizer group of $2^6 = 64$ elements and define the Steane code, one of the earliest discovered quantum error-correcting codes.

We already know the six single qubit stabilizer states located on the extremities of the x , y , and z axes. For the next four values $n = 2, 3, 4, 5$, Proposition 1 informs us there are 60, 840, 15120, and 332640 stabilizer states. An example two-qubit stabilizer state is the EPR pair $(|00\rangle + |11\rangle)/\sqrt{2}$ with the stabilizer group $\mathcal{S} = \langle X \otimes X, Z \otimes Z \rangle = \{I \otimes I, X \otimes X, Z \otimes Z, -Y \otimes Y\}$.

2.1.4 Clifford Group Operations

The Clifford group of n -qubit unitaries $\mathcal{C}(n)$ forms a subset of all possible unitary operations and is defined as the normalizer of the Pauli group $\mathcal{P}(n)$. In more precise notation and terms, the elements of the Clifford group map the Pauli group to itself under conjugation:

$$\mathcal{C}(n) = \{U \in U(2^n) \mid U\mathcal{P}(n)U^\dagger = \mathcal{P}(n)\} \quad (2.12)$$

where $U(2^n)$ is the group of $2^n \times 2^n$ unitary matrices.

The Clifford group is generated entirely by three quantum gates that we have already seen: the Hadamard, Phase, and Controlled-NOT gates, and the interesting part about Clifford operators is the effect they have on the vector space $V_{\mathcal{S}}$ of a stabilizer group \mathcal{S} .

Suppose we have a quantum state $|\psi\rangle \in V_{\mathcal{S}}$ and a Clifford unitary C . Then for all items $g \in \mathcal{S}$, we have

$$C|\psi\rangle = Cg|\psi\rangle = CgC^\dagger C|\psi\rangle. \quad (2.13)$$

Since we know CgC^\dagger is another Pauli operator, we may treat $C|\psi\rangle$ as being an eigenvector of CgC^\dagger . The result is a transformation of \mathcal{S} to another group \mathcal{S}' that fixes a different vector space $V_{\mathcal{S}'}$ of quantum states. We only have to remember how the three Clifford group generators affect a Pauli operator. For the Controlled-NOT, we return

$$\begin{aligned} \text{CNOT}(X \otimes I)\text{CNOT} &= X \otimes X, & \text{CNOT}(I \otimes X)\text{CNOT} &= I \otimes X, \\ \text{CNOT}(Z \otimes I)\text{CNOT} &= Z \otimes I, & \text{CNOT}(I \otimes Z)\text{CNOT} &= Z \otimes Z, \end{aligned} \quad (2.14)$$

whereas for the single qubit Hadamard and Phase gates, we perform

$$HXH = Z, \quad HZH = X, \quad PXP^\dagger = Y, \quad PYP^\dagger = -X. \quad (2.15)$$

It is, however, good to keep in mind that each Clifford unitary preserves the structure of the Pauli group, and is therefore uniquely determined by its mapping on the $2n$ Pauli operators $X^{(i)}$ and $Z^{(i)}$, which apply Pauli X and Z gates on qubit i and the single qubit identity elsewhere. We may use this fact to conceive an analogous formula to Proposition 1 to count the size of $\mathcal{C}(n)$.

Proposition 2 (Calderbank et al. [17]) *The number of n -qubit Clifford unitaries is given by the expression*

$$|\mathcal{C}(n)| = 2^{n^2+2n} \prod_{k=1}^n (4^k - 1). \quad (2.16)$$

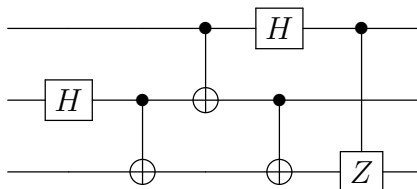


Figure 2.2: Example three-qubit unitary stabilizer or Clifford circuit.

According to Proposition 2, we gather there are $|\mathcal{C}(1)| = 24$ single qubit, $|\mathcal{C}(2)| = 11520$ two-qubit, and $|\mathcal{C}(3)| = 92897280$ three-qubit Clifford operations. Evidently, the size of $|\mathcal{C}(n)|$ grows faster than the number of pure n -qubit stabilizer states.

2.1.5 Stabilizer Circuits

Stabilizer circuits are quantum circuits that are limited to certain quantum gates and measurements and are the subject of many research activities.

Definition 2 (Stabilizer Circuit) *A stabilizer circuit is a quantum circuit that consists only of the following three kinds of components: (1) Clifford group gates, (2) measurements in the Pauli Z basis, and (3) classically conditioned Clifford group gates.*

Measurements in the Pauli X or Y basis are technically valid, but is a minor detail as we can always convert them in some way to the Z basis. On the other hand, if we exclude measurements entirely from a stabilizer circuit, then we form what is typically termed a *unitary stabilizer circuit*, or alternatively, a *Clifford circuit* (Figure 2.2). Since no measurements are involved, Clifford circuits implement group unitary operations only and are always reversible. Drawing on prior knowledge about Clifford matrices, we arrive at an alternative definition of stabilizer states with respect to such circuits: an n -qubit quantum state $|\psi\rangle$ is a stabilizer state if and only if $|\psi\rangle$ is the result of an n -qubit Clifford circuit acting on the input $|0\rangle^{\otimes n}$.

2.1.6 Gottesman-Knill Theorem

Simulating general quantum systems on classical computers is difficult due to the exponential explosion of variables to maintain, but the story changes completely for stabilizer circuits and stabilizer states. The next theorem represents one of the hallmarks of the stabilizer formalism.

Theorem 1 (Gottesman [34], Thm. 1) *A classical computer can simulate in polynomial time a quantum computation with the following elements: (1) preparation of $|0\rangle / |1\rangle$ qubits, and (2) stabilizer circuits.*

The three types of activities under Definition 2, along with stabilizer state preparation, are collectively known as stabilizer operations. Stabilizer circuits with stabilizer state inputs alone are enough to produce highly entangled states, such as the n -qubit

$$|\text{GHZ}\rangle = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}. \quad (2.17)$$

That we can forgo the $2^n - 1$ complex amplitudes of a stabilizer state means we may study a number of quantum systems within reasonable time bounds on a classical computer.

The Gottesman-Knill theorem is a byproduct of the generator representation of stabilizer states. By storing the generators as rows in a matrix, we may treat each column a qubit, and the Hadamard, Phase, and Controlled-NOT as column operations on the Pauli I, X, Y, Z strings. Measurements are a bit more involved, but worst case may require a Gaussian elimination-like process. For an n -qubit system, Aaronson and Gottesman [3] report an $O(n^2)$ time algorithm, while Anders and Briegel [7] further improve the simulation to $O(n \log n)$ for typical applications using so-called graph states. In Chapter 8, we will revisit this generator array when we look at a matrix decomposition problem for Clifford unitaries.

2.1.7 Application to Error Correction and Fault Tolerance

Error detection and correction, both classical and quantum, work by adding redundancy to allow the possibility of recovery from corrupted data units. The most prominent quantum error correcting codes are *stabilizer codes*, and as the name suggests, stabilizer codes are derived from the stabilizer formalism. The premise is to encode the quantum state that we want to protect into the vector subspace $V_{\mathcal{S}}$ defined by a stabilizer group \mathcal{S} of the Pauli group. An $[[n, k, d]]$ stabilizer code thus encodes k data qubits into n physical qubits and is determined by the $n - k$ independent and commuting generators of \mathcal{S} . The coding space $V_{\mathcal{S}}$ obviously has dimension 2^k ; there must be an n -qubit basis codeword in $V_{\mathcal{S}}$ to every k -qubit basis state. The third attribute d is the *distance* and decides the limits of what the code can do. For example, a code that is able to detect and correct errors on up to t qubits must have minimum distance $2t + 1$. The requirement is less severe for detection only: a code that detects t errors will need minimum distance $t + 1$ [33]. Some of the earliest stabilizer codes are the $[[9, 1, 3]]$ Shor code, $[[7, 1, 3]]$ Steane code, and $[[5, 1, 3]]$ code. The CSS (Calderbank-Shor-Steane) subclass of stabilizer codes is founded by importing classical linear codes carrying certain properties. The encoding and decoding of data qubits depend on stabilizer circuits only.

Once we have an encoded state, we want the ability to perform computations in a manner that minimizes the propagation of errors. Quantum gates that achieve this property are said to be *fault-tolerant*, and the use of stabilizer codes means we can reliably apply Clifford group operations with the least amount of damage. Provided the hardware's error rate is below some preset value specific to the underlying architecture, the *threshold theorem* says arbitrary long quantum computations with fault-tolerant *logical* operations are possible [5]. By logical operations, we mean a unitary that acts on the space of physical qubits but has a different effect on the encoded state. For example,

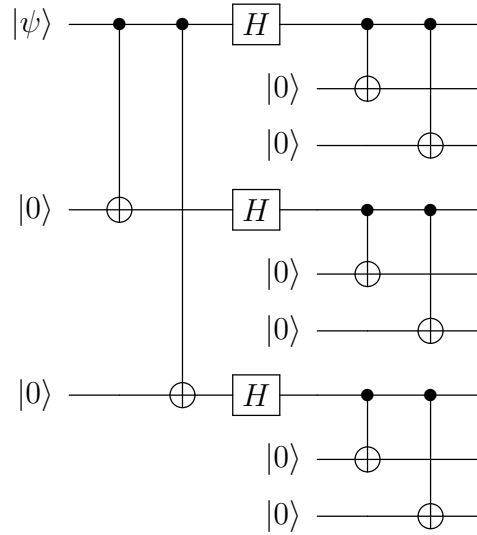


Figure 2.3: Stabilizer circuit to encode a qubit $|\psi\rangle$ into the nine-qubit Shor code [56]. The Shor code can correct any single-qubit bit flip and phase flip error.

$X^{\otimes 7}$ represents the logical bit flip operation for a qubit in the Steane code. Fault tolerance schemes with “high” threshold values around 1% per gate already exist, but they do not come cheaply [31]. As Fowler et al. [31] explains, we may need on the order of 10^4 physical qubits to implement one logical qubit able to withstand the error rates of current devices. We will discover later that even this number is but a small fraction of the physical qubits involved in the overall picture.

2.2 Universal Quantum Computation

In spite of the stabilizer formalism’s many strengths, there is one important hurdle that stabilizer operations are unable to overcome on their own: sufficient conditions for universal quantum computation. The analogous issue arises when we are confronted with only $\{\text{AND}, \text{OR}\}$ operators to construct boolean circuits, which are known to be non-universal. Unfortunately, the power of stabilizer circuits have a lower ceiling than

what we may have initially imagined. To elaborate, Aaronson and Gottesman [3] prove that the problem of simulating stabilizer circuits belongs to a complexity class called $\oplus\mathbf{L}$ (parity-L), and membership in $\oplus\mathbf{L}$ means that stabilizer operations are not universal even for classical computation. Due to this inherent limitation of the stabilizer formalism, we must augment the set of stabilizer operations with non-stabilizer logic devices to enable the construction of more arbitrary quantum circuits. In the same way $\{\text{NOT}, \text{AND}, \text{OR}\}$ creates an adequate set of boolean operations, we may add special quantum gates to form a more diverse *universal gate set* with the Clifford group.

2.2.1 Universal Gate Sets

If we want to stay faithful to Clifford operators for the reasons stated earlier, then we must find the additional quantum gates necessary to achieve UQC. According to Shi [66], not much is required. In particular, any single qubit quantum gate outside the Clifford group $\mathcal{C}(1)$ is sufficient to form a universal gate set with the Controlled-NOT, Hadamard, and Phase gates. The most popular by far in the quantum computing literature is the $\pi/8$ or T gate to form the so-called universal Clifford+ T basis, which we denote

$$\mathcal{G}_T = \{\text{CNOT}, H, T\}. \quad (2.18)$$

Since the T operation is the same as $R_z(\pi/4)$, this means that $T^2 = P$. A preference for the T gate is also not unfounded, as there is a nice construction to fault-tolerantly apply $\pi/4$ rotations on stabilizer encoded qubits [56].

Some other well-known non-Clifford quantum gates are

$$V_1 = \frac{I + 2iX}{\sqrt{5}}, \quad V_2 = \frac{I + 2iY}{\sqrt{5}}, \quad V_3 = \frac{I + 2iZ}{\sqrt{5}} \quad (2.19)$$

to form the universal Clifford+ V basis

$$\mathcal{G}_V = \{\text{CNOT}, H, P, V_1, V_2, V_3\}. \quad (2.20)$$

Sadly, many single qubit non-Clifford gates are not as ideal as the T gate and do not carry the same fault-tolerant properties. But regardless of our selection, we will see momentarily in Subsection 2.2.3 that the price to build non-Clifford gates is significantly higher than that of Clifford operations.

2.2.2 Gate Synthesis

The quest for UQC does not end merely by discovering a universal gate set. If we glance at \mathcal{G}_T (or \mathcal{G}_V), the first thought that comes to mind is that \mathcal{G}_T is finite. Now the problem is readily apparent: the set of unitary operations is uncountable, so a finite universal gate set cannot possibly accommodate every possibility. For instance, the single qubit z -rotation $R_z(\theta)$ alone is parameterized by an angle $\theta \in [0, 2\pi]$. Luckily, we do not have to discard \mathcal{G}_T and everything we know about universal gate sets quite yet. If an exact implementation is out of the question, the next best option is an approximation: we apply a different operation V that is not quite the same as the target unitary U but is fully expressible as a sequence of elements from \mathcal{G}_T . As long as the “error” between the two operations is within an acceptable value ϵ , then we are satisfied with the final result. It is actually in this sense that we say a quantum gate set is universal. The fact that any single qubit rotation can be approximated to arbitrary accuracy by H and T gates alone lies at the core of the universality proof for \mathcal{G}_T [56].

The challenge of approximating U with an implementable V over a universal gate set \mathcal{G} is called the *approximate synthesis problem*. The \mathcal{G} of interest does not have to be \mathcal{G}_T , and often times U takes the form of a 2×2 matrix. This task is sometimes better

understood by splitting the question into two parts. First, given a desired error accuracy ϵ , figure a unitary V decomposable into gates from \mathcal{G} such that

$$D(U, V) \leq \epsilon \tag{2.21}$$

where D is some distance function between two matrices. A frequently used measure is the trace distance defined as

$$D(U, V) = \frac{1}{2} \text{tr} \left[\sqrt{(U - V)^\dagger (U - V)} \right]. \tag{2.22}$$

The second part is to assemble a quantum circuit $V = V_l \cdots V_1$ such that $V_i \in \mathcal{G}$. Both steps are nontrivial, which is why the Solovay-Kitaev theorem [23] is one of the most remarkable results to come out of this study. Informally, it says that if \mathcal{G} is a universal gate set for single qubit quantum gates, then the sequence length l of the approximating matrix scales with $O(\log^{3.97}(1/\epsilon))$. The runtime is similarly $O(\log^{2.71}(1/\epsilon))$.

Ever since, the subject has only grown. Recent years have seen major progressions by incorporating number theoretic ideas, with many paying special attention to the single qubit gate set $\{H, T\}$. To name a few, Kliuchnikov, Maslov, and Mosca (KMM) [47] design an algorithm that uses $O(\log(1/\epsilon))$ gates and two helper or *ancilla* $|0\rangle$ qubits to approximate single qubit unitaries to arbitrary accuracy. The algorithm also executes in $O(\log^2(1/\epsilon) \log \log(1/\epsilon))$ operations. Around the same time, the same individuals KMM release an algorithm for exact synthesis of certain 2×2 unitaries that is optimal in the number H and T gates [48]. Ross and Selinger [63] instead give a solution to approximate single qubit z -rotations that does not require ancilla qubits and uses the fewest number of gates but does depend on a factoring oracle. Other universal bases [11, 62] have also been considered with similar success. Given these advances, we can tell that the focus is

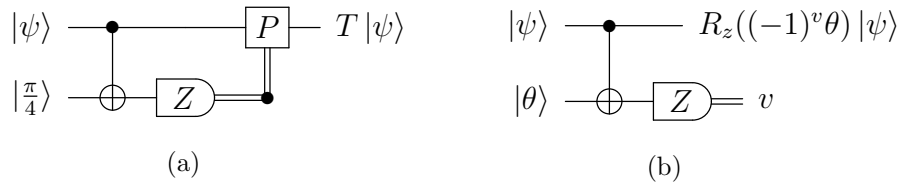


Figure 2.4: (a) The combination of $|\frac{\pi}{4}\rangle$ and a two-qubit stabilizer circuit permits an implementation of the non-Clifford T gate. (b) More general z -rotations on an arbitrary qubit $|\psi\rangle$ are possible by initializing $|\theta\rangle$ such that $0 < \theta < \pi/2$.

no longer limited to generating any approximating sequence of gates, but to return an optimal or near-optimal sequence, especially in regard to non-Clifford gates.

2.2.3 State Distillation and Quantum Resource States

At this point, we know which quantum gates with stabilizer operations enable universal quantum computation, and we know of algorithms to synthesize quantum circuits over a universal basis. All we have left is to pursue some manner of acquiring the chosen non-Clifford gates. The model we consider utilizes an indirect method to achieving such operations using specially prepared qubits e.g. $|H\rangle = \cos(\frac{\pi}{8})|0\rangle + \sin(\frac{\pi}{8})|1\rangle$, the $+1$ eigenstate of the Hadamard gate.

To illustrate this, suppose we initialize a single qubit in the non-stabilizer state

$$|\frac{\pi}{4}\rangle = HP^\dagger |H\rangle = \frac{|0\rangle + e^{i\frac{\pi}{4}}|1\rangle}{\sqrt{2}}. \quad (2.23)$$

Then we may implement the T gate with the simple two-qubit stabilizer circuit in Figure 2.4a. The way in which we carry out the T operation with $|\frac{\pi}{4}\rangle$ generalizes to other angles $0 < \theta < \pi/2$. If we inject the non-stabilizer qubit state

$$|\theta\rangle = \frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}}, \quad (2.24)$$

then depending on the outcome in Figure 2.4b, the effect is either $R_z(\theta)$ or $R_z(-\theta)$; the qubit $|\psi\rangle$ is rotated by $+\theta$ if $v = 0$ and by $-\theta$ otherwise. Small modifications to the Figure 2.4b stabilizer circuit allow us to additionally perform $R_x(\theta)$ and $R_y(\theta)$ operations. This approach to T means an approximation to any single qubit unitary U is possible given a supply of $|H\rangle$ (or $|\frac{\pi}{4}\rangle$) qubits and stabilizer operations. We begin to see reasons why non-stabilizer states may serve as some sort of resource for universal quantum computation, but the previous demonstration does not fully justify this type of special treatment yet. If we have the ability to freely create $|H\rangle$ qubits, then there is no point to valuing $|H\rangle$ or any non-stabilizer state more than $|0\rangle$ and $|1\rangle$.

To understand why non-stabilizer states have such high value, we need to consider the process by which we prepare qubits. In general, state preparation is error-prone, so the initial qubits are usually flawed in some way. Stabilizer states are the exception. The hardware we design is normally set to prepare $|0\rangle$ to good accuracy. Then with the help of stabilizer codes and fault-tolerant Clifford operations, all stabilizer states are attainable to precise levels. On the other hand, the preparation of a non-stabilizer state is more difficult. For this reason, an initial non-stabilizer qubit is typically modeled as a noisy (and hence mixed) state ρ rather than a pure element like $|H\rangle$. We just need to ensure the quality of the initialization process is not too poor, or ρ will end up as a mixture of stabilizer states, which renders the qubit unsuitable for UQC. The border separating the two types is easy to visualize: the six single qubit stabilizer states form an octahedron within the Bloch sphere (see Figure 2.5). Thus if the initial qubit's Bloch vector ends outside the octahedron, we still have a chance at implementing non-Clifford gates, albeit imperfectly. However, if the Bloch vector resides inside the octahedron, the qubit state is no more useful than a stabilizing state.

What is encouraging is that we may employ a technique called state distillation [16] to further improve the non-stabilizer state quality. The premise is quite simple: prepare

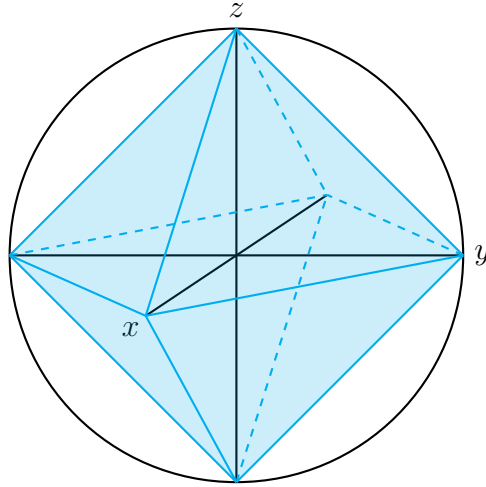


Figure 2.5: Octahedron formed by the six single qubit stabilizer states.

many identical copies of a non-stabilizer mixed qubit ρ , then measure the generators of some error-detecting stabilizer code. If the syndrome measurement indicates no error, then the distillation is successful and the output is a smaller number of higher quality qubits ρ' . Otherwise, we discard the product and start over. We recursively apply the distillation on outputs from previous rounds until the final qubits achieve the desired level of fidelity with the ideal pure state.

As far as we know, current state distillation protocols are only able to target the purification towards a select number of non-stabilizer pure states, which have come to be known as *magic states*. These include $|\theta\rangle$ qubits at angles $\theta = \pi/2^l$ for $l = 2, 3, 4, \dots$ [36], and the eigenstate

$$|K\rangle\langle K| = \frac{1}{2} \left(I + \frac{1}{\sqrt{3}} (X + Y + Z) \right) \quad (2.25)$$

of the Clifford gate $K = e^{i\pi/4}PH$ [16].¹ This may not be obvious, but two $|K\rangle$ qubits are enough to prepare $|\frac{\pi}{6}\rangle$ and hence implement $R_z(\pm\pi/6)$. We should also note that

¹Bravyi and Kitaev [16] define $T = e^{i\pi/4}PH$ and consequently $|T\rangle$. We use $|K\rangle$ to avoid conflicts with the T gate, which is connected to the other magic state $|H\rangle$.

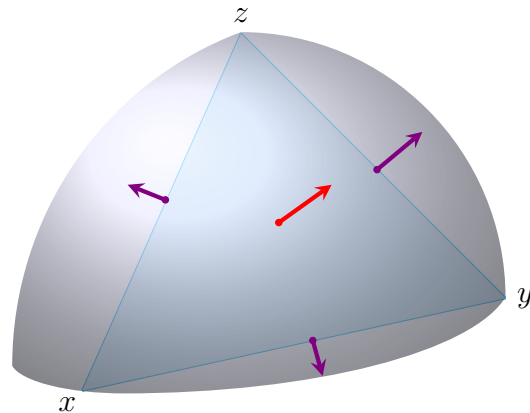


Figure 2.6: The Bloch vector of H -type magic states (purple) go through the edges of the stabilizer octahedron, while the Bloch vector of K -type magic states (red) go through the faces.

given any magic state $|\psi\rangle$, there is no fundamental difference between $|\psi\rangle$ and $G|\psi\rangle$ for realizing UQC, where G is a Clifford gate. Figure 2.6 depicts the relationship between several so-called H -type ($G|H\rangle$) and K -type ($G|K\rangle$) magic states with respect to the stabilizer octahedron. Although any one of these previously mentioned qubits is enough for UQC, Duclos-Cianci and Poulin [25] suggest that utilizing a variety of magic states may be more efficient when trying to handle certain complex transformations. The latest distillation schemes by Haah et al. [36] gives hope that we may someday distill many more non-stabilizer qubits of interest.

The main drawback of state distillation is efficiency. Compared to the physical demand of protecting data qubits, the overhead to deliver universal fault-tolerant quantum computation is much higher. In the magic state model, physical qubits are used in two ways. One set is treated like a quantum register that stores quantum information for later processing. The other set is consumed over the course of performing an actual computation on the register i.e. to implement non-Clifford gates, and magic state distillation comprises a substantial portion of the cost. An analysis by Fowler et al. [31] suggests that with current devices, we must have about 800000 physical qubits ready to generate

one logical $|\frac{\pi}{4}\rangle$. These 800000 physical qubits consist of noisy $|\frac{\pi}{4}\rangle$ qubits to input into the purifier, and stabilizer states to help with the encoding. Moreover, large quantum computations often entail many non-Clifford gates. For example, to fault-tolerantly run Shor's algorithm on a 2000 bit number requires $\sim 10^{12}$ logical $|\frac{\pi}{4}\rangle$ states [31]. In comparison, about 4000 logical qubits and hence $\sim 10^7$ total physical qubits are sufficient for data protection, meaning most of the physical resources are devoted to implementing the algorithm [31]. For this reason, non-stabilizer qubits are treated as precious quantum resources for UQC.

There are two general avenues for improvement: (1) designing more efficient state distillation protocols, and (2) reducing the number of resource states needed during a quantum computation. Research on state distillation has been quite active over the past decade [15, 19, 21, 25, 36, 39, 52], whereas the earlier gate synthesis algorithms address the second area for specific gate sets. In this thesis, we target the second problem but in a more abstract manner. Starting in Chapter 4, we will explore an approach to help manage the cost of some expensive quantum computations.

Chapter 3

Postselected Stabilizer Circuits

The previous chapter introduced stabilizer quantum computation, and the model by which we may obtain UQC with stabilizer operations at the core. For this chapter, we focus on stabilizer circuits with postselected Pauli measurements, or *postselected stabilizer circuits* as we call them. By adding postselection, the circuit output is accepted only when some predetermined measurement values are detected.

We produce some important results regarding postselected stabilizer circuits that we will repeatedly reference throughout this thesis. We identify a useful equivalence relation involving such circuits, and provide a single condition that determines whether any two such circuits belong to the same equivalence class with respect to this relation. Next we look at the consequences of allowing a circuit input to contain a stabilizer qubit e.g. $|0\rangle$ as part of its initial state. We show that such inputs limit the set of circuit outputs that we should expect when we pass it through a postselected stabilizer circuit.

3.1 Notation

We review some notation concerning Pauli operators. We use $X^{(i)}$, $Y^{(i)}$, and $Z^{(i)}$ to mean a Pauli operator that applies a single qubit X , Y , or Z gate on qubit i , and the single qubit identity gate otherwise. For example, if we are told we have an $n = 4$ qubit

system, then $Z^{(2)} = I \otimes Z \otimes I \otimes I$. The symbol I indicates an identity matrix, but the dimensions are not always specified. In some cases, I may be 2×2 , and in others, given parameters n and k , may be $2^{n-k} \times 2^{n-k}$. Its size should be clear from context but will be stated explicitly whenever necessary: we write $I^{\otimes k}$ with a tensor product in the superscript to mean the k -qubit or $2^k \times 2^k$ identity matrix.

3.2 Basic Definitions

We concentrate on stabilizer circuits with arbitrary state inputs, irrespective of the possible applications to magic state distillation. By widening our scope to gain a more abstract understanding, we discover other quantum processes with potential room for resource improvement. For now we start with a few definitions. As a reminder, a Z -measurement on one qubit returns one bit $b \in \{0, 1\}$, and the state of the qubit afterwards is $|b\rangle$. Applying Z -measurements on k qubits returns a length k bit string $v \in \{0, 1\}^k$, so the k measured qubits become $|v\rangle = |v_1 \dots v_k\rangle$.

Definition 3 (Postselected n -to- k Stabilizer Circuit) *A postselected n -to- k stabilizer circuit (C, v) is a quantum circuit that implements an n -qubit Clifford unitary C , followed by Z -measurements on the last $n - k > 0$ qubits for an outcome $v \in \{0, 1\}^{n-k}$.*

The next definition establishes two more related terms.

Definition 4 (Probability and Output) *Let (C, v) be a postselected n -to- k stabilizer circuit and let ρ be an n -qubit state. Then the probability Q_v of outcome v on the transformed state $C\rho C^\dagger$ is*

$$Q_v(C, \rho) = \text{tr} \left((I \otimes \langle v|) C \rho C^\dagger (I \otimes |v\rangle) \right). \quad (3.1)$$

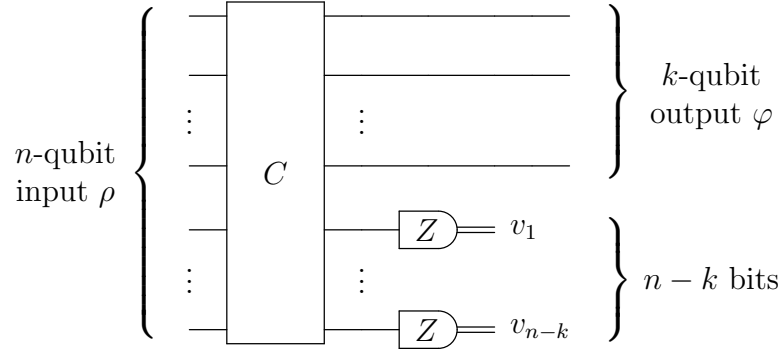


Figure 3.1: Given an n -qubit state ρ , a postselected n -to- k stabilizer circuit (C, v) describes a quantum process in which we apply a Clifford unitary C , measure the last $n - k$ qubits in the Pauli Z basis, then postselect on an outcome $v = v_1 \dots v_{n-k} \in \{0, 1\}^{n-k}$. The unmeasured k -qubit state $\varphi = \Phi_v(C, \rho)$ is the output of the postselected stabilizer circuit on ρ .

If $Q_v(C, \rho) > 0$, then the output Φ_v of the postselected stabilizer circuit (C, v) on ρ is

$$\Phi_v(C, \rho) = \frac{(I \otimes \langle v |) C \rho C^\dagger (I \otimes |v \rangle)}{Q_v(C, \rho)}. \tag{3.2}$$

Figure 3.1 contains the diagram of a postselected n -to- k stabilizer circuit. Measurements on all n qubits is not forbidden, although this is no different than preparing an n -qubit stabilizer state. Notice also in Definition 3 that the circuit concludes with measurements on the *last* $n - k$ qubits. This appears limiting, but a simple argument shows why this assumption is allowed. If we recall from Section 1.6, a SWAP is implemented by three CNOT gates, which implies SWAP is also a Clifford operation. Supposing we permit measurements on any of the qubits, we can always make rearrangements using a combination of SWAPs until the measurements align with the last $n - k$ qubits. Figure 3.2 illustrates this modification for a postselected four-to-two stabilizer circuit. It is easy to see that the state of the unmeasured qubits remains the same in both versions, and the addition of a permutation circuit still leaves a stabilizer circuit.

Having said that, it is perhaps no surprise that different postselected stabilizer circuits are capable of producing the same output on a given input ρ . In the following definition,

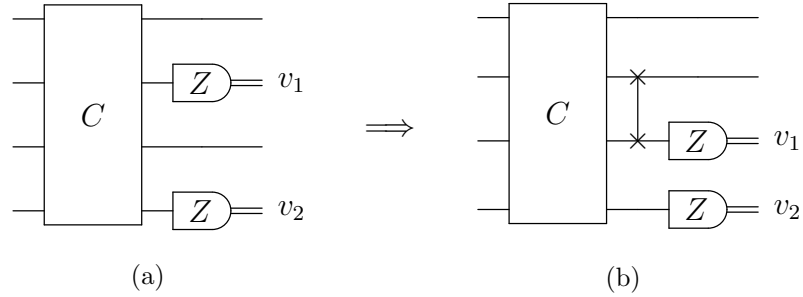


Figure 3.2: We may append a permutation circuit of SWAPs after a Clifford gate C to transform the postselected stabilizer circuit in (a) to the one in (b), where Z -measurements are applied on the last two qubits.

we give two equivalence relations – one weaker and one stronger – that are applicable on the set of postselected stabilizer circuits.

Definition 5 (Clifford Equivalent and Equivalent) *Two postselected n -to- k stabilizer circuits are Clifford equivalent $(C_1, v_1) \sim (C_2, v_2)$ if and only if there is a k -qubit Clifford gate U such that for all n -qubit states ρ , we have the equality*

$$(I \otimes \langle v_1 |) C_1 \rho C_1^\dagger (I \otimes |v_1\rangle) = U (I \otimes \langle v_2 |) C_2 \rho C_2^\dagger (I \otimes |v_2\rangle) U^\dagger. \quad (3.3)$$

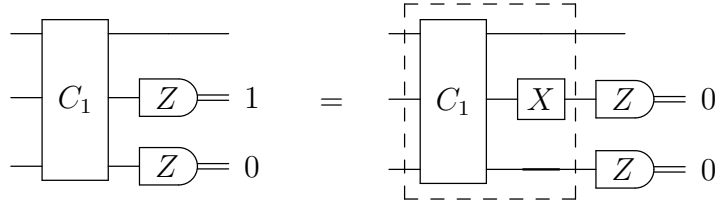
Note that a Clifford equivalence implies that the probabilities of observing v_1 or v_2 are the same i.e. $Q_{v_1}(C_1, \rho) = Q_{v_2}(C_2, \rho)$. We say two postselected stabilizer circuits are simply equivalent, $(C_1, v_1) \equiv (C_2, v_2)$, if and only if $U = I$ in Equation 3.3.

3.3 Properties of Postselected Stabilizer Circuits

We now present some general properties of postselected stabilizer circuits. The first one is obvious and involves just a minor stabilizer circuit manipulation.

Lemma 1 *Let (C_1, v) be a postselected n -to- k stabilizer circuit. Then there exists a Clifford unitary C_2 such that $(C_2, 0^{n-k}) \equiv (C_1, v)$.*

Proof: The proof is straightforward and easiest to explain using circuit diagrams. When $|v_j\rangle = |1\rangle$, we may alter the circuit using $|1\rangle = X|0\rangle$ so that the postselection is on bit value 0. An example of this transformation on a postselected three-to-one stabilizer circuit with $v = 10$ is depicted below:



The dashed box enclosing C_1 and the trailing X gate forms a new Clifford gate C_2 , thus yielding a new postselected stabilizer circuit $(C_2, 0^2) \equiv (C_1, v = 10)$. ■

Hence if we know $(C_1, v_1) \sim (C_2, v_2)$, we may use the same arguments to obtain postselected stabilizer circuits such that $v_1 = v_2$. However if we are only provided (C_1, v_1) and (C_2, v_2) without any prior knowledge of their relationship, then some extra steps are necessary to determine whether the two will be Clifford equivalent. The next proof is inspired from earlier work by Reichardt [61].

Lemma 2 *Let (C_1, v_1) and (C_2, v_2) be postselected n -to- k stabilizer circuits. If*

$$C_1^\dagger (I \otimes |v_1\rangle\langle v_1|) C_1 = C_2^\dagger (I \otimes |v_2\rangle\langle v_2|) C_2, \tag{3.4}$$

then $(C_1, v_1) \sim (C_2, v_2)$.

Proof: Without loss of generality, assume $v_1 = v_2 = 0^{n-k}$ by Lemma 1. We start by simply rewriting Equation 3.3, that for all n -qubit states ρ , there is a k -qubit Clifford gate U such that

$$C_1 \Pi_1 \rho \Pi_1 C_1^\dagger = (U \otimes I^{\otimes n-k}) C_2 \Pi_2 \rho \Pi_2 C_2^\dagger (U^\dagger \otimes I^{\otimes n-k}) \tag{3.5}$$

where $\Pi_1 = C_1^\dagger(I^{\otimes k} \otimes |0^{n-k}\rangle\langle 0^{n-k}|)C_1$ and $\Pi_2 = C_2^\dagger(I^{\otimes k} \otimes |0^{n-k}\rangle\langle 0^{n-k}|)C_2$ are projection operators. Equation 3.5 is true if $\Pi_1 = \Pi_2$ (which we now refer to as Π), and we instantly recognize that Π is a product of $n - k$ projectors

$$\Pi = \frac{1}{2^{n-k}} \prod_{i=1}^{n-k} (I^{\otimes n} + s_i) \quad (3.6)$$

where $\{s_1, \dots, s_{n-k}\}$ are independent and commuting n -qubit Pauli matrices that generate a stabilizer group $\mathcal{S} = \langle s_1, \dots, s_{n-k} \rangle$. This assertion stems from

$$I^{\otimes k} \otimes |0^{n-k}\rangle\langle 0^{n-k}| = \frac{1}{2^{n-k}} \prod_{i=1}^{n-k} (I^{\otimes n} + Z^{(k+i)}), \quad (3.7)$$

meaning $\mathcal{S} = \langle C_1^\dagger Z^{(k+1)} C_1, \dots, C_1^\dagger Z^{(n)} C_1 \rangle = \langle C_2^\dagger Z^{(k+1)} C_2, \dots, C_2^\dagger Z^{(n)} C_2 \rangle$.

Since we can expand any n -qubit density matrix ρ as a linear combination of 4^n Pauli operators with real coefficients, only those terms in the expansion that commute with all $n - k$ generators s_i will survive the projection by Π . The reason is as follows. If a matrix $g \in \mathcal{P}(n)$ commutes with a generator s_i , then

$$\left(\frac{I^{\otimes n} + s_i}{2} \right) g \left(\frac{I^{\otimes n} + s_i}{2} \right) = \left(\frac{I^{\otimes n} + s_i}{2} \right) \left(\frac{I^{\otimes n} + s_i}{2} \right) g = \left(\frac{I^{\otimes n} + s_i}{2} \right) g \quad (3.8)$$

where the change from the middle to the right expression is due to idempotence. On the other hand, any anticommuting $g \in \mathcal{P}(n)$ leads to cancellation:

$$\left(\frac{I^{\otimes n} + s_i}{2} \right) g \left(\frac{I^{\otimes n} + s_i}{2} \right) = \left(\frac{I^{\otimes n} + s_i}{2} \right) \left(\frac{I^{\otimes n} - s_i}{2} \right) g = 0. \quad (3.9)$$

As such, we define \mathcal{P}_k to be the set of n -qubit Pauli operators $g = g_1 \otimes \dots \otimes g_k \otimes I^{\otimes n-k}$ such that $g_i \in \{I, X, Y, Z\}$. Let $\mathcal{P}'_k = \{C_1^\dagger g C_1 \mid g \in \mathcal{P}_k\}$. Consequently, each element $h \in \mathcal{P}'_k$ belongs to the normalizer of \mathcal{S} and commutes with every $s \in \mathcal{S}$. In this manner,

if $h = C_1^\dagger g C_1$ for some $g \in \mathcal{P}_k$, then we obtain

$$C_1 \left(\sum_{s \in \mathcal{S}} sh \right) C_1^\dagger = g_1 \otimes \cdots \otimes g_k \otimes |0^{n-k}\rangle\langle 0^{n-k}|. \quad (3.10)$$

For each operator $h \in \mathcal{P}'_k$, we define coefficients

$$c_h = \text{tr} \left(\sum_{s \in \mathcal{S}} sh \rho \right). \quad (3.11)$$

We are ready to see how Π affects ρ . When we apply the projection, the linear expansion turns into

$$\Pi \rho \Pi = \left(\frac{1}{2^{n-k}} \sum_{s \in \mathcal{S}} s \right) \rho \left(\frac{1}{2^{n-k}} \sum_{s \in \mathcal{S}} s \right) = \frac{1}{2^{3n-2k}} \sum_{h \in \mathcal{P}'_k} c_h \left(\sum_{s \in \mathcal{S}} sh \right). \quad (3.12)$$

We know beforehand that for each $h \in \mathcal{P}'_k$, that there is a matrix $g \in \mathcal{P}_k$ such that $C_1 h C_1^\dagger = g$. We have a similar occurrence with C_2 . For each $h \in \mathcal{P}'_k$, there is an $h' \in \mathcal{S}h = \{sh \mid s \in \mathcal{S}\}$ and $g' \in \mathcal{P}_k$ such that either $C_2 h' C_2^\dagger = g'$ or $C_2 h' C_2^\dagger = -g'$. This implies an appropriate k -qubit Clifford unitary U exists to satisfy Equation 3.5. To find such a desired gate U , we only have to track the $2k$ elements $r_1, \dots, r_k, t_1, \dots, t_k \in \mathcal{P}'_k$ such that $C_1 r_i C_1^\dagger = X^{(i)}$ and $C_1 t_i C_1^\dagger = Z^{(i)}$. Then we select U that fulfills

$$(U \otimes I^{\otimes n-k}) C_2 r_i C_2^\dagger (U^\dagger \otimes I^{\otimes n-k}) = X^{(i)} \quad (3.13)$$

$$(U \otimes I^{\otimes n-k}) C_2 t_i C_2^\dagger (U^\dagger \otimes I^{\otimes n-k}) = Z^{(i)} \quad (3.14)$$

for all r_i and t_i . The value of $C_1 h C_1^\dagger = (U \otimes I^{\otimes n-k}) C_2 h C_2^\dagger (U^\dagger \otimes I^{\otimes n-k})$ for the other $h \in \mathcal{P}'_k$ is preset given the $2k$ associations above. Since the unnormalized postmeasurement qubits are the same after C_1 and $(U \otimes I^{\otimes n-k}) C_2$, we deduce that $(C_1, 0^{n-k}) \sim (C_2, 0^{n-k})$, or more generally, that $(C_1, v_1) \sim (C_2, v_2)$. ■

We may expand on Lemma 2 and show that extra $|0\rangle$ qubits do not add to the power of a postselected stabilizer circuit. The original result attributed to Reichardt [61], and Campbell and Browne [18] applies to postselected n -to-one stabilizer circuits. We give a statement that covers the more general n -to- k .

Lemma 3 *Let ρ be an n -qubit state. Suppose a postselected $(n+1)$ -to- n stabilizer circuit (C_1, v) produces a potentially unnormalized n -qubit density matrix*

$$\rho' = (I \otimes \langle v|)C_1 (\rho \otimes |0\rangle\langle 0|) C_1^\dagger (I \otimes |v\rangle) \quad (3.15)$$

with nonzero probability $\text{tr}(\rho') > 0$. Then there exists an n -qubit Clifford unitary C_2 such that one of the following holds: (1) $\rho' = C_2 \rho C_2^\dagger$, or (2) $\rho' = C_2 (\rho'_1 \otimes |v'\rangle\langle v'|) C_2^\dagger$, where ρ'_1 is an unnormalized $(n-1)$ -qubit density state and $v' \in \{0, 1\}$.

Proof: Let $(-1)^v g = C_1^\dagger Z^{(n+1)} C_1$ be an $(n+1)$ -qubit Pauli operator. Then applying C_1 and postselecting on v is the same as performing a projection, then applying C_1 :

$$\rho' \otimes |v\rangle\langle v| = C_1 \left(\frac{I^{\otimes n+1} + g}{2} \right) (\rho \otimes |0\rangle\langle 0|) \left(\frac{I^{\otimes n+1} + g}{2} \right) C_1^\dagger. \quad (3.16)$$

We will show, as seen in [61], that we may eliminate the projection with g , or convert it to a projection on the first n qubits only. There are essentially three possibilities:

1. If $g = Z^{(n+1)}$, then the projection has no effect. The other case $g = -Z^{(n+1)}$ is not possible by $\text{tr}(\rho') > 0$. Then for each n -qubit $h \in \{X^{(1)}, \dots, X^{(n)}, Z^{(1)}, \dots, Z^{(n)}\}$, there is an n -qubit Pauli operator h' such that

$$C_1(h \otimes I)C_1^\dagger = h' \otimes I \text{ and } C_1(h \otimes Z)C_1^\dagger = (-1)^v h' \otimes Z, \text{ or} \quad (3.17)$$

$$C_1(h \otimes Z)C_1^\dagger = h' \otimes I \text{ and } C_1(h \otimes I)C_1^\dagger = (-1)^v h' \otimes Z. \quad (3.18)$$

This implies there is a separate n -qubit Clifford unitary C_2 that also obeys the above equalities $C_2 h C_2^\dagger = h'$ for all h . For this reason, we may eliminate the projection and replace C_1 with $C_2 \otimes X^v$.

2. If $g = g_1 \otimes \cdots \otimes g_{n+1}$ anticommutes with $Z^{(n+1)}$, then $g_{n+1} \in \{\pm X, \pm Y\}$. Suppose $g_{n+1} = X$ without loss of generality. Let $h = g_1 \otimes \cdots \otimes g_n$, and $U = \Lambda^{(n+1)}(h)$ be a Clifford gate that applies h controlled on qubit $n + 1$. According to [61], we have

$$\frac{I^{\otimes n+1} + g}{2} = U^\dagger \left(I^{\otimes n} \otimes \frac{I + X}{2} \right) U. \quad (3.19)$$

It is not difficult to see why Equation 3.19 holds. Seeing how

$$\text{CZ} = (\text{CZ})^\dagger, \quad \text{CZ} = (I \otimes H) \text{CNOT} (I \otimes H), \quad (3.20)$$

$$\text{CY} = (\text{CY})^\dagger, \quad \text{CY} = (I \otimes P) \text{CNOT} (I \otimes P^\dagger), \quad (3.21)$$

we may use our existing knowledge about CNOT to realize

$$\text{CZ} (X \otimes Z) \text{CZ} = X \otimes I \quad (3.22)$$

$$\text{CY} (X \otimes Y) \text{CY} = X \otimes I. \quad (3.23)$$

However, the initial U has no effect because the last qubit is $|0\rangle$, so we project $|0\rangle$ onto $|+\rangle$ and end with

$$\rho' \otimes |v\rangle\langle v| = C_1 U^\dagger (\rho \otimes |+\rangle\langle +|) U C_1^\dagger. \quad (3.24)$$

From here, we use similar reasoning from Case 1 to replace $C_1 U^\dagger$ with $C_2 \otimes HZ^v$, or simply remove the projection and C_1 altogether and proceed with $C_2 \otimes X^v$.

3. The g matrix is either $h \otimes I$ or $h \otimes Z$, where $h \neq I^{\otimes n}$ is an n -qubit Pauli operator.

Assume $g = h \otimes I$ first. Then there are n -qubit Clifford gates C_2 and C_3 so that

$$C_1 = \left(C_2 \otimes I \right) \left(I^{\otimes n-1} \otimes \text{SWAP}^{(n,n+1)} \right) \left(C_3 \otimes I \right) \quad (3.25)$$

and $C_3^\dagger Z^{(n)} C_3 = (-1)^v h$. From here, we know

$$C_3 \left(\frac{I^{\otimes n} + h}{2} \right) = \left(I^{\otimes n-1} \otimes |v\rangle\langle v| \right) C_3 \quad (3.26)$$

and we thus have

$$\left(I^{\otimes n-1} \otimes |v\rangle\langle v| \right) C_3 \rho C_3^\dagger \left(I^{\otimes n-1} \otimes |v\rangle\langle v| \right) = \rho'_1 \otimes |v\rangle\langle v|. \quad (3.27)$$

The $\text{SWAP}^{(n,n+1)}$ gate moves $|v\rangle$ in Equation 3.27 to position $n+1$ and brings $|0\rangle$ to position n , achieving $\rho' = C_2 (\rho'_1 \otimes |0\rangle\langle 0|) C_2^\dagger$. The other case is similar except we insert a CNOT before SWAP:

$$C_1 = \left(C_2 \otimes I \right) \left(I^{\otimes n-1} \otimes \text{CNOT}^{(n,n+1)} \right) \left(I^{\otimes n-1} \otimes \text{SWAP}^{(n,n+1)} \right) \left(C_3 \otimes I \right). \quad (3.28)$$

Projecting with $h \otimes Z$ on $\rho \otimes |0\rangle\langle 0|$ has the same effect as projecting with $h \otimes I$, so we basically still apply $\left(I^{\otimes n-1} \otimes |v\rangle\langle v| \right) C_3$ like before. Since qubit n is $|0\rangle$ after the SWAP gate, the CNOT changes nothing, and the outcome is the same.

Therefore one of the two conditions hold. ■

A trivial example of the second scenario in Lemma 3 is $C_1 = I^{\otimes n-1} \otimes \text{SWAP}^{(n,n+1)}$. This suggests we may exchange (C_1, v) for a postselected n -to- $(n-1)$ stabilizer circuit to generate ρ'_1 from ρ . Starting in Chapter 4, we further examine postselected stabilizer circuits that yield a single qubit output.

3.4 Summary

We proved some nice features about postselected stabilizer circuits. The Clifford equivalence condition of Lemma 2 is especially helpful when we take a detailed look at two-qubit stabilizer circuits in the next chapter.

Chapter 4

Two-qubit Stabilizer Circuits with Recovery

We have so far mentioned some general properties of postselected stabilizer circuits, but none of which offer anything specific to help us resolve our resource concerns. That being said, let us now focus exclusively on two-qubit stabilizer circuits with one Z -measurement, the smallest quantum circuit available for producing qubits not possible by Clifford unitaries only. Because the set of stabilizer states is closed under Clifford group operations, we furthermore assume circuit inputs of non-stabilizer qubits only to produce other, more diverse non-stabilizer qubits. Our intent is to explore these processes from a different angle, outside the realm of state distillation, and simply examine their behavior on more arbitrary non-stabilizer inputs. Only by understanding how a stabilizer circuit responds to such qubits may we gain better insight into formulating an effective strategy for resource management. We will soon discover that despite limiting the problem size to a mere two qubits, we encounter some encouraging ideas that are worth pursuing in larger settings.

With this in mind, we start by refining some implementation details initially provided by Reichardt [61] to identify three configurations characterizing all postselected two-to-one stabilizer circuits. These three forms suggest that in addition to Pauli measurements

and postselection, single qubit Clifford gates and at most one CNOT or SWAP are enough to realize any such procedure acting on two qubits. When the input set is further confined to certain product states, we discover an interesting connection between stabilizer circuits of the single CNOT variety: there are “recovery circuits” that can recuperate a product state input qubit from a corrupted stabilizer circuit output qubit. At the end of the chapter, we prove Theorem 3, which informally states the following: *any postselected two-to-one stabilizer circuit (C, v) realizable by one CNOT has recovery circuits, and that all such recovery circuits of (C, v) are equivalent to one-and-another.*

4.1 Notation and Conventions

Before continuing, we establish some conventions for use here and in the few chapters ahead. Given the scale of our circuits, we let I stand for the single qubit identity, $\text{CNOT} = \text{CNOT}^{(1,2)}$, and $\text{SWAP} = \text{SWAP}^{(1,2)}$. Once we start the discussion on recovery circuits, we use the symbol ψ as a convenient substitute for the density matrix $|\psi\rangle\langle\psi|$. This results in more elegant notation overall. Given a postselected two-to-one stabilizer circuit (C, v) , at times we may say *run circuit C* , which translates to an application of the unitary C on a two-qubit input ρ , followed by one Z -measurement on the second qubit. This is often followed by details on what course of action to take condition on v (or otherwise not v). The term *circuit C* thus references the stabilizer circuit piece only of the postselected circuit, including the measurement at the end.

4.2 Postselected Two-to-One Stabilizer Circuits

While there are many two-qubit Clifford gates ($|\mathcal{C}(2)| = 11520$) relative to the input size ($n = 2$), the number of actual circuits we need to consider is 30 [61]. The reason

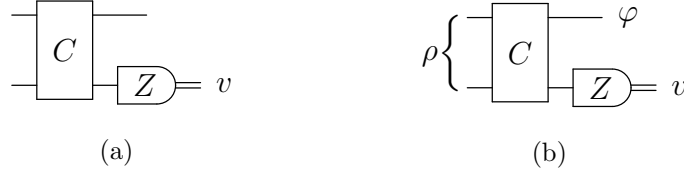


Figure 4.1: (a) A postselected two-to-one stabilizer circuit (C, v) consists of a stabilizer circuit component C that ends with one Z -measurement, and a postselected bit value v . (b) The qubit $\varphi = \Phi_v(C, \rho)$ is the output of a postselected two-to-one stabilizer circuit (C, v) on the two-qubit input ρ .

is a natural consequence of Lemma 2. More precisely, it states that two postselected stabilizer circuits (C_1, v_1) and (C_2, v_2) are Clifford equivalent if

$$C_1^\dagger (I \otimes |v_1\rangle\langle v_1|) C_1 = \Pi = C_2^\dagger (I \otimes |v_2\rangle\langle v_2|) C_2. \quad (4.1)$$

Since $|v_1| = |v_2| = 1$ in this situation, our operator Π involves exactly one projector:

$$\Pi = \frac{I \otimes I + s}{2} \quad (4.2)$$

where s is a two-qubit Pauli operator. We have 15 unique possibilities for s , and 30 different two-to-one circuits total after accounting for the bit. As such, we can introduce three forms in the following theorem to represent all two-to-one circuits (C, v) .

Theorem 2 (van Dam and Wong [69], Lemma 4) *For each postselected two-to-one stabilizer circuit (C, v) , there are single qubit Clifford gates G_1 and G_2 such that either*

1. $(C, v) \sim (I \otimes G_1, 0)$
2. $(C, v) \sim ((I \otimes G_1)\text{SWAP}, 0)$
3. $(C, v) \sim (\text{CNOT}(G_1 \otimes G_2), 0)$.

Proof: First, a Clifford equivalence $(C_1, v_1) \sim (C_2, v_2)$ is invariant with respect to Clifford circuits that execute prior to the gates C_1 and C_2 i.e. $(C_1, v_1) \sim (C_2, v_2)$ if and only if $(C_1U, v_1) \sim (C_2U, v_2)$ for any Clifford unitary U . We partition the 15 nontrivial Pauli operators into the following sets:

$$\mathcal{P}_A = \{g_1 \otimes g_2 \mid g_1, g_2 \in \{X, Y, Z\}\} \quad (4.3)$$

$$\mathcal{P}_B = \{I \otimes X, I \otimes Y, I \otimes Z\} \quad (4.4)$$

$$\mathcal{P}_C = \{X \otimes I, Y \otimes I, Z \otimes I\}. \quad (4.5)$$

We look at $g = Z \otimes Z$ first. Suppose there is a bit v' such that

$$CgC^\dagger = (-1)^{v'} I \otimes Z. \quad (4.6)$$

Knowing $\text{CNOT}(Z \otimes Z)\text{CNOT}^\dagger = I \otimes Z$, we obtain

$$(C, v) \sim (\text{CNOT}, v + v' \bmod 2) \quad (4.7)$$

by Lemma 2. For the remaining two-qubit Pauli operators $g \in \mathcal{P}_A$, suppose $CgC^\dagger = \pm I \otimes Z$. Choose single qubit Clifford gates G_1 and G_2 such that

$$(G_1 \otimes G_2)g(G_1^\dagger \otimes G_2^\dagger) = Z \otimes Z. \quad (4.8)$$

Define $C'' = C(G_1^\dagger \otimes G_2^\dagger)$. Then $C''(Z \otimes Z)C''^\dagger = (-1)^{v'} I \otimes Z$ for some bit v' . The rest follows from previous arguments to conclude

$$(C''(G_1 \otimes G_2), v) = (C, v) \sim (\text{CNOT}(G_1 \otimes G_2), v + v' \bmod 2). \quad (4.9)$$

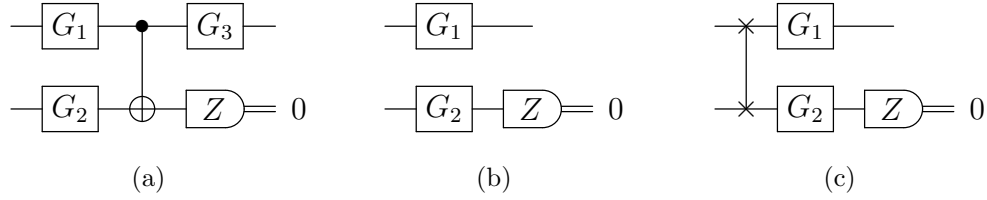


Figure 4.2: Any postselected two-to-one stabilizer circuit (C, v) may be represented by another resembling circuit (a), circuit (b), or circuit (c). The choice of single qubit Clifford gates G_1 , G_2 , and G_3 depend on the Clifford gate C and bit value v .

For the operator $I \otimes Z \in \mathcal{P}_B$, assume $C(I \otimes Z)C^\dagger = (-1)^{v'}I \otimes Z$. Then $(C, v) \sim (I \otimes I, v + v' \bmod 2)$. Coverage of the remaining five Pauli operators from \mathcal{P}_B and \mathcal{P}_C is similar to the above.

To finish, suppose $(C, v) \sim (I \otimes G, v + v' \bmod 2)$, where G is a single qubit Clifford gate. If $v + v' \bmod 2 = 1$, then $(C, v) \sim (I \otimes G, 1) \equiv (I \otimes XG, 0)$. The same reasoning applies when $(C, v) \sim ((I \otimes G)\text{SWAP}, 1)$. If $(C, v) \sim (\text{CNOT}(G_1 \otimes G_2), 1)$, then we include $(I \otimes X)\text{CNOT}(G_1 \otimes G_2) = \text{CNOT}(G_1 \otimes XG_2)$. The other case $v + v' \bmod 2 = 0$ follows directly from Lemma 2. \blacksquare

Corollary 1 *Let $(C_{eq}, v_{eq}) \sim (C, v)$. Then $(C, 1 - v)$ is Clifford equivalent to a slightly modified version (C'_{eq}, v'_{eq}) of (C_{eq}, v_{eq}) :*

$(C_{eq}, v_{eq}) \sim (C, v)$	$(C'_{eq}, v'_{eq}) \sim (C, 1 - v)$	(4.10)
$(I \otimes G_1, 0)$	$(I \otimes XG_1, 0)$	
$((I \otimes G_1)\text{SWAP}, 0)$	$((I \otimes XG_1)\text{SWAP}, 0)$	
$(\text{CNOT}(G_1 \otimes G_2), 0)$	$((I \otimes X)\text{CNOT}(G_1 \otimes G_2), 0)$	

Due to Theorem 2, we have a remarkably much easier time studying two-to-one circuits. We may substitute (C, v) with another that likely uses fewer gates but behaves in exactly the same way. Because there are many identities on Pauli operators and

Clifford gates, G_1 and G_2 are not unique. For example,

$$((\text{CNOT}(Z \otimes I), 0) \equiv ((Z \otimes I)\text{CNOT}, 0) \sim (\text{CNOT}, 0). \quad (4.11)$$

Of the 30 postselected circuits that we observed, it is easy to see that there are 18 varieties of $(\text{CNOT}(G_1 \otimes G_2), 0)$, and 6 each for $(I \otimes G_1, 0)$ and $((I \otimes G_1)\text{SWAP}, 0)$. If we want to separate the circuits by the stricter kind of equivalence “ \equiv ”, the number of classes is multiplied by 24 e.g. $18 \cdot 24 = 432$ for $((G_3 \otimes I)\text{CNOT}(G_1 \otimes G_2), 0)$, since there are $|\mathcal{C}(1)| = 24$ choices of G_3 .

4.3 Two-qubit Recovery Circuits

For any quantum circuit involving measurements, there is usually one subset of outcomes that is preferred more than others. If we are less than fortunate, convention dictates that we discard the output and rerun the circuit on new input instances until we succeed. This is not much of an issue when the overhead to prepare more initial state copies is low, but can become problematic otherwise. If the cost associated with state preparation is a barrier to large computations, any method that alleviates this burden is highly desirable. It turns out when our two-qubit input ρ is a tensor product state, i.e. $\rho = \varphi \otimes |\psi\rangle\langle\psi|$, we have an alternative: there exist operations capable of reusing an undesirable output to try and recovery φ .

This input selection also means the only configuration worth considering from Theorem 2 is $(\text{CNOT}(G_1 \otimes G_2), 0)$; the other two arrangements lead to rather trivial outputs. We can easily see that when $(C, v) \sim (I \otimes G_1, 0)$, the output of (C, v) on $\varphi_1 \otimes \varphi_2$ is essentially φ_1 . The output is always an input, and the same is similarly true for all postselected circuits $(C, v) \sim ((I \otimes G_1)\text{SWAP}, 0)$.

Definition 6 (Interacting Postselected Stabilizer Circuit) *A postselected two-to-one stabilizer circuit (C, v) is interacting if and only if there are single qubit Clifford gates G_1 and G_2 such that $(C, v) \sim (\text{CNOT}(G_1 \otimes G_2), 0)$. We say circuit C is interacting if and only if $(C, 0)$ is interacting.*

With that, we define the notion of a recovery circuit.

Definition 7 (Recovery Circuit) *Let (C, v) be an interacting postselected stabilizer circuit. Then a postselected two-to-one stabilizer circuit (C', v') is a recovery circuit of (C, v) if and only if for all two-qubit states $\varphi \otimes \psi$, we have*

$$\varphi = \Phi_{v'}(C', \Phi_{1-v}(C, \varphi \otimes \psi) \otimes \psi). \quad (4.12)$$

Notice that the first input qubit to (C', v') is the output of $(C, 1-v)$ on $\varphi \otimes \psi$. In this context, since we are targeting v instead of $1-v$, then we say circuit C is *successful* upon measuring v on the second qubit of $C(\varphi \otimes \psi)C^\dagger$. Otherwise circuit C is *unsuccessful*, and the recovery circuit provides a second chance at obtaining the output of (C, v) on $\varphi \otimes \psi$. Assuming $|\psi\rangle$ is relatively cheaper to prepare than φ , the presumption is that an implementation of C' is much simpler to pursue than the initialization process of φ . Our next lemma presents one way on how to design such a recovery circuit to (C, v) .

Lemma 4 *Every interacting postselected stabilizer circuit (C, v) has a recovery circuit.*

Proof: Let $(C, v) \sim (\text{CNOT}(G_1 \otimes G), 0)$, where G_1 and G are single qubit Clifford gates. By Corollary 1, we know

$$(C, 1-v) \sim ((I \otimes X)\text{CNOT}(G_1 \otimes G), 0) \equiv (\text{CNOT}(G_1 \otimes G), 1) \quad (4.13)$$

which means there is a single qubit Clifford gate G_2 such that

$$(C, 1 - v) \equiv ((G_2^\dagger \otimes I)\text{CNOT}(G_1 \otimes G), 1). \quad (4.14)$$

We shall show that $((G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G), 0)$ is a recovery circuit of (C, v) . Figure 4.3 includes reference diagrams to aid comprehension.

If $\varphi_1 \otimes \psi$ is the initial state, consider $\varphi'_1 \otimes \psi' = G_1 \varphi_1 G_1^\dagger \otimes G \psi G^\dagger$. Let (x_1, y_1, z_1) be the Bloch vector of φ'_1 and (x, y, z) be the Bloch vector of $|\psi'\rangle$. For ease of notation, we define symbols

$$\varphi'_2 = \Phi_1(\text{CNOT}, \varphi'_1 \otimes \psi') \quad (4.15)$$

$$\varphi_2 = G_2^\dagger \varphi'_2 G_2 = \Phi_{1-v}(C, \varphi_1 \otimes \psi). \quad (4.16)$$

Then the Bloch vector (x_2, y_2, z_2) of φ'_2 becomes

$$x_2 = \frac{x_1 x + y_1 y}{1 - z_1 z}, \quad y_2 = \frac{y_1 x - x_1 y}{1 - z_1 z}, \quad z_2 = \frac{z_1 - z}{1 - z_1 z}. \quad (4.17)$$

Now suppose $\varphi_3 = \Phi_0(\text{CNOT}(G_2 \otimes G), \varphi_2 \otimes \psi)$. For postselected stabilizer circuits that basically apply a single CNOT, the equations for computing the output's Bloch vector are essentially the same:

$$x_3 = \frac{x_2 x - y_2 y}{1 + z_2 z}, \quad y_3 = \frac{y_2 x + x_2 y}{1 + z_2 z}, \quad z_3 = \frac{z_2 + z}{1 + z_2 z}, \quad (4.18)$$

where (x_3, y_3, z_3) represents the Bloch vector of φ_3 . Using $x^2 + y^2 + z^2 = 1$, we can show

$$x_3 = \frac{x_1 x^2 + x y_1 y - x y_1 y + x_1 y^2}{1 - z_1 z + z_1 z - z^2} = x_1. \quad (4.19)$$

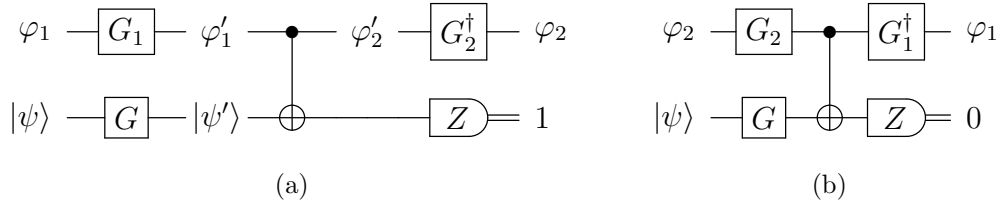


Figure 4.3: Suppose $(C, 1-v) \equiv ((G_2^\dagger \otimes I)\text{CNOT}(G_1 \otimes G), 1)$. This equivalence allows us to study $(C, 1-v)$ via its substitute in (a) and come up with the recovery circuit in (b). We include intermediate states like φ'_1 and $\varphi'_2 = G_2\varphi_2G_2^\dagger$ in (a) to signify stages in the circuit.

Likewise, $y_3 = y_1$ and $z_3 = z_1$, which means $\varphi_3 = \varphi'_1 = G_1\varphi_1G_1^\dagger$. The postselected stabilizer circuit $((G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G), 0)$ is therefore a recovery circuit of (C, v) . ■

Between (C, v) and its recovery circuit $(C', 0)$, where $C' = (G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G)$, there is a relatively straightforward relationship between the probability that circuit C would have been successful and the probability that circuit C' will be successful.

Corollary 2 *Let $\varphi_1 \otimes \psi$ be a two-qubit state and let $C' = (G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G)$ be a two-qubit Clifford unitary such that $(C', 0)$ is a recovery circuit of an interacting postselected stabilizer circuit (C, v) . Then*

$$Q_0(C', \Phi_{1-v}(C, \varphi_1 \otimes \psi) \otimes \psi) = \frac{(1 - z^2)/4}{1 - Q_v(C, \varphi_1 \otimes \psi)} \quad (4.20)$$

where $z = \langle \psi | G^\dagger Z G | \psi \rangle$.

Proof: We assume for simplicity that $C = \text{CNOT}$ and $v = 0$, which implies $G_1 = G_2 = G = I$. Let $z_1 = \text{tr}(Z\varphi_1)$ and $z = \langle \psi | Z | \psi \rangle$. Also let $\varphi_2 = \Phi_1(C, \varphi_1 \otimes \psi)$. Then the probability of not observing $v = 0$ is

$$Q_1(C, \varphi_1 \otimes \psi) = \frac{1 - z_1 z}{2}. \quad (4.21)$$

From what we saw in Lemma 4, we know that the Bloch vector z -component of φ_2 is

$$z_2 = \text{tr}(Z\varphi_2) = \frac{z_1 - z}{1 - z_1 z}. \quad (4.22)$$

The probability of recovering φ_1 is now clear:

$$Q_0(C', \varphi_2 \otimes \psi) = \frac{1 + z_2 z}{2} = \frac{1 - z_1 z + z_1 z - z^2}{4 \left(\frac{1 - z_1 z}{2}\right)} \quad (4.23)$$

$$= \frac{(1 - z^2)/4}{1 - Q_0(C, \varphi_1 \otimes \psi)} \quad (4.24)$$

since the circuits perform a single measurement. ■

Another implication of the proof to Lemma 4 is that $\Phi_{1-v}(C, \varphi_1 \otimes \psi)$ is the same as φ_1 , up to a single qubit Clifford gate, whenever $|\psi\rangle$ is an eigenstate of the Pauli X , Y , or Z matrix (a stabilizer qubit). This is simply a special case of Lemma 3 from Chapter 3. Since the behavior of (C, v) under these circumstances is no different than non-interacting circuits, it does not warrant the use of circuit $C' = (G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G)$ to try and perform a recovery. It is also quite evident by now that there is only one type of recovery circuit, especially given our construction in Lemma 4.

Lemma 5 *All recovery circuits are interacting postselected stabilizer circuits.*

Proof: Let (C, v) be an interacting postselected stabilizer circuit. Suppose (C', v') is a recovery circuit of (C, v) but is not interacting. Then (C', v') is Clifford equivalent to some postselected stabilizer circuit $(I \otimes G, 0)$ or $((I \otimes G)\text{SWAP}, 0)$, where G is a single qubit Clifford gate. In both cases, we can easily find a two-qubit state $\varphi \otimes \psi$ such that (C', v') fails to recover φ on the input $\Phi_{1-v}(C, \varphi \otimes \psi) \otimes \psi$. ■

We should mention that Lemma 4 promises existence of a recovery circuit but gave no guarantees about uniqueness. Moving forward, we want to prove that all recovery circuits

of an interacting postselected circuit (C, v) are equivalent with one another. This way, we dispel concerns that one amongst many has a better chance of succeeding than the others. We start by showing a basic fact about any two recovery circuits of (C, v) .

Lemma 6 *Let (C_1, v_1) and (C_2, v_1) be recovery circuits of an interacting postselected stabilizer circuit (C, v) . Then*

$$C_1^\dagger(I \otimes |v_1\rangle\langle v_1|)C_1 = C_2^\dagger(I \otimes |v_2\rangle\langle v_2|)C_2. \quad (4.25)$$

Proof: It is easier to prove the contrapositive: we show the recovery by (C_2, v_2) will fail on one particular pair of qubits φ_2 and $|\psi\rangle$, although many exists that are equally as good. Let

$$\Pi_2 = C_2^\dagger(I \otimes |v_2\rangle\langle v_2|)C_2 = \frac{I \otimes I + s}{2} \quad (4.26)$$

be a projection operator, where $s = s_1 \otimes s_2 \in \mathcal{P}(2)$ is a two-qubit Pauli operator with an overall factor ± 1 , and neither $s_1 = I$ nor $s_2 = I$. Let $g, h \in \mathcal{P}(2)$ also be two-qubit Pauli operators with factors ± 1 such that $[s, g] = 0$ (they commute) and $s = gh$. The qubits φ_2 and $|\psi\rangle$ we choose shall have Bloch vectors

$$\varphi_2: (x_2, y_2, z_2) = \left(\sqrt{\frac{2}{17}}, \sqrt{\frac{5}{17}}, \sqrt{\frac{10}{17}} \right) \quad (4.27)$$

$$|\psi\rangle: (x, y, z) = \left(\sqrt{\frac{1}{11}}, \sqrt{\frac{3}{11}}, \sqrt{\frac{7}{11}} \right). \quad (4.28)$$

Let φ_1 be a qubit so that $\varphi_2 = \Phi_{1-v}(C, \varphi_1 \otimes \psi)$. Let $\varphi'_1 = \Phi_{v_2}(C_2, \varphi_2 \otimes \psi)$.

To prove the recovery by (C_2, v_2) will fail, we merely need to verify that the Bloch vector of φ'_1 resulting from all 18 choices of s are different, which implies $\varphi'_1 \neq \varphi_1$

whenever $C_1^\dagger(I \otimes |v_1\rangle\langle v_1|)C_1 \neq \Pi_2$. This job requires us to track the three coefficients

$$c_s = \text{tr}(s(\varphi_2 \otimes \psi)), \quad c_g = \text{tr}(g(\varphi_2 \otimes \psi)), \quad c_h = \text{tr}(h(\varphi_2 \otimes \psi)). \quad (4.29)$$

Performing a projection Π_2 on the two-qubit state $\varphi_2 \otimes \psi$ leads to

$$\text{tr}(\Pi_2(\varphi_2 \otimes \psi)\Pi_2) = \frac{1 + c_s}{2}, \quad (4.30)$$

$$\text{tr}(g\Pi_2(\varphi_2 \otimes \psi)\Pi_2) = \frac{c_g + c_h}{2}. \quad (4.31)$$

Renormalizing by $(1 + c_s)/2$ yields the following as a Bloch vector component of φ_1' :

$$v = \frac{c_g + c_h}{1 + c_s}. \quad (4.32)$$

We need to substitute in values for c_s , c_g , and c_h to prove our assertion. The most convenient choices for g and h are two-fold tensor products with one identity matrix e.g. $s = -Z \otimes Z$, $g = Z \otimes I$, $h = -I \otimes Z$, and $s = X \otimes X$, $g = X \otimes I$, $h = I \otimes X$, and etc. This selection means $c_s = c_g c_h$. If we look at the coefficients from the first example with $s = -Z \otimes Z$, then $c_g = z_2$ and $c_h = -z$. We get the following components v for each possibility of s :

s	c_s	g	c_g	h	c_h	v
$X \otimes X$	$x_2 x$	$X \otimes I$	x_2	$I \otimes X$	x	0.5841
$X \otimes Y$	$x_2 y$	$X \otimes I$	x_2	$I \otimes Y$	y	0.7338
$X \otimes Z$	$x_2 z$	$X \otimes I$	x_2	$I \otimes Z$	z	0.8957
$Y \otimes X$	$y_2 x$	$Y \otimes I$	y_2	$I \otimes X$	x	0.7252
$Y \otimes Y$	$y_2 y$	$Y \otimes I$	y_2	$I \otimes Y$	y	0.8296

(continuation of table):

s	c_s	g	c_g	h	c_h	v
$Y \otimes Z$	$y_2 z$	$Y \otimes I$	y_2	$I \otimes Z$	z	0.9354
$Z \otimes X$	$z_2 x$	$Z \otimes I$	z_2	$I \otimes X$	x	0.8678
$Z \otimes Y$	$z_2 y$	$Z \otimes I$	z_2	$I \otimes Y$	y	0.9205
$Z \otimes Z$	$z_2 z$	$Z \otimes I$	z_2	$I \otimes Z$	z	0.9708
$-X \otimes X$	$-x_2 x$	$X \otimes I$	x_2	$-I \otimes X$	$-x$	0.0463
$-X \otimes Y$	$-x_2 y$	$X \otimes I$	x_2	$-I \otimes Y$	$-y$	-0.2183
$-X \otimes Z$	$-x_2 z$	$X \otimes I$	x_2	$-I \otimes Z$	$-z$	-0.6260
$-Y \otimes X$	$-y_2 x$	$Y \otimes I$	y_2	$-I \otimes X$	$-x$	0.2879
$-Y \otimes Y$	$-y_2 y$	$Y \otimes I$	y_2	$-I \otimes Y$	$-y$	0.0280
$-Y \otimes Z$	$-y_2 z$	$Y \otimes I$	y_2	$-I \otimes Z$	$-z$	-0.4501
$-Z \otimes X$	$-z_2 x$	$Z \otimes I$	z_2	$-I \otimes X$	$-x$	0.6055
$-Z \otimes Y$	$-z_2 y$	$Z \otimes I$	z_2	$-I \otimes Y$	$-y$	0.4083
$-Z \otimes Z$	$-z_2 z$	$Z \otimes I$	z_2	$-I \otimes Z$	$-z$	-0.0792

Neither are any of the values v the same if we multiple each one by -1 , which may come about from an application of a single qubit Pauli gate on φ'_1 . Thus (C_2, v_2) is not a recovery circuit if Equation 4.25 does not hold. ■

With Lemma 6 in place, we can now guarantee that not any one recovery circuit will outperform another.

Lemma 7 *Let (C, v) be an interacting postselected stabilizer circuit, and let*

$$C'' = (G_1^\dagger \otimes I) \text{CNOT}(G_2 \otimes G) \quad (4.33)$$

be a two-qubit Clifford unitary such that

$$(C, 1 - v) \equiv ((G_2^\dagger \otimes I)\text{CNOT}(G_1 \otimes G), 1). \quad (4.34)$$

Then (C', v') is a recovery circuit of (C, v) if and only if $(C', v') \equiv (C'', 0)$.

Proof: In the reverse direction, equivalence of postselected stabilizer circuits means both produce the exact same output at the same success rate for all two-qubit states ρ . This certainly includes all two-qubit product states $\varphi_2 \otimes \psi$, where φ_2 is the output of $(C, 1 - v)$ on another input $\varphi_1 \otimes \psi$.

In the forward direction, Lemma 2 and Lemma 6 do most of the work: $(C', v') \sim (C'', 0)$. We just need to prove equivalence with “ \equiv ”. We look back at the definition of Clifford equivalent postselected stabilizer circuits, where we must have a single qubit Clifford gate G' such that

$$(I \otimes \langle v'|)C'\rho C'^\dagger(I \otimes |v'\rangle) = (G' \otimes \langle 0|)C''\rho C''^\dagger(G'^\dagger \otimes |0\rangle) \quad (4.35)$$

for all two-qubit states ρ . If it is indeed the case that they are strictly Clifford equivalent i.e. $G' \neq I$, then (C', v') cannot have been a recovery circuit of (C, v) because the output from (C', v') on ρ will be rotated by G' . Thus the two must be equivalent (with “ \equiv ”). ■

From Lemmas 4 and 7, we reach the second main result of this chapter.

Theorem 3 (van Dam and Wong [69], Thm. 12) *For each interacting postselected stabilizer circuit (C, v) , there is a recovery circuit (C', v') of (C, v) . Moreover, all recovery circuits of (C, v) are equivalent to (C', v') .*

Corollary 3 *Every recovery circuit (C', v') has its own recovery circuit (C'', v'') . Furthermore, there exist recovery circuits of (C', v') such that $v'' = v'$.*

Proof: The first claim is a given since recovery circuits are interacting postselected stabilizer circuits themselves by Lemma 5. If the initial recovery circuit (C'', v'') we construct already satisfies $v'' = v'$, then we are done. Otherwise, $v'' = 1 - v'$, so $(C'', v'') \equiv ((I \otimes X)C'', v')$. ■

4.4 Example Routines Featuring Recovery Circuits

Recovery circuits appear in the literature, where the most prominent use case for our recovery technique is a state injection process to implement non-Clifford operations. For instance, the programmable ancilla rotation (PAR) of [41] uses qubits of the type

$$|\theta\rangle = \frac{|0\rangle + e^{i\theta}|1\rangle}{\sqrt{2}} \quad (4.36)$$

and an interacting circuit CNOT to rotate $|q\rangle = \alpha|0\rangle + \beta|1\rangle$ about the Z -axis by an angle θ . This is demonstrated in Figure 4.4a. On the chance that the Z -measurement returns 1, then instead of $|q + \theta\rangle = \alpha|0\rangle + e^{i\theta}\beta|1\rangle$, the output becomes $|q - \theta\rangle = \alpha|0\rangle + e^{-i\theta}\beta|1\rangle$, which is $|q\rangle$ rotated by $-\theta$. Jones et. al [41] suggest pairing $|q - \theta\rangle$ with $|2\theta\rangle$ as a direct line to $|q + \theta\rangle$, but we can alternatively break this down into two smaller steps if $|\theta\rangle$ are the only states available. We first run the CNOT circuit on $|q - \theta\rangle \otimes |\theta\rangle$. If we measure 0, then we recover $|q\rangle$, and we proceed with rerunning circuit CNOT on $|q\rangle \otimes |\theta\rangle$.

The method by Duclos-Cianci and Svore [26] is similar. The idea is to use the same interacting circuit CNOT to first generate “ladder” qubit states of the kind

$$|H_i\rangle = \cos(\gamma_i)|0\rangle + \sin(\gamma_i)|1\rangle, \quad \cot(\gamma_i) = \cot^{i+1}(\pi/8) \quad (4.37)$$

for $i \geq 0$, then inject the “ladder” qubits into a similar circuit to perform phase rotations on some qubit $|q\rangle$. The production starts by supplying two copies of the magic state

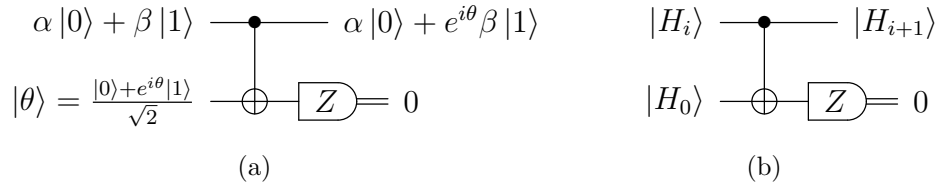


Figure 4.4: The postselected stabilizer circuit $(\text{CNOT}, 0)$ appears in both [26] and [41], with each one supplying a different input set to circuit CNOT. The qubit $|H_0\rangle$ in (b) is the $+1$ eigenstate of the H gate, and the process of generating $|H_{i+1}\rangle$ in [26] starts with $|H_0\rangle \otimes |H_0\rangle$. For (a), a qubit $|\theta\rangle$ leads to a $+\theta$ rotation about the z -axis on the first qubit $\alpha|0\rangle + \beta|1\rangle$ upon measuring 0.

$|H_0\rangle = H|H_0\rangle$ to the CNOT circuit in Figure 4.4b. Each time we gain a new state $|H_i\rangle$, we can reuse the qubit in an attempt to create the next $|H_{i+1}\rangle$. If the attempt fails, then the output of $(\text{CNOT}, 1)$ on $|H_i\rangle \otimes |H_0\rangle$ is $|H_{i-1}\rangle$. Given that the recovery circuit of $(\text{CNOT}, 0)$ is itself, the method to recover $|H_i\rangle$ from $|H_{i-1}\rangle \otimes |H_0\rangle$ is no different than the procedure to create it.

In addition to pure qubits, our notation for the two-qubit state $\varphi \otimes \psi$ throughout the previous section indicates that φ is allowed to be mixed, and it can even be part of a larger entangled system. As a quick demonstration, suppose we have the situation as illustrated in Figure 4.5a. Let (C', v') be a recovery circuit of (C, v) and let

$$U\rho U^\dagger = \frac{1}{2^n} (\mathbf{P}_I \otimes I + \mathbf{P}_X \otimes X + \mathbf{P}_Y \otimes Y + \mathbf{P}_Z \otimes Z) \quad (4.38)$$

where \mathbf{P}_L are Pauli operator sums on the first $n-1$ qubits. While the proof to Lemma 4 is generalizable to include the unused portions \mathbf{P}_L of the entangled state, the math is simpler and works out the same if we trace out the first $n-1$ qubits, keeping only the last qubit $\varphi = \text{tr}_{1,n-1}(U\rho U^\dagger)$ that we need for the two-qubit circuit. If we are unlucky, then qubit n becomes $\varphi' = \Phi_{1-v}(C, \varphi \otimes \psi)$, but we can try to regain φ by executing circuit C' on $\varphi' \otimes \psi$. If the recovery is successful, then we have another opportunity at the output $\Phi_v(C, \varphi \otimes \psi)$. In all likelihood, this is a less lengthy process than preparing

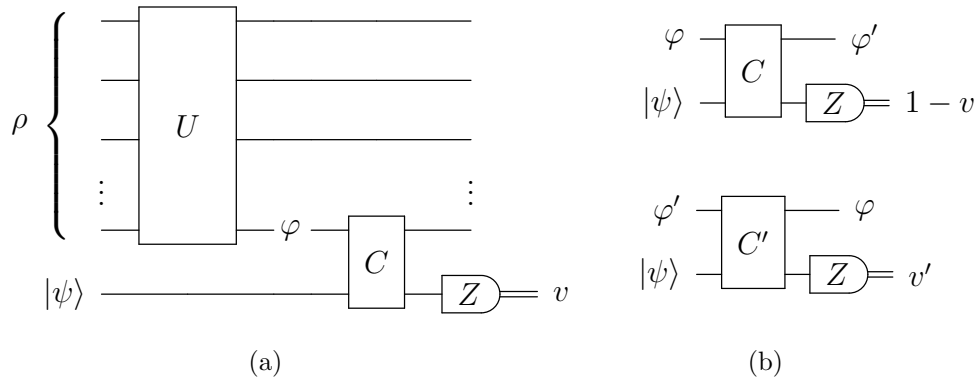


Figure 4.5: Recovery circuits also work when one of the qubits is entangled with another system. In (a), we trace out all but the n th qubit of $U\rho U^\dagger$ to get $\varphi \otimes \psi$ as input to circuit C . If we measure $1 - v$ as pictured in the top circuit of (b), then we execute circuit C' on $\varphi' \otimes \psi$ to try and recover φ . We succeed with the recovery if we measure v' .

another copy of ρ and running the circuit of U again; by some estimates, a synthesis of U over a universal gate set may require an exponential number of gates [32]. This is a stark contrast to C' , which consists of one CNOT possibly surrounded by some single qubit Clifford gates.

4.5 Summary

We have shown that two-qubit stabilizer circuits require nothing more than a few Clifford gates to perform a job. These simplifications shed light into the reciprocal nature between interacting circuits. Despite measurements generally being irreversible, we find an exception when handling a two-qubit product state $\varphi \otimes \psi$. That is, we can use a pure qubit $|\psi\rangle$ in conjunction with a specific circuit to salvage the resource qubit φ . In the next chapter, we follow-up on Corollary 3 and extended the recovery procedure given here to employ multiple recovery circuits.

Chapter 5

Extending the Recovery for Two-qubit Stabilizer Circuits

Recovery circuits are attractive because they describe a simple manner in which we may “invert” a two-qubit stabilizer operation on non-stabilizer qubits φ and ψ . Be that as it may, recovery circuits are not perfect: the nature of our “inverse” operation means that a successful recovery is not guaranteed on every try. On such occasions, we have two choices: stop and reset, or proceed with a secondary recovery attempt on the output of the unsuccessful recovery trial. If the subsequent effort is successful, then we rerun our primary recovery operation; if not, then we are once again left with the same predicament. Based on this observation, we may stitch together multiple circuit layers to form one nested recovery protocol that extends the lifetime of the recovery process. Due to the way our protocol is structured, we show in Theorem 4 that the circuits’ recovery success rates are recursively related. We conclude with a couple numerical experiments showcasing the benefits of this technique.

5.1 Nested Recovery Protocol

Let (C_1, v_1) be a postselected two-to-one stabilizer circuit, and suppose we want the output of (C_1, v_1) on two qubits $\varphi_1 \otimes \psi$. Then by repeated application of Corollary 3, we

can derive a depth k protocol on $k-1$ interacting postselected stabilizer circuits such that (C_i, v_i) is a recovery circuit of (C_{i-1}, v_{i-1}) . The minimum depth value is $k = 2$, which stands for a protocol without recovery, and we may assume without loss of generality a desirable outcome $v_i = 0$ for all $k-1$ circuits. Thus when circuit C_1 is unsuccessful i.e. measure the value 1, we fall back on circuit C_2 . If circuit C_2 is also unsuccessful, we depend on circuit C_3 , and so on all the way down to circuit C_{k-1} . In more detail, our protocol works as follows.

Depth k Nested Recovery Protocol:

1. Prepare the initial two-qubit state $\varphi_1 \otimes \psi$. Let $(C_1, 0), \dots, (C_{k-1}, 0)$ be interacting postselected stabilizer circuits such that $(C_i, 0)$ is a recovery circuit of $(C_{i-1}, 0)$.
2. Run circuit C_1 on $\varphi_1 \otimes \psi$. If we measure 0, then we declare *success*. Otherwise, let φ_2 be the output of $(C_1, 1)$ on $\varphi_1 \otimes \psi$.
3. Run circuit C_2 on $\varphi_2 \otimes \psi$. If we measure 0, then we recover φ_1 and we repeat step 2. Otherwise we get the output φ_3 of $(C_2, 1)$ on $\varphi_2 \otimes \psi$.
4. Repeat step 3 as necessary for other circuits C_i . That is, let φ_i be the output of $(C_{i-1}, 1)$ on $\varphi_{i-1} \otimes \psi$. Run circuit C_i on $\varphi_i \otimes \psi$. On measuring 0, the output is φ_{i-1} and we rerun circuit C_{i-1} on $\varphi_{i-1} \otimes \psi$. Otherwise, we move forward with circuit C_{i+1} on $\varphi_{i+1} \otimes \psi$.
5. If circuit C_{k-1} is unsuccessful on $\varphi_{k-1} \otimes \psi$, then we declare *failure* and stop.

This process is repeated until we secure the target output $\varphi_0 = \Phi_0(C_1, \varphi_1 \otimes \psi)$, which requires preparing a new copy of φ_1 each time. But by adding more circuits, we prolong our attempts at gaining φ_0 while reducing the number of times we rerun the computation on a new φ_1 . Notice also that the circuit layers are organized in a nested manner, creating

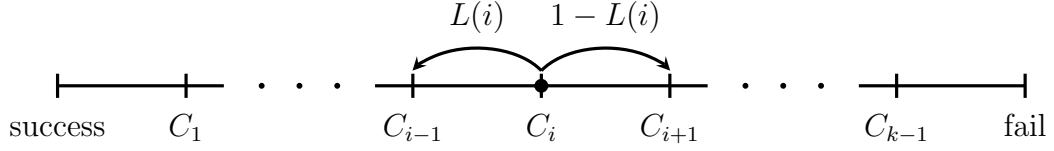


Figure 5.1: The behavior of our depth k protocol corresponds to a bounded random walk over the integers $\{0, \dots, k\}$ and starts at position 1. The random walk ends upon reaching 0 or k , with 0 representing success and k representing failure. The transition from i to $i - 1$ is the success probability of the i th circuit C_i in the protocol.

a depth k protocol that contains within a depth $k - 1$ protocol. Since the depth affects the number of $|\psi\rangle$ qubits spent on each invocation, we need to be somewhat prudent with increasing the depth so as to not render the recovery more expensive than to restart.

Given the description above, we model our depth k protocol as a one-dimensional random walk on $k + 1$ integers $\{0, \dots, k\}$, starting at location 1. A step onto the left boundary 0 translates to success, and a step onto the right boundary k indicates failure. The success probability of circuit C_i is the left step transition probability from position i to $i - 1$. Movement in either direction consumes one resource qubit $|\psi\rangle$. Not surprisingly, the recovery probability for every circuit C_2 to C_{k-1} is computable from the first success probability $Q_0(C_1, \varphi_1 \otimes \psi)$. The next theorem is an extension of Corollary 2.

Theorem 4 (van Dam and Wong [70], Lemma 7) *Let $(C_1, 0), \dots, (C_{k-1}, 0)$ be interacting postselected stabilizer circuits such that $(C_i, 0)$ is a recovery circuit of $(C_{i-1}, 0)$. Then given a two-qubit state $\varphi_1 \otimes \psi$ and outputs $\varphi_i = \Phi_1(C_{i-1}, \varphi_{i-1} \otimes \psi)$, the success probability of circuit C_i is*

$$L(i) = Q_0(C_i, \varphi_i \otimes \psi) = \begin{cases} Q_0(C_1, \varphi_1 \otimes \psi) & \text{if } i = 1 \\ \frac{(1 - z^2)/4}{1 - L(i-1)} & \text{if } i \in \{2, \dots, k-1\} \end{cases} \quad (5.1)$$

where $z \in \{\langle \psi | X | \psi \rangle, \langle \psi | Y | \psi \rangle, \langle \psi | Z | \psi \rangle\}$.

Proof: We primarily need to explain why the numerator stays the same at every step i , since we can infer the form from Equation 4.20 of Corollary 2. Suppose

$$(C_1, 1) \equiv ((G_2^\dagger \otimes I)\text{CNOT}(G_1 \otimes G), 1) \quad (5.2)$$

where G , G_1 , and G_2 are single qubit Clifford gates. This means we have equivalences

$$(C_1, 0) \sim (\text{CNOT}(G_1 \otimes G), 0) \quad (5.3)$$

$$(C_2, 0) \equiv ((G_1^\dagger \otimes I)\text{CNOT}(G_2 \otimes G), 0). \quad (5.4)$$

Next, there is a Clifford gate G_3 such that $(C_2, 1) \equiv ((G_3^\dagger \otimes I)\text{CNOT}(G_2 \otimes G), 1)$, which then implies $(C_3, 0) \equiv ((G_2^\dagger \otimes I)\text{CNOT}(G_3 \otimes G), 0)$. Continuing in this manner, we find single qubit Clifford gates G_i and G_{i+1}^\dagger satisfying

$$(C_i, 1) \equiv ((G_{i+1}^\dagger \otimes I)\text{CNOT}(G_i \otimes G), 1) \quad (5.5)$$

$$(C_{i+1}, 0) \equiv ((G_i^\dagger \otimes I)\text{CNOT}(G_{i+1} \otimes G), 0) \quad (5.6)$$

for all $i \geq 1$. We study the effects of each postselected circuit $(C_i, 1)$ on $\varphi_i \otimes \psi$ and $(C_{i+1}, 0)$ on $\varphi_{i+1} \otimes \psi$ via the equivalent postselected circuits just described.

Consider the qubits $|\psi'\rangle = G|\psi\rangle$ and $\varphi'_i = G_i\varphi_iG_i^\dagger$. From our G_{i+1} selection, this means that

$$\varphi'_{i+1} = \Phi_1(\text{CNOT}, \varphi'_i \otimes \psi') = G_{i+1}\varphi_{i+1}G_{i+1}^\dagger \quad (5.7)$$

$$\varphi'_i = \Phi_0(\text{CNOT}, \varphi'_{i+1} \otimes \psi'). \quad (5.8)$$

Observe that both gates G_i and G_{i+1}^\dagger to the control qubit in $((G_{i+1}^\dagger \otimes I)\text{CNOT}(G_i \otimes G), 1)$ are always neutralized by the recovery circuit $((G_i^\dagger \otimes I)\text{CNOT}(G_{i+1} \otimes G), 0)$. In other

words, at each step i , we always apply CNOT on qubits φ'_i and $|\psi'\rangle$ as if the rotations by G_i and G_{i+1}^\dagger never took place. In Chapter 4, we saw $(C_1, 0) \sim (\text{CNOT}(G_1 \otimes G), 0)$ and $(C_2, 0) \sim (\text{CNOT}(G_2 \otimes G), 0)$ pave the way to Equation 4.20. We apply the same arguments between $(C_i, 0)$ and $(C_{i+1}, 0)$ to obtain the recurrence above. ■

Lucky for us, the form of the success probability equation $L(i)$ in Theorem 4 is a familiar one in mathematics called a rational difference equation. We shall use this connection to help us assess the protocol more thoroughly in Chapter 6. Before then, we narrow the success probability of each circuit C_i to a more specific range.

Corollary 4 *Let $(C_1, 0), \dots, (C_{k-1}, 0)$ be interacting postselected stabilizer circuits such that $(C_i, 0)$ is a recovery circuit of $(C_{i-1}, 0)$. Then given a two-qubit state $\varphi_1 \otimes \psi$ and outputs $\varphi_i = \Phi_1(C_{i-1}, \varphi_{i-1} \otimes \psi)$, the success probability of circuit C_i is bounded above and below by*

$$\frac{1 - \sqrt{1 - 4\lambda}}{2} \leq L(i) = Q_0(C_i, \varphi_i \otimes \psi) \leq \frac{1 + \sqrt{1 - 4\lambda}}{2} \quad (5.9)$$

where $\lambda = \frac{1 - z^2}{4}$ and $z \in \{\langle \psi | X | \psi \rangle, \langle \psi | Y | \psi \rangle, \langle \psi | Z | \psi \rangle\}$.

Proof: Assume $C_i = \text{CNOT}$ for simplicity. Then $z = \langle \psi | Z | \psi \rangle$ and $z_i = \text{tr}(Z\varphi_i)$.

This gives

$$\frac{1 - |z|}{2} \leq L(i) = \frac{1 + z_i z}{2} \leq \frac{1 + |z|}{2} \quad (5.10)$$

since $z_i \in [-1, 1]$. But we can also say

$$\frac{1 + \sqrt{1 - 4\lambda}}{2} = \frac{1 + |z|}{2}, \quad \frac{1 - \sqrt{1 - 4\lambda}}{2} = \frac{1 - |z|}{2} \quad (5.11)$$

which implies the inequality. ■

Note that only positive values of $\lambda \leq 1/4$ are relevant under our circumstances because $\lambda = 0$ if $z = \pm 1$, which occurs whenever $G|\psi\rangle = |0\rangle$ or $|1\rangle$ prior to CNOT (see proof to Theorem 4 for greater details about the single qubit Clifford gate G). We have similar implications when

$$L(i) = \frac{1 + \sqrt{1 - 4\lambda}}{2} > \frac{1}{2} \quad \text{or} \quad L(i) = \frac{1 - \sqrt{1 - 4\lambda}}{2} < \frac{1}{2}. \quad (5.12)$$

Equality with either limit for all i suggests one of the input qubits φ_1 or $|\psi\rangle$ is a stabilizer state. By Lemma 3, we may exchange the postselected stabilizer circuit $(C_1, 0)$ for some single qubit operation, and the protocol is no longer suitable for this domain. On the other hand, we cannot completely eliminate the case of $\lambda = 1/4$, which causes $L(i) = 1/2$. This situation applies to all $|\psi\rangle$ qubits belonging to the Bloch sphere equator. We generally assume that $|\psi\rangle$ is not one four single qubit stabilizer states $\{|+\rangle, |-\rangle, |+i\rangle, |-i\rangle\}$.

Finally, observe that with Theorem 4, we may essentially treat our depth k protocol more abstractly as a sequence of numbers $L(1), \dots, L(k-1)$, generated entirely by a recurrence relation $L(i)$ defined on two real values

$$\lambda = \frac{1 - z^2}{4}, \quad \gamma = L(1) = Q_0(C_1, \varphi_1 \otimes \psi) \quad (5.13)$$

where z depends on C_1 and $|\psi\rangle$. The depth k only serves to indicate a stopping point when generating that sequence, so our protocol is basically controlled by λ , γ , and k . This observation will come in handy when we present our analysis in Chapter 6.

5.2 Experimentation with Recovery Circuits

Since our depth k protocol behaves like a random walk, we may conduct simulations of the Markov process to obtain a better estimate for N_k , the expected number of $|\psi\rangle$

resources needed to create one $\varphi_0 = \Phi_0(C_1, \varphi_1 \otimes \psi)$. Let d be the cost to prepare a single instance of φ_1 relative to the cost of $|\psi\rangle$. Then the cost of one trial is the same as d plus the number of $|\psi\rangle$ qubits used before halting, regardless of outcome. The costs from all trials are tallied and divided by the number of successes to obtain N_k . We compare this value against the expected cost without recovery, which is simply $N_2 = (d+1)/L(1)$. We assume for the sake of simplicity that $C_1 = \text{CNOT}$, which means $C_2 = \text{CNOT}$, and so forth for the other $k - 3$ circuits.

Another variable that we keep constant is $L(1) = 1/2$. Since we fix the first success probability, N_k is dependent on the parameter $z = \langle \psi | Z | \psi \rangle$ that appears in the recovery success rate of Theorem 4. Generally, we need a different φ_1 with each choice of $|\psi\rangle$ to maintain the same $L(1)$ as well as the same output φ_0 . Usually different φ_1 means different costs d , but we will ignore this momentarily and assume the preparation overhead d for each φ_1 is the same for the purposes of a broader comparison of N_k across different $|\psi\rangle$ resources. In the first set of experiments, we include only one recovery circuit ($k = 3$). The following table summarizes the expected costs for four samples of z obtained over the course of 100000 trials:

d	N_2	$N_3: z = \sqrt{0.96}$	$N_3: z = \sqrt{0.50}$	$N_3: z = \sqrt{0.04}$	$N_3: z = 0$
10^{-1}	2.2	3.20	3.18	3.15	3.15
10^0	4.0	4.99	4.75	4.51	4.50
10^1	22	22.7	20.5	18.2	18.0
10^2	202	200.4	177.9	155.1	157.7
10^3	2002	1988.9	1750.7	1521.9	1498.7
10^4	20002	19816.4	17488.0	15215.4	14998.7
10^5	200002	198246.6	174852.6	151946.3	149719.3

The first row with $d = 0.1$ should be interpreted as φ_1 being cheaper to create than $|\psi\rangle$.

We clearly see an improvement when factoring in recovery in the face of large relative preparation overhead between φ_1 and $|\psi\rangle$. We also see a trend of lower costs as z grows smaller, when $|\psi\rangle$ is moving closer to the xy -plane in the Bloch sphere. This is due to the differences in the recovery success rate at circuit C_2 , which are 0.02, 0.25, 0.48, and 0.5, respectively.

In the second batch of experiments, we maintain $d = 1000$ but vary the number of circuits parameterized by k . Again, $L(1) = 1/2$ and we run 100000 trials. Data for N_k is compiled together in the table below, starting with $k = 3$:

k	$N_k: z = \sqrt{0.96}$	$N_k: z = \sqrt{0.50}$	$N_k: z = \sqrt{0.04}$	$N_k: z = 0$
3	1981.7	1753.2	1522.9	1501.6
4	1982.9	1720.5	1372.2	1336.9
5	1982.4	1716.5	1302.9	1255.2
6	1987.5	1710.9	1266.6	1206.2
7	1982.5	1715.3	1246.7	1174.7
8	1989.2	1714.9	1232.5	1151.5
9	1994.5	1714.6	1224.9	1133.9
10	1991.7	1717.0	1221.5	1120.8
20	2002.5	1727.3	1220.2	1072.9
30	2006.3	1734.6	1231.4	1064.5
40	2023.5	1743.7	1240.8	1066.3
50	2025.6	1762.5	1250.7	1071.4
60	2044.1	1768.7	1255.3	1077.8

Observe that the value of N_k continues to lower noticeably for some of the $|\psi\rangle$ cases as more circuits are added before increasing again. This behavior is no surprise since at some

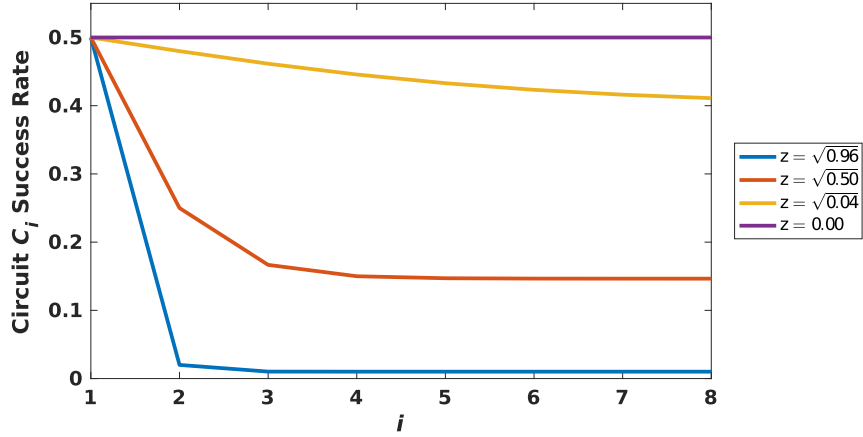


Figure 5.2: The success probability between circuit C_i and circuit C_{i+1} defined recursively in Theorem 4 drops more dramatically as z moves closer to 1. This leads to a greater expected cost N_k of our protocol since the recovery is less likely to succeed relative to other choices of z . On the other end of the spectrum, the success probability of each circuit C_i is uniform when $z = 0$.

point, the penalty to sustain the recovery process will exceed the overhead of repeating the computation. If we look at the success probabilities for the first seven circuits of the protocol for each of the four z samples in Figure 5.2, we also see the success rates decrease to some lower boundary as i increases, with the exception of when $z = 0$. This is precisely what we predicted in Corollary 4. For the four z values we tested, the lower bounds are

$$z = \sqrt{0.96} : 0.0101, \quad z = \sqrt{0.50} : 0.1464, \quad (5.14)$$

$$z = \sqrt{0.04} : 0.4, \quad z = \sqrt{0.00} : 1/2. \quad (5.15)$$

The drop in probabilities from circuit C_1 to circuit C_3 is quite significant when z is close to 1 (and $1 - z^2$ is small), so the chance of recovery at circuit C_3 is only slightly larger than 0. This explains why there is no apparent change in N_k between one recovery circuit ($k = 3$) versus two ($k = 4$) for the case $z = \sqrt{0.96}$. The ideal situation is to know beforehand how many circuits to include to minimize resource usage.

5.3 Summary

Using Corollary 3, we derived a depth k protocol that employs up to $k - 1$ two-qubit stabilizer circuits to prolong the recovery of an expensive input qubit φ_1 . Experiments corroborate the belief that as the cost of φ_1 increases, then so too are the benefits of this scheme. To further improve on the $|\psi\rangle$ resource consumption, we must identify the optimal nesting depth to minimize the expected cost. Fortunately, the success probability of each circuit in the protocol can be determined recursively and resembles a rational difference equation. As such, we are able to deliver a more comprehensive look into the protocol's effectiveness in Chapter 6.

Chapter 6

Performance Analysis of Nested Recovery

At this stage, we are confident enough to say that recovery circuits will make a nice addition to our quantum computational toolkit, but the extent of its impact is not yet fully known. Validated by the experiments in Section 5.2, we continue the evaluation of recovery circuits in this chapter. More precisely, we pursue a rigorous examination of the nested recovery protocol from Chapter 5 to answer questions about its optimal nesting depth. We do this by modeling our protocol as a bounded random walk with a special transition function to obtain exact expressions for calculating the expected cost (Theorem 5). Through our analysis, we learn that for a startup cost d to initialize the input two-qubit state, that a protocol of depth $o(d)$ is optimal and attains the desired minimization of resources (Theorem 6). Under this assumption, we discover up to a factor of two savings is achievable over a protocol that ignores recovery (Theorem 7).

6.1 Prelude to Analysis

To facilitate the presentation of our analysis, recall from Theorem 4 that for our depth k protocol, that our success probability function $L(i)$ (Equation 5.1) is in fact a rational difference equation parameterized by two numbers. If $\varphi_1 \otimes \psi$ is the initial two-qubit

state, and $(C_1, 0)$ is the first postselected stabilizer circuit in the protocol, then we may simplify Equation 5.1 by setting

$$\lambda = \frac{1 - z^2}{4}, \quad \gamma = L(1) = Q_0(C_1, \varphi_1 \otimes \psi) \quad (6.1)$$

where $z = \langle \psi | G^\dagger Z G | \psi \rangle$, and G is a C_1 dependent single qubit Clifford gate. Since the $k - 1$ probabilities $L(1), \dots, L(k - 1)$ of our protocol can be determined as long as we know (λ, γ, k) , throughout this chapter, we will usually say that an instance of our protocol is set according to an assignment on these three values.

Being parameters of $L(i)$, there are certain constraints that λ and γ must comply with in order for the $L(i)$ numbers to be valid probabilities. According to the equations above, λ must lie between 0 and $1/4$. Another way to imagine λ is the amount of overlap that $G|\psi\rangle$ makes with respect to the Bloch sphere xy -plane, rescaled by $1/4$, since $1 - z^2 = x^2 + y^2$ for the Bloch vector (x, y, z) of any pure qubit. Knowing λ , we then need γ to adhere to the conditions that we identified in Corollary 4. We establish the concept of a probability specification to serve that very purpose, allowing us to specify a rational difference equation solely with respect to these two values. Formally, a probability specification is defined as follows.

Definition 8 (Probability Specification and Boundary) *Let (λ, γ) be a pair of real numbers. Given λ , define*

$$\alpha = \frac{1 + \sqrt{1 - 4\lambda}}{2}, \quad \beta = 1 - \alpha = \frac{1 - \sqrt{1 - 4\lambda}}{2}. \quad (6.2)$$

Then (λ, γ) is a probability specification if and only if $0 \leq \lambda \leq 1/4$ and $\beta \leq \gamma \leq \alpha$. A probability specification is restricted if and only if $0 < \lambda < 1/4$ and $\beta < \gamma < \alpha$. The values (α, β) are the boundaries of the probability specification.

Before we explain the motivation behind a restricted probability specification, let us introduce a few convenience functions, as well as detail the version of rational difference equations that we depend on.

Definition 9 (Intermediate Functions) *Let (α, β) be the boundaries of a probability specification (λ, γ) . The following are the intermediate functions of (λ, γ) :*

$$A_1(i) = \alpha^i - \beta^i, \quad B_1(i) = A_1(i+1) - \gamma A_1(i), \quad (6.3)$$

$$A_2(i) = \alpha^i + \beta^i, \quad B_2(i) = A_2(i+1) - \gamma A_2(i). \quad (6.4)$$

Definition 10 (Rational Difference Equation (RDE)) *Let (λ, γ) be a probability specification. Then the following is a rational difference equation (RDE) with parameters (λ, γ) :*

$$L(i) = \frac{\lambda B_1(i-2)}{B_1(i-1)} = \begin{cases} \gamma & \text{if } i = 1 \\ \frac{\lambda}{1 - L(i-1)} & \text{otherwise} \end{cases} \quad (6.5)$$

where $B_1(i)$ is an intermediate function of (λ, γ) .

There are some obvious yet important qualities that we immediately notice about RDEs. The proof simply follows from $\lambda = \alpha\beta$.

Lemma 8 *Let (λ, γ) be a probability specification, let (α, β) be its boundaries, and let $L(i)$ be an RDE with parameters (λ, γ) . Then the three statements below are true:*

1. if $\lambda = 1/4$, then $L(i) = \beta = \gamma = \alpha = 1/2$.
2. if $\gamma = \alpha$, then $L(i) = \alpha$.
3. if $\gamma = \beta$, then $L(i) = \beta$.

We have already touched on $\lambda = \alpha > 1/2$ and $\lambda = \beta < 1/2$ in Section 5.1. That is to say, these two cases correspond to situations when either input qubit φ_1 or $|\psi\rangle$ is a stabilizer state, which yields φ_1 or $|\psi\rangle$ as the output (up to a single qubit Clifford gate). The same is also true when $\lambda = 0$. Hence we define a restricted probability specification as satisfying both $0 < \lambda < 1/4$ and $\beta < \gamma < \alpha$.

6.2 Expected Cost

To gain a better estimate for the optimal depth, we need to define an expected cost function that accurately captures the resource requirements of our protocol. There are essentially three main ingredients to computing the expected cost.

Definition 11 (Success Probability (NRP)) *The success probability for a depth k protocol is the probability of declaring success before declaring failure.*

Definition 12 (Startup Cost) *Consider a depth k protocol that starts by running circuit C_1 on a two-qubit state $\varphi_1 \otimes \psi$. Then the startup cost is the cost to prepare one φ_1 qubit relative to the cost of $|\psi\rangle$ qubit.*

Definition 13 (Expected Demand) *Consider again a depth k protocol that starts by running circuit C_1 on two qubits $\varphi_1 \otimes \psi$. Then the expected demand is the expected number of $|\psi\rangle$ states used in each execution, regardless of the final success or fail outcome.*

That being said, we now give a precise definition of the expected cost. Intuitively, it expresses the expected number of $|\psi\rangle$ qubits utilized before the protocol succeeds.

Definition 14 (Expected Cost (NRP)) *The expected cost for a depth k protocol is $N = (d + s)/p$, where d represents the startup cost, p is the protocol's success probability, and s is the expected demand.*

As we stated earlier, certain combinations of λ and γ bear no significance due to the kinds of input qubits they imply. Therefore we ignore cases when $\lambda = \alpha > 1/2$, $\lambda = \beta < 1/2$, and $\lambda = 0$, where α and β are the boundaries of the probability specification (λ, γ) . This leaves us with two options: (λ, γ) is a restricted probability specification, and $\lambda = 1/4$. The latter of the two means $L(i) = \beta = \gamma = \alpha = 1/2$ and corresponds to a uniform random walk on the number line. We treat this separate from the more generic, former situation. In the next lemma, we present the success probability and expected demand for a protocol in the generic case.

Lemma 9 *Let $A_1(i)$ and $B_2(i)$ be intermediate functions of a restricted probability specification (λ, γ) . Then the success probability for a depth k protocol set to (λ, γ) is*

$$p = \frac{\gamma A_1(k-1)}{A_1(k)}. \quad (6.6)$$

The expected demand for a depth k protocol set to the same parameters is

$$s = \frac{A_1(k-1)(\gamma - 2\lambda) + (k-1)A_1(1)B_2(k-1)}{(A_1(1))^2 A_1(k)}. \quad (6.7)$$

Proof: The proof follows from Lemma 14 in Appendix Section A.2. To summarize, we model our protocol as a random walk with an RDE set to (λ, γ) as the transition. ■

With that in mind, we finally have the pieces necessary to compute the expected cost for both a restricted (λ, γ) and $\lambda = 1/4$.

Theorem 5 (van Dam and Wong [70], Lemma 15) *The expected cost for a depth k protocol with startup cost d and set to a restricted probability specification (λ, γ) is*

$$N(k) = \frac{dA_1(k)}{\gamma A_1(k-1)} + \frac{(k-1)B_2(k-1)}{\gamma A_1(1)A_1(k-1)} + \frac{\gamma - 2\lambda}{\gamma (A_1(1))^2} \quad (6.8)$$

where $A_1(i)$ and $B_2(i)$ are intermediate functions of the probability specification (λ, γ) . The expected cost for a protocol with the assignment $\lambda = 1/4$ is

$$N(k) = \frac{k^2 + kd - k}{k - 1}. \quad (6.9)$$

Proof: The proof is straightforward from $N(k) = (d + s)/p$, Lemma 15 in Appendix Section A.2, and Lemma 9 above. ■

6.3 Minimizing Expected Cost

We want to find the integer depth $k_{\text{opt}} \geq 2$ that minimizes the expected cost $N(k)$. We initially distinguish this from the problem of solving the global minimum $N_{\min} = \min_{k \geq 2} N(k)$, and finding the real number k_{\min} such that $N_{\min} = N(k_{\min})$. Fortunately, there is evidence to suggest that $N(k)$ has a single critical point. Figures 6.1 and 6.2 show the expected cost for several protocols set to varying restricted probability specifications (λ, γ) and startup costs d . The data provides a convincing argument to assume our expected cost function has one global minimum. This means if we find k_{\min} , we can easily obtain k_{opt} .

There is good reason to utilizing $k_{\text{opt}} - 1$ max circuits: if the depth is too shallow, then we are stopping prematurely and not taking full advantage of the recovery option; if the depth is too deep, then we put more work into performing the recovery than desired. In the latter event, it is wiser to start over with new copies of φ_1 and $|\psi\rangle$.

6.3.1 Optimal Depth: Generic Case

Given the nature of $N(k)$ from Theorem 5, we devote most of our efforts to answering k_{opt} for a protocol set to a restricted probability specification (λ, γ) . By the end of our

analysis, we propose that k_{opt} scales logarithmically with respect to the startup cost d . Let (α, β) be the boundaries of (λ, γ) . Then the first derivative in its entirety is

$$N'(k) = - \frac{\ln(\alpha/\beta) ((\alpha - \beta)^2 d + (k - 1)(1 - 2\gamma)) \lambda^{k-1}}{(\alpha - \beta) (\alpha^{k-1} - \beta^{k-1})^2 \gamma} + \frac{(\alpha^k + \beta^k - \gamma \alpha^{k-1} - \gamma \beta^{k-1})}{(\alpha - \beta) (\alpha^{k-1} - \beta^{k-1}) \gamma}. \quad (6.10)$$

Seeing how $N'(k)$ is transcendental, we rely on a combination of numerical and analytical approaches to justify our claim. A quick look at the limits of $N'(k)$ reveals the behavior of $N(k)$ falls within our expectations. That is, observe that the derivative is unbounded on one side:

$$\lim_{k \rightarrow 1^+} N'(k) \rightarrow -\infty \quad (6.11)$$

and that it reaches a constant in the other direction:

$$\begin{aligned} \lim_{k \rightarrow \infty} N'(k) &= \lim_{k \rightarrow \infty} - \frac{\ln(\alpha/\beta) ((\alpha - \beta)^2 d + (k - 1)(1 - 2\gamma))}{(\alpha - \beta) \left(1 - (\beta/\alpha)^{k-1}\right) \left((\alpha/\beta)^{k-1} - 1\right) \gamma} \\ &+ \frac{\left(\alpha + (\beta/\alpha)^{k-1} \beta - \left(1 + (\beta/\alpha)^{k-1}\right) \gamma\right)}{(\alpha - \beta) \left(1 - (\beta/\alpha)^{k-1}\right) \gamma} = \frac{\alpha - \gamma}{(\alpha - \beta) \gamma} > 0 \end{aligned} \quad (6.12)$$

since $\beta < \gamma < \alpha$. This is typical of a function with at least one minimum. If we let $k' = k - 1$ and make some rearrangements, we can rewrite $N'(k)$ as

$$\begin{aligned} N'(k') &= - \frac{\ln(\alpha/\beta) ((\alpha - \beta)^2 d + (1 - 2\gamma) k')}{(\alpha - \beta) \left(1 - (\beta/\alpha)^{k'}\right) \left((\alpha/\beta)^{k'} - 1\right) \gamma} \\ &+ \frac{(\alpha - \gamma) (\alpha/\beta)^{k'} + (\gamma - \beta) (\beta/\alpha)^{k'} - \alpha + \beta}{(\alpha - \beta) \left(1 - (\beta/\alpha)^{k'}\right) \left((\alpha/\beta)^{k'} - 1\right) \gamma}. \end{aligned} \quad (6.13)$$

We come up with a lower bound of $N'(k')$ by dropping the term $(\gamma - \beta)(\beta/\alpha)^{k'} \leq 1$:

$$N'_{\text{lb}}(k') = \frac{(\alpha - \gamma)(\alpha/\beta)^{k'} - \alpha + \beta - \ln(\alpha/\beta) \left((\alpha - \beta)^2 d + (1 - 2\gamma) k' \right)}{(\alpha - \beta) \left(1 - (\beta/\alpha)^{k'} \right) \left((\alpha/\beta)^{k'} - 1 \right) \gamma} \quad (6.14)$$

which may be used to locate an upper bound of k_{\min} . Starting with $N'_{\text{lb}}(k') = 0$, we get

$$\left(\frac{\alpha}{\beta} \right)^{k'} = \ln \left(\frac{\alpha}{\beta} \right) \left(\frac{1 - 2\gamma}{\alpha - \gamma} \right) k' + \frac{\ln(\alpha/\beta) (\alpha - \beta)^2 d + \alpha - \beta}{\alpha - \gamma}. \quad (6.15)$$

Making the substitution

$$-t = k' + \frac{\ln(\alpha/\beta) (\alpha - \beta)^2 d + \alpha - \beta}{\ln(\alpha/\beta) (1 - 2\gamma)} \quad (6.16)$$

turns Equation 6.15 into

$$t \left(\frac{\alpha}{\beta} \right)^t = -\frac{1}{t_0} \left(\frac{\alpha}{\beta} \right)^{-\frac{t_1}{t_0}} \quad (6.17)$$

where

$$t_0 = \ln \left(\frac{\alpha}{\beta} \right) \left(\frac{1 - 2\gamma}{\alpha - \gamma} \right), \quad t_1 = \frac{\ln(\alpha/\beta) (\alpha - \beta)^2 d + \alpha - \beta}{\alpha - \gamma}. \quad (6.18)$$

The solution t to Equation 6.17 indicates that

$$k_{\min} \leq k_{\text{up}} = -\frac{W \left(-\frac{\ln(\alpha/\beta)}{t_0} \left(\frac{\alpha}{\beta} \right)^{-\frac{t_1}{t_0}} \right)}{\ln(\alpha/\beta)} - \frac{t_1}{t_0} + 1 \quad (6.19)$$

where W is the product log or Lambert W -function. The product log function is defined as the inverse of $f(x) = xe^x$, so $x = W(xe^x)$. If in addition $\gamma = 1/2$, then $N'_{\text{lb}}(k') = 0$ is

easier to solve, leading to

$$k_{\text{up}} = \frac{\ln(\ln(\alpha/\beta)(\alpha - \beta)^2 d + \alpha - \beta) - \ln(\alpha - 1/2)}{\ln(\alpha/\beta)} + 1. \quad (6.20)$$

Figures 6.3 and 6.4 contain plots of k_{min} found using conventional optimization techniques. Aside from smaller values of the startup cost d , the graphs provide a compelling case that $k_{\text{opt}} = \Theta(\log d)$. Equation 6.20 is a good starting point to begin a search for the exact value of k_{opt} .

6.3.2 Optimal Depth: Special Case

The derivative of $N(k)$ when $\lambda = 1/4$ is much simpler by comparison:

$$N'(k) = \frac{(k-1)^2 - d}{(k-1)^2}. \quad (6.21)$$

The roots are $1 \pm \sqrt{d}$, of which only one is positive. From what we can gather, the optimal depth has a sublinear relationship with respect to the startup cost in both domains.

Theorem 6 (van Dam and Wong [70], Thm. 16) *Let d be the startup cost of a protocol set to a probability specification (λ, γ) . Then the optimal depth is $k_{\text{opt}} \in \left\{ \lceil 1 + \sqrt{d} \rceil, \lfloor 1 + \sqrt{d} \rfloor \right\}$ such that $N(k_{\text{opt}}) = \min(N(\lceil 1 + \sqrt{d} \rceil), N(\lfloor 1 + \sqrt{d} \rfloor))$ when $\lambda = 1/4$, and $k_{\text{opt}} = \Theta(\log d)$ when (λ, γ) is a restricted probability specification.*

6.4 Cost Ratio

To determine the effectiveness of our recovery protocol, we compare $N(2)$ – the method with no recovery whatsoever – against $N(k_{\text{opt}})$. We look at the $N(2)/N(k_{\text{opt}})$ cost ratio under the k_{opt} assumptions of Theorem 6.

Theorem 7 (van Dam and Wong [70], Thm. 17) *Let k_{opt} be the optimal depth of a protocol with startup cost d . Then*

$$\lim_{d \rightarrow \infty} \frac{N(2)}{N(k_{\text{opt}})} \leq 2. \quad (6.22)$$

Proof: We consider a restricted probability specification (λ, γ) first. Let (α, β) be its boundaries, and let $A_1(i)$, $B_2(i)$ be its intermediate functions. Given that $N(2) = (d+1)/\gamma$, the exact ratio is

$$\frac{N(2)}{N(k)} = \frac{(d+1) A_1(k-1) (A_1(1))^2}{d A_1(k) (A_1(1))^2 + (k-1) B_2(k-1) A_1(1) + (\gamma - 2\lambda) A_1(k-1)}. \quad (6.23)$$

In addition to $A_1(i) \leq 1$ and $B_2(i) \leq 2$ for all integers $i \geq 0$, we can factor out α^{k-1} from the top and bottom to say

$$\frac{N(2)}{N(k_{\text{opt}})} = \frac{(A_1(1))^2 \left(1 - (\beta/\alpha)^{k_{\text{opt}}-1}\right) (d+1)}{(A_1(1))^2 \left(1 - (\beta/\alpha)^{k_{\text{opt}}}\right) \alpha d + O(k_{\text{opt}})} \quad (6.24)$$

where we ignore lower order terms in the denominator. Since in this case $k_{\text{opt}} = \Theta(\log d)$ and $\beta < \alpha$, our conclusion now is more apparent:

$$\lim_{d \rightarrow \infty} \frac{(A_1(1))^2 \left(1 - (\beta/\alpha)^{\Theta(\log d)}\right) \left(1 + \frac{1}{d}\right)}{(A_1(1))^2 \left(1 - (\beta/\alpha)^{\Theta(\log d)}\right) \alpha + \frac{\Theta(\log d)}{d}} = \frac{1}{\alpha}. \quad (6.25)$$

The $\lambda = 1/4$ instance is very much the same. For simplicity, we use $k_{\text{min}} = 1 + \sqrt{d}$:

$$\lim_{d \rightarrow \infty} \frac{N(2)}{N(k_{\text{min}})} = \lim_{d \rightarrow \infty} \frac{2d\sqrt{d} + 2\sqrt{d}}{d\sqrt{d} + 2d + \sqrt{d}} = \frac{1}{\alpha} \quad (6.26)$$

since $\alpha = 1/2$. ■

6.5 Potential Improvements with Commonly Used Resource Qubits

According to Theorem 7, the best scenario is when $\lambda = 1/4$, which translates to $\alpha = 1/2$ and an expected cost reduction by up to half. We achieve this when performing phase rotations with a single CNOT and $|\psi\rangle = |\theta\rangle = (|0\rangle + e^{i\theta}|1\rangle)/\sqrt{2}$ at angles $0 < \theta < \pi/2$ and $\theta \neq \pi/4$. The probability of rotating in either $+\theta$ or $-\theta$ direction is both $1/2$. An alternative to recovery is to try a correction with $|2\theta\rangle$, but this turns out to be less optimal due to the preparation of $|2\theta\rangle$ from two $|\theta\rangle$ qubits. Observe that if we fail with $|2\theta\rangle$, then we need to prepare and succeed with $|2^2\theta\rangle$. If unsuccessful for a second straight time, then we need to succeed with $|2^3\theta\rangle$, and so on up to some max power of 2 exponent j . Since the optimal depth is about \sqrt{d} for startup cost d , the gap between 2^j and $\sqrt{d} + 1$ may be large, meaning this is worse than following the recovery protocol directly. Besides, the process to generate $|2^j\theta\rangle$ is probabilistic and basically identical to our nested recovery protocol. If we are successful in creating $|2^j\theta\rangle$, and $2^j > \sqrt{d} + 1$, then the effort that went into producing this qubit could have been redirected towards a successful recovery in the first place.

One particular example that may benefit are the V -basis gate implementations from [11]. Take the non-Clifford operation

$$V_3 = \frac{1+2i}{\sqrt{5}} \begin{bmatrix} 1 & 0 \\ 0 & -\frac{3}{5} - i\frac{4}{5} \end{bmatrix}, \quad (6.27)$$

as an example. The idea is to inject $|\theta_1\rangle$ such that $\cos(\theta_1) = 7\sqrt{2}/10$ and $\sin(\theta_1) = \sqrt{2}/10$ after performing T . Bocharov, Gurevich, and Svore [11] show that single qubit unitary approximations in the Clifford+ V universal basis has the potential to be lower than

Clifford+ T . If we have a long sequence of Clifford+ V gates $U_l \cdots U_1$, then including recovery for V gate implementations towards the end of the $U_l \cdots U_1$ circuit may prove helpful. More research is needed to determine one way or the other.

The upper limit savings for other resource qubits is more modest in comparison. Assuming $(C_1, 0) \equiv (\text{CNOT}, 0)$ and $|\psi\rangle = |H\rangle$ to yield $1/\alpha \approx 1.172$, Theorem 7 says that no recovery is upwards to about 17% $|H\rangle$ states more expensive. Direct use of the other magic state $|K\rangle$ in $(\text{CNOT}, 0)$ means $1/\alpha \approx 1.267$, but compared to $|H\rangle$, there are yet to be significant applications that directly use $|K\rangle$, besides the creation of $|\pi/6\rangle$ [16]. This starts from $|K\rangle \otimes |K\rangle$, so our recovery operation is not beneficial in this use case.

6.6 Summary

We conducted an analysis to better understand the influence our nested recovery protocol may have on a quantum computation. We determined the optimal depth at which our protocol performs best, and showed that integrating recovery is conducive to conserving resources. Although the reduction factor is at most two, this applies to the most useful of our two-qubit applications – phase rotations. Even if the changes for one instance are small, the protocol is flexible enough to insert into several places of a larger existing scheme to further improve our resource economy. The savings will accumulate, so the overall reduction can feel substantial. We present a scenario on how the recovery protocol may be incorporated in the next couple chapters, and provide small examples in which our recovery protocol is able to make a slight contribution.

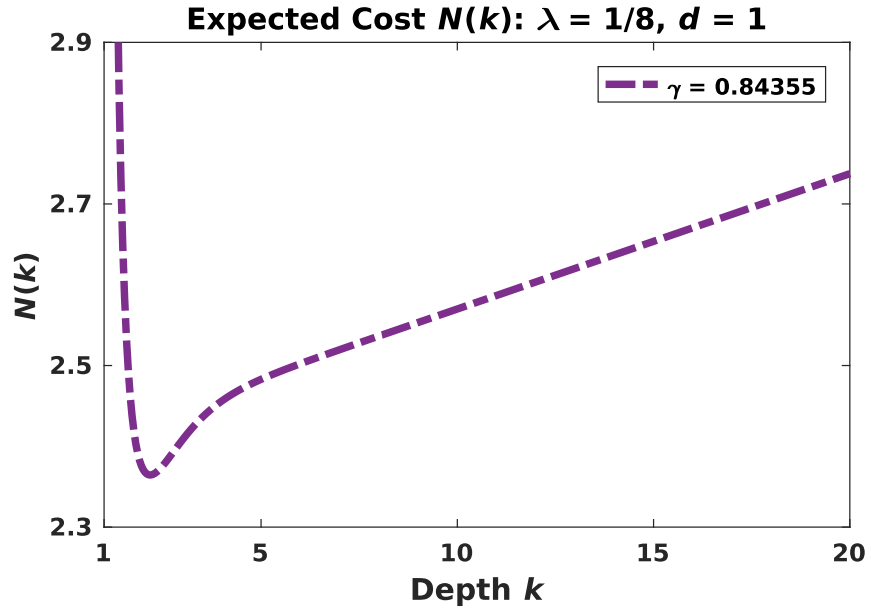
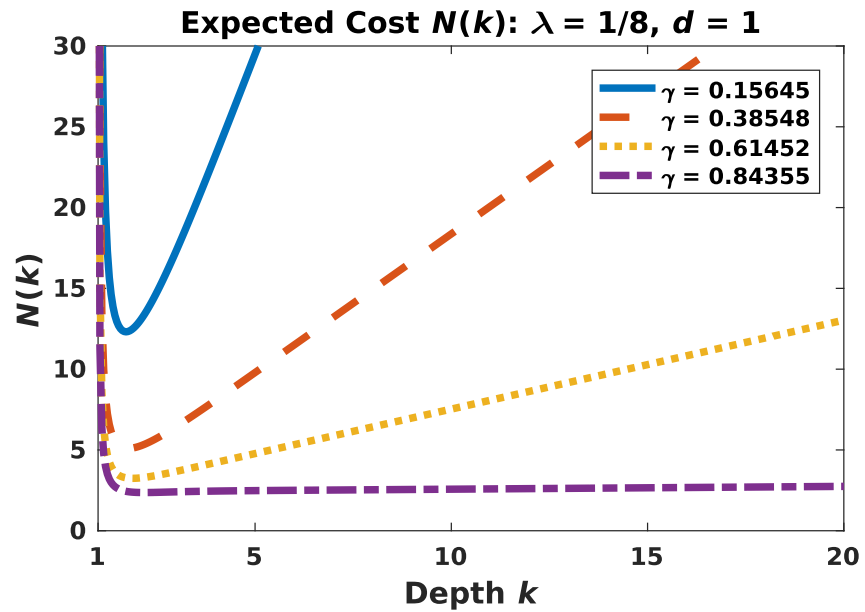
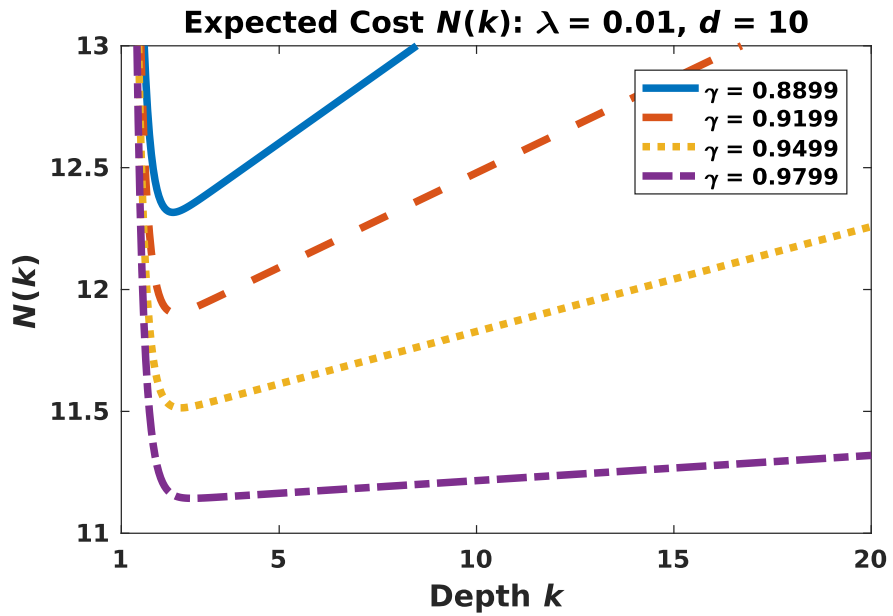
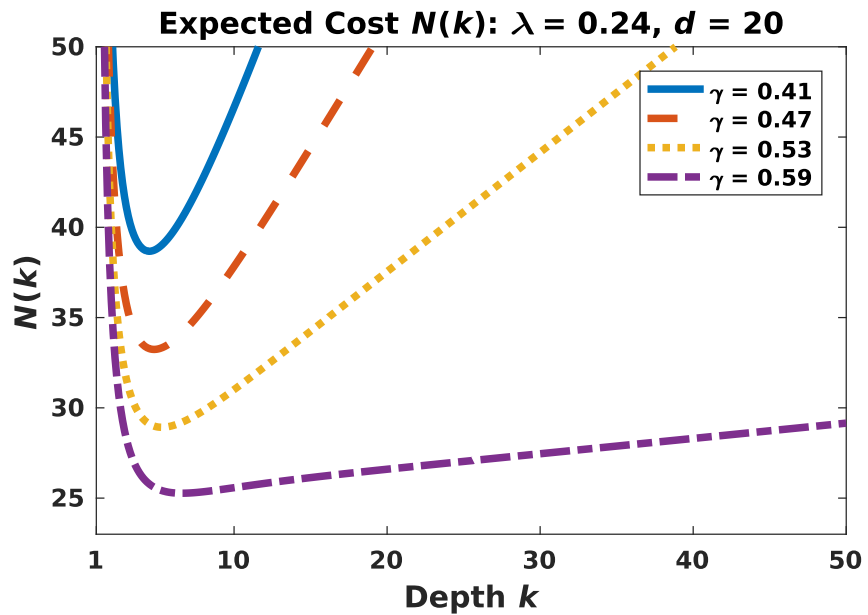


Figure 6.1: This figure contains plots of the expected cost $N(k)$ for protocol instances set to $\lambda = 1/8$ and varying starting probabilities γ . Although the curve of $\gamma = 0.84355$ in (a) appear to reach a constant, the close-up in (b) suggests otherwise. Notice how every curve has a minimum at a point $k > 1$ before a region of continuous increase.



(a)



(b)

Figure 6.2: Additional data of the expected cost $N(k)$ to support the assertion of a single global minimum. The values are generated for protocol instances set to various restricted probability specifications.

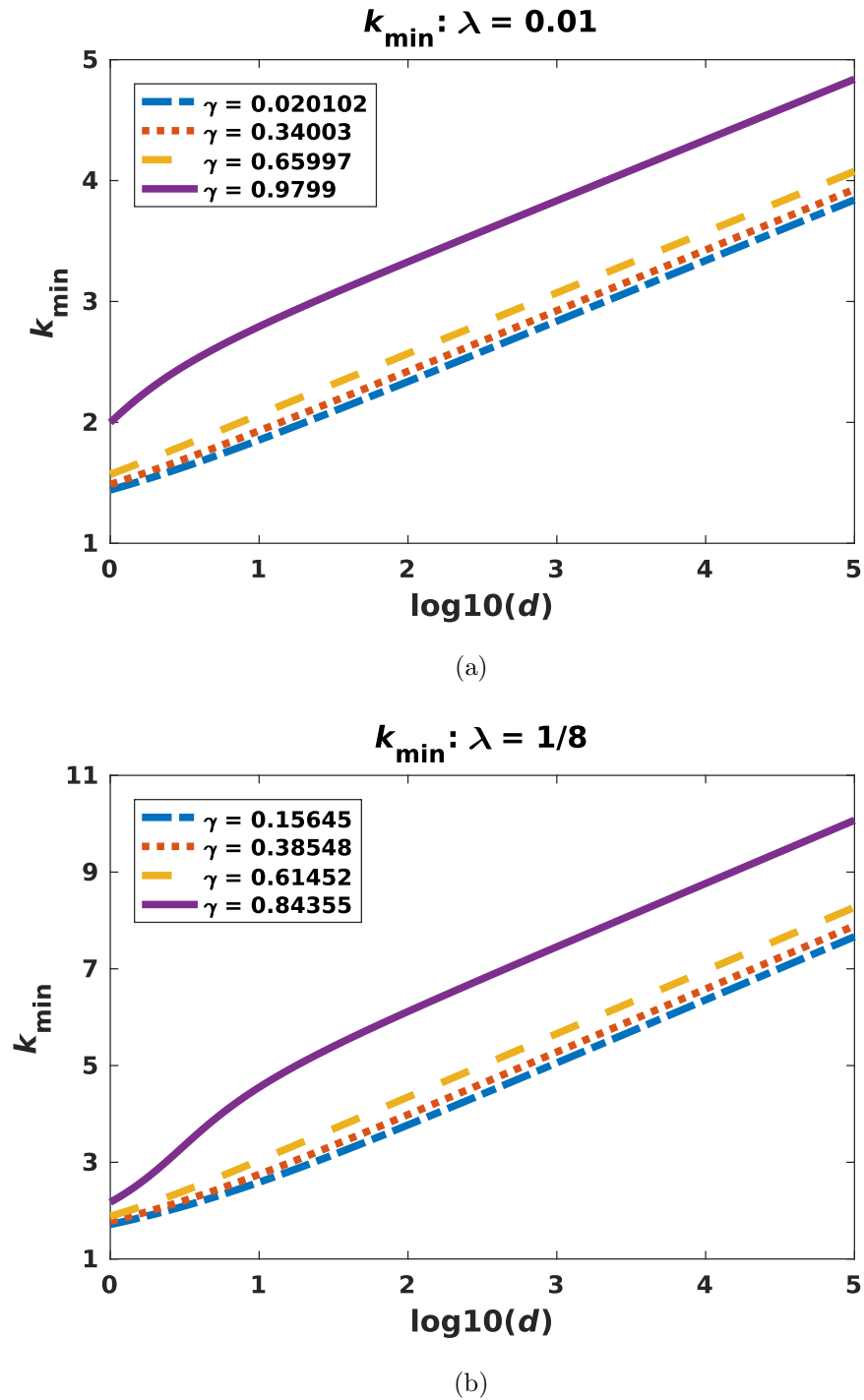


Figure 6.3: The data for k_{\min} suggests a protocol set to a restricted probability specification (λ, γ) should stop at a max depth proportional to $\log(d)$ to keep costs to a minimum, where d is the startup cost.

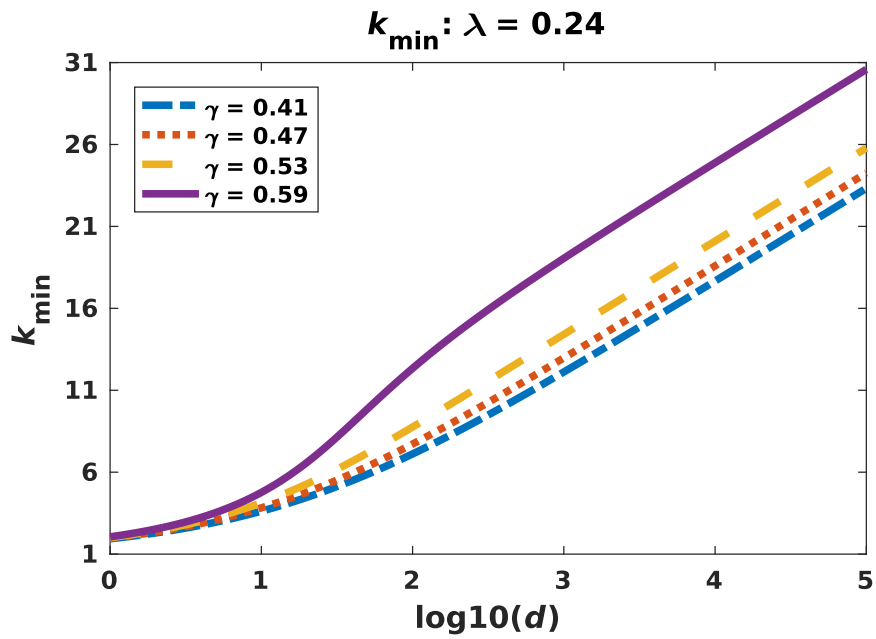
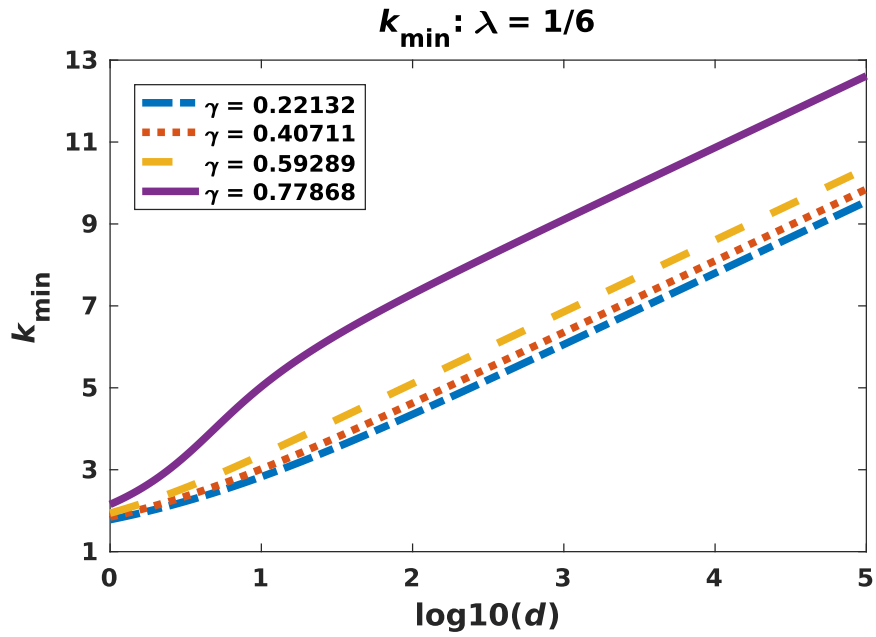


Figure 6.4: Additional data in support of $k_{\min} = \Theta(\log d)$ for a protocol set to a restricted probability specification (λ, γ) and startup cost d .

Chapter 7

Stabilizer Circuits with Binary In-tree Form: Introduction

Given our deeper understanding of two-qubit stabilizer circuits, we may use such objects as the basic building block for larger computations. In particular, we examine postselected n -to-one stabilizer circuits, and the problem of implementing Clifford unitaries as a sequence of smaller Clifford gates that exhibit a “binary in-tree” structure. We show this type of decomposition leads to a more efficient process for generating qubits from n -qubit stabilizer circuits and n -qubit product states. In the next chapter, we describe an algorithm to synthesize Clifford circuits with such “binary in-tree” constructions.

7.1 Case Study: Four-qubit Quantum Circuit

We start with a motivating example. Consider the quantum circuit in Figure 7.1a. Assume that after the U gate, the first qubit is needed for a future computation, while the other three qubits remain idle. If we want to draw a flow diagram to reflect the qubit dependencies in this circuit, one possible illustration is the directed acyclic graph (DAG) in Figure 7.1b. The graph node ϱ_1 encapsulates the first qubit of $U|q_1, q_2, q_3, q_4\rangle$, and the four arrows signify its dependence on the initial $|q_i\rangle$. Because the second to fourth qubits are, in a sense, no longer active after U , there are no corresponding graph nodes

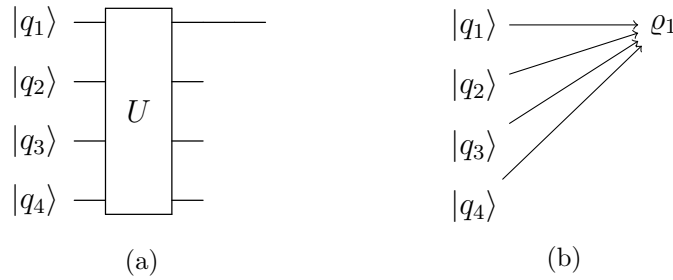


Figure 7.1: (a) Example of a quantum circuit with in-tree form. The second to fourth wires are shortened to indicate the point in time when the qubits become idle. (b) If we interpret the node ϱ_1 as the first qubit after the gate U , we can draw the qubit dependencies as a directed graph that is also an in-tree.

alongside ϱ_1 . We see from Figure 7.1b that the DAG is reminiscent of an *in-tree*: a directed, rooted tree in which the edges are reversed, so that every node has a pointer to its parent but not vice versa [68].¹ Borrowing the usual tree terminology, we say the initial $|q_i\rangle$ are *leaves*, and ϱ_1 is the *root*.

Suppose now that $U = (U_3 \otimes I)(U_1 \otimes U_2)$, where U_i are smaller quantum gates. The new circuit is portrayed in Figure 7.2a. In addition to what we know about the first qubit after U , assume further that U_3 acts on the first and third qubits only after U_1 and U_2 . Some wires in the circuit diagram are deliberately shortened to highlight the inactivity of certain qubits after a period of time. Using the more detailed blueprint in Figure 7.2a, we obtain the DAG in Figure 7.2b, where we add graph nodes ϱ_2 and ϱ_3 to represent the first and third qubits of $(U_1 \otimes U_2) |q_1, q_2, q_3, q_4\rangle$. The DAG in Figure 7.2b now resembles a *binary in-tree*, where each node has at most two incoming references.

Notice how the graph produced between Figures 7.1b and 7.2b rely heavily on the abstraction level or amount of the details that we know (or wish to see). Both are equally valid, and the one we choose to utilize depends on the task at hand. For example, the logical picture in Figure 7.1a is likely better when delivering high level algorithmic

¹Tutte uses the term *arborescence converging to some root*.

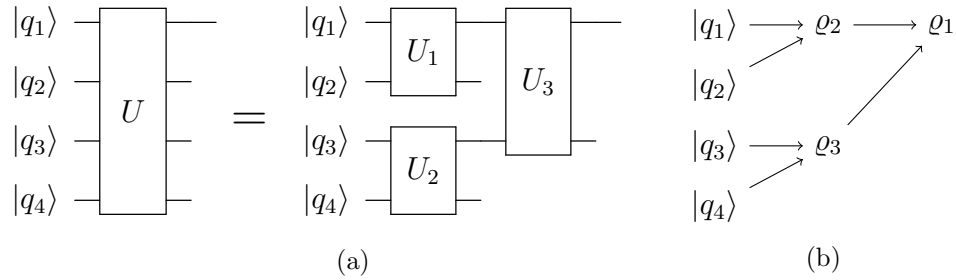


Figure 7.2: If we know $U = (U_3 \otimes I)(U_1 \otimes U_2)$, where each U_i is a nontrivial quantum gate on two qubits only, then we can draw the qubit dependencies as a directed graph resembling a binary in-tree. The nodes q_2 and q_3 are interpreted as the first and third qubits after applying $U_1 \otimes U_2$.

descriptions. On the other hand, the decomposition of U in Figure 7.2a can reveal novel ideas for running the quantum circuit, as we shall explain.

The real benefit to this arrangement comes into play when we apply measurements on qubits two to four. This is perhaps the most convincing element we add to solidify the in-tree image. Such circuits typically have one subset of outcomes that is preferred more than others. If we defer measurements until after U , it is not immediately clear how we may recuperate from a “failure”. As is usually the case, we discard the qubits and start over, preparing four new copies of $|q_i\rangle$. Now consider the alternative where we measure qubit two after applying U_1 . If the outcome is part of a desirable set, we may move forward to the next measurement on qubit four. If, on the contrary, the outcome is part of an undesirable set, then we prepare new copies of $|q_1\rangle$ and $|q_2\rangle$ and rerun the subcircuit again. Notice when we repeat the application of U_1 , we leave $|q_3\rangle$ and $|q_4\rangle$ untouched. If the measurement on qubit two is considered a success, we may hold onto the output until the other subcircuit also yields a positive outcome. And so by executing the quantum circuit in this multistep tree fashion, the parallel subcircuits can run in relative isolation from one another, preventing unsuccessful subroutines from having any ill-effects on the successful ones.

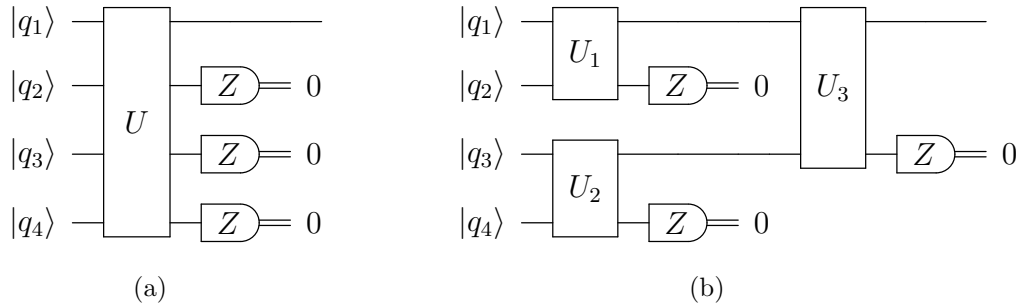


Figure 7.3: Given the decomposition $U = (U_3 \otimes I)(U_1 \otimes U_2)$, we may move up two measurements in circuit (a) to create two independent parallel units in the first half of circuit (b).

7.2 Basic Concepts, Notation, and Review

The meaning of $X^{(i)}$, $Y^{(i)}$, $Z^{(i)}$ is the same: apply the Pauli X , Y , Z gate on qubit i , and the single qubit identity otherwise. We adopt the same convention $H^{(i)}$ and $P^{(i)}$ for the Clifford gates Hadamard and Phase. The symbol I is once again an identity operator whose dimensions fluctuate with context, but will include a tensor superscript to indicate its size when needed i.e. $I^{\otimes k}$ for the k -qubit identity. We let $\mathcal{P}_{\pm}(n)$ be the subset of nontrivial n -qubit Pauli operators with factors ± 1 e.g. $\mathcal{P}_{\pm}(1) = \{\pm X, \pm Y, \pm Z\}$. Since the input size is no longer limited to $n = 2$, we default back to $\text{CNOT}^{(i,j)}$ for the Controlled-NOT with control qubit i and target qubit j . The same applies to $\text{SWAP}^{(i,j)}$ to exchange qubits i and j . Given a Pauli operator g , we define its weight \mathbf{w} to be the number of characters not equal to I e.g. $\mathbf{w}(X \otimes I \otimes Z \otimes Y) = 3$. For convenience, we use $\{\mathbf{n}\}$ to stand for $\{1, \dots, n\}$, a set containing the first n positive integers.

7.3 Postselected and Delegated Two-Op Circuits

To precisely capture our notion of stabilizer circuits with binary in-tree form, we need to reserve special notation with respect to Clifford operators. Specifically, we desire a

system for expressing n -qubit Clifford gates that affect fewer than n qubits. Since every non-leaf node in a binary in-tree has no more than two inbound edges, we have the following definition.

Definition 15 (Clifford Two-operator) *An n -qubit Clifford two-operator $F(i, j)$ is an n -qubit Clifford gate that acts nontrivially on at most two qubits i and j . We assume $i < j$ without loss of generality, and we let F_t stand short for $F_t(i_t, j_t)$.*

We typically use the more compact notation F_t in the presence of a two-operator sequence ($F_k \cdots F_1$ over $F_k(i_k, j_k) \cdots F_1(i_1, j_1)$) unless there is a special circumstance where the indices must be indicated explicitly.

Since the Clifford group is generated by the Controlled-NOT, Hadamard, and Phase gates, every Clifford unitary C can be described by a sequence (circuit) of two-operators $C = F_k \cdots F_1$. Some examples for an $n = 4$ qubit system are $F(1, 4) = \text{CNOT}^{(4,1)}$, $F(1, 3) = H \otimes I \otimes I \otimes I$, and

$$F(2, 4) = P^{(2)} P^{(4)} H^{(2)} H^{(4)} \text{CNOT}^{(2,4)} \text{SWAP}^{(2,4)}. \quad (7.1)$$

If $C = H \otimes H \otimes H \otimes I$, then we must split C across more than one two-operator. One potential breakdown is $C = F(3, 4)F(1, 2)$, where

$$F(1, 2) = H \otimes H \otimes I \otimes I \quad (7.2)$$

$$F(3, 4) = I \otimes I \otimes H \otimes I. \quad (7.3)$$

When $F(i, j)$ is a tensor product of $n - 1$ (or n) single qubit identity gates, one (or both) of the qubit indices can act as a wildcard and chosen according to the situation. To convey this better, consider trying to assign $I \otimes I \otimes H \otimes I$ to a two-operator $F(i, j)$. We require $i = 3$ or $j = 3$, but may select 1, 2, or 4 for the other index. Mainly there

is little consequence to writing $F(1, 3)$, $F(2, 3)$, or $F(3, 4)$ as long as the qubit that is acted upon nontrivially is captured accurately in the notation, and so any of these three should suffice. Unfortunately, the current definition of postselected stabilizer circuit is not quite fitting, so we need similar ideas specially catered for two-operators to handle the current problem.

Definition 16 (Postselected Two-Op Circuit) *A postselected n -qubit two-op circuit $(F(i, j), v)$ is a stabilizer circuit that implements an n -qubit two-operator $F(i, j)$, followed by one Z -measurement on qubit j for an outcome $v \in \{0, 1\}$.*

Definitions for the probability of v and the output are almost identical to Definition 4 for postselected stabilizer circuits. We add the abbreviations “tc” and “sc” to distinguish between the two kinds.

Definition 17 (Probability and Output (TC)) *Let $(F(i, j), v)$ be a postselected n -qubit two-op circuit and let ρ be an n -qubit state. Then the probability Q_v^{tc} of outcome v on the transformed state $F(i, j)\rho F^\dagger(i, j)$ is*

$$Q_v^{\text{tc}}(F(i, j), \rho) = \text{tr} (AF(i, j)\rho F^\dagger(i, j)A^\dagger) \quad (7.4)$$

where $A = I^{\otimes j-1} \otimes |v\rangle\langle v| \otimes I^{\otimes n-j}$. If $Q_v^{\text{tc}}(F(i, j), \rho) > 0$, then the n -qubit output Φ_v^{tc} of a postselected two-op circuit $(F(i, j), v)$ on an input ρ is

$$\Phi_v^{\text{tc}}(C, \rho) = \frac{AF(i, j)\rho F^\dagger(i, j)A^\dagger}{Q_v^{\text{tc}}(F(i, j), \rho)}. \quad (7.5)$$

We set the number of output qubits to be the same as the number of input qubits by leaving in the measured qubit $|v\rangle$. If F_1 and F_2 are n -qubit two-operators, then it is easier to describe a process in which we pass the output ρ_2 of (F_1, v_1) on ρ_1 directly as input

to (F_2, v_2) , where one of the constituent qubits in ρ_2 is $|v_1\rangle$. Unsurprisingly, postselected two-op circuits are effectively postselected two-to-one stabilizer circuits, leading to the next definition.

Definition 18 (Delegated Two-Op Circuit) *Let $(F(i, j), v)$ be a postselected n -qubit two-op circuit, and let g_1, g_2, h_1, h_2 be n -qubit Pauli operators such that*

$$F(i, j)X^{(i)}F^\dagger(i, j) = g_1, \quad F(i, j)Z^{(i)}F^\dagger(i, j) = h_1, \quad (7.6)$$

$$F(i, j)X^{(j)}F^\dagger(i, j) = g_2, \quad F(i, j)Z^{(j)}F^\dagger(i, j) = h_2. \quad (7.7)$$

Note that by definition of two-operators, that the weights of $g_1, g_2, h_1,$ and h_2 are at most two. Then a delegated two-op circuit of $(F(i, j), v)$ is a postselected two-to-one stabilizer circuit $(C_{F(i, j)}, v)$ such that

$$C_{F(i, j)}(X \otimes I)C_{F(i, j)}^\dagger = g_{1, i} \otimes g_{1, j} \quad (7.8)$$

$$C_{F(i, j)}(Z \otimes I)C_{F(i, j)}^\dagger = h_{1, i} \otimes h_{1, j} \quad (7.9)$$

$$C_{F(i, j)}(I \otimes X)C_{F(i, j)}^\dagger = g_{2, i} \otimes g_{2, j} \quad (7.10)$$

$$C_{F(i, j)}(I \otimes Z)C_{F(i, j)}^\dagger = h_{2, i} \otimes h_{2, j}. \quad (7.11)$$

We now lay the foundations for describing Clifford unitaries that are implemented by stabilizer circuits with a binary in-tree figure.

7.4 Binary In-tree Decomposition

We only consider n -qubit stabilizer circuits with $n - 1$ Pauli Z -measurements and postselection to produce a single qubit output. For the moment, we limit the input to n -qubit product states $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$. Given a postselected stabilizer circuit (C, v) ,

we desire a Clifford circuit $C = F_k \cdots F_1$, where F_t are two-operators, that displays the same in-tree qualities as Figure 7.2a. If C does not permit such a decomposition, then we may consider an equivalent circuit $(C', v') \equiv (C, v)$ whose unitary C' does have such an implementation. Using Lemma 1, we may assume $v = v' = 0^{n-1}$ and simplify the equivalence between postselected stabilizer circuits.

Definition 19 (Equivalent Clifford Unitaries) *Two n -qubit Clifford unitaries C_1 and C_2 are equivalent with respect to the first qubit if and only if*

$$(I \otimes \langle 0^{n-1} |) C_1 \rho C_1^\dagger (I \otimes |0^{n-1}\rangle) = (I \otimes \langle 0^{n-1} |) C_2 \rho C_2^\dagger (I \otimes |0^{n-1}\rangle) \quad (7.12)$$

holds for all n -qubit states ρ . We denote this equivalence by $C_1 \sim_1 C_2$.

As we explained earlier, our circumstances might recommend one arrangement of Clifford gates over the other: if $C_1 \sim_1 C_2$ but C_2 has a tree-like configuration, then there may be greater benefits to the circuit of C_2 when accounting for measurements. To forge a concrete definition of binary in-tree form, we re-examine the quantum circuit in Figure 7.2a. We recognize right away that a crucial element of the in-tree structure is related to qubit participation. That is, as time progresses, certain qubits must enter an idle condition while other qubits remain part of the later computation. To help with this distinction, we first introduce the following term.

Definition 20 (Inactive Qubit) *Consider an n -qubit two-operator sequence $F_k \cdots F_1$. Then the q th qubit is inactive relative to $F_k \cdots F_1$ if and only if $q \notin \{i_1, j_1, \dots, i_k, j_k\}$.*

The definition of inactivity will allow us to start characterizing Clifford unitaries with the decomposition of a binary in-tree. The problem is that Definition 20 applies to qubits only, and we need to establish another feature that covers the gates within the sequence.

Definition 21 (Binary Connected) *A two-operator sequence $F_k \cdots F_1$ is binary connected if and only if qubit i_t or qubit j_t is inactive relative to the subsequence $F_k \cdots F_{t+1}$ for all indices $t \in \{\mathbf{k} - \mathbf{1}\}$.*

Note that we may always modify each F_t slightly so that qubit j_t is the inactive qubit. The last matter we should address is the sequence length. We already know that after every F_t , we disregard one qubit due to inactivity. Eventually, those qubits will undergo measurements. To increase modularity in the circuit, the Section 7.1 study suggests we spread measurements across different places whenever possible so as to create multiple stages or in-tree layers. Given the previous definitions, we can easily identify suitable locations for a measurement: it is precisely after every two-operator F_t on the inactive qubit. Since there are $n - 1$ measurements total, we arrive at the final requirement.

Definition 22 (Binary In-tree Clifford Unitary) *A Clifford unitary C on $n > 1$ qubits is a binary in-tree unitary if and only if there are $n - 1$ two-operators F_t such that $C = F_{n-1} \cdots F_1$ is binary connected and $i_{n-1} = 1$.*

A binary in-tree decomposition, or a Clifford circuit with binary in-tree form, thus corresponds to a binary in-tree Clifford unitary. If the sequence length k is smaller than $n - 1$, then we may supplement the sequence with identity operators $F_t = I$ to fill the gap. The indices i_t and j_t are chosen to satisfy the binary connected constraint. We propose this solution only to accommodate the $n - 1$ measurements by giving a clear indication of which qubit to measure following each two-operator. If $k \geq n$ and each $F_t \neq I$, then the two-operator sequence cannot be in binary in-tree form. Consequently, operations like $C = G_1 \otimes \cdots \otimes G_n$, where G_i are single qubit Clifford gates, are guaranteed to have at least one binary in-tree decomposition. For such unitaries, the pairing of single qubit Clifford gates to F_t can be arbitrary. Figure 7.4 shows a couple legal possibilities. Note that since we currently restrict inputs to n -qubit product states $\varphi_1 \otimes \cdots \otimes \varphi_n$, the output

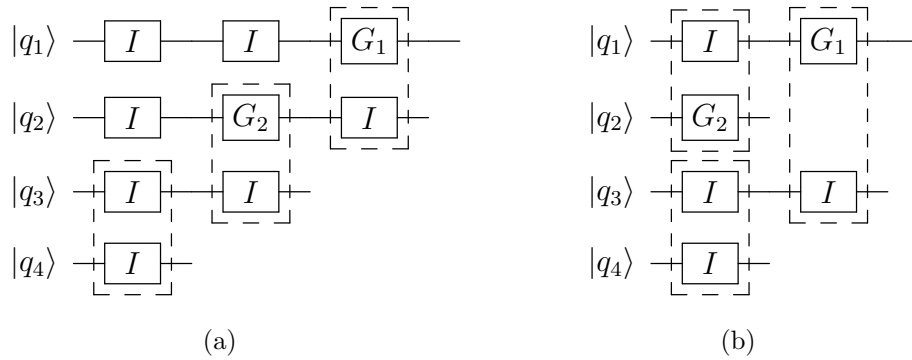


Figure 7.4: The dashed boxes represent two-operators. When the circuit consists of many single qubit Clifford gates, there are no pairing restrictions, hence both (a) and (b) sequences are valid and implement the same operation.

is trivially $G_1\varphi_1G_1^\dagger$ for such occasions, and we may actually simplify the circuit to a one-qubit procedure. As a result, our primary interest are binary connected sequences in which every F_t has a Controlled-NOT.

In Definition 22, we also enforce $i_{n-1} = 1$ so that qubit one forms the tree root and becomes the output when we add measurements on qubits two to n . This is not mandatory for a Clifford circuit to demonstrate an in-tree structure, but does simplify the criteria. Besides, we can always permute the qubits prior to measurement while maintaining the in-tree form, so demanding $i_{n-1} = 1$ is acceptable. For brevity, we write $C = F_{n-1} \cdots F_1$ for a binary in-tree unitary C with the assumption that $i_{n-1} = 1$.

7.5 Multistep Tree Execution and Expected Cost

By now, we should be familiar with the *multistep tree* execution protocol for a binary connected circuit sequence: we perform a Z -measurement after every two-operator F_t on the inactive qubit and proceed to F_{t+1} when the measurement is successful. If unsuccessful, we repeat the prerequisite computations leading up to F_t . This obviously excludes any subcircuits that may run independently (see Figure 7.2a for an example).

For any n -qubit binary in-tree Clifford unitary $C = F_{n-1} \cdots F_1$, the new strategy converts a procedure implemented by the postselected stabilizer circuit $(C, 0^{n-1})$ into a series of postselected two-op circuits $(F_1, 0), \dots, (F_{n-1}, 0)$. We shall assume that for each F_t , that qubit j_t is inactive. Formally, the main step consists of the following instructions.

Multistep Tree Execution:

1. Let $\rho_t = \varphi_1 \otimes \cdots \otimes \varphi_n$ be an n -qubit state. If $t = 1$, then ρ_1 is the initial state; if $t > 1$, then ρ_t is the output of $(F_{t-1}, 0)$ on ρ_{t-1} . Apply F_t on ρ_t and perform a Z -measurement on the inactive qubit j_t .
2. If the outcome is 0, proceed to the next two-operator (if any left).
3. If the outcome is 1, prepare another ρ_t by replacing qubits i_t and j_t with new instances of φ_{i_t} and φ_{j_t} . The other $n - 2$ qubits unaffected by F_t are left alone. Repeat Instruction 1.

We soon find that only by following this multistep policy may we extract any real benefit from the in-tree structure. On top of that, the recovery technique of Chapter 5 may be applicable whenever one of $n - 1$ measurements is unsuccessful. If we do not execute the procedure in a multistep manner, then we default to the *basic* single step process.

Basic Execution:

1. Let $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$ be the initial n -qubit state. Apply C on ρ and perform Z -measurements on qubits two to n .
2. If the outcome is not 0^{n-1} , prepare a new copy of ρ and repeat Instruction 1.

The basic strategy applies to all Clifford unitaries C , regardless of whether or not C has a binary in-tree implementation.

Given $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$, we now compare the expected cost of both approaches to create an output qubit $\varphi' = \Phi_{0^{n-1}}^{\text{sc}}(C, \rho)$. Neither is difficult to determine. We explain how to compute the expected cost when dealing with a postselected stabilizer circuit first. As always, we presume there is a fundamental resource necessary to prepare non-stabilizer states e.g. magic state $|H\rangle$, and the cost is measured with respect to such a resource. The price of any stabilizer state is zero.

Definition 23 (Expected Cost (SC)) *Consider a procedure given by a postselected n -to-one stabilizer circuit (C, v) . Let d be the cost of an n -qubit state ρ . Then the expected cost to produce an output qubit φ of (C, v) on ρ is*

$$E_{\text{sc}}(\varphi) = \frac{d}{Q_v^{\text{sc}}(C, \rho)}. \quad (7.13)$$

The expected cost of an individual postselected two-op circuit is not much different.

Definition 24 (Expected Cost (TC)) *Consider a procedure given by a postselected n -qubit two-op circuit $(F(i, j), v)$. Let $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$ be an n -qubit state, and let d_i and d_j be the cost of φ_i and φ_j , respectively. Then the expected cost to produce an n -qubit output ρ' of $(F(i, j), v)$ on ρ is*

$$E_{\text{tc}}(\rho') = \frac{d_i + d_j}{Q_v^{\text{tc}}(F(i, j), \rho)}. \quad (7.14)$$

Observe that the expected cost for a two-operator based process only depends on the initial costs of the two qubits φ_i and φ_j . The expected cost of ρ' is also computable via a delegated two-op circuit, which we denote E_{dc} (instead of E_{sc}).

Proposition 3 (Wong, unpublished) *Suppose there are two procedures, one based on a postselected two-op circuit $(F(i, j), v)$, and another based on its delegated two-op circuit*

$(C_{F(i,j)}, v)$. Let $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$ be an n -qubit state, and let d_i and d_j be the cost of φ_i and φ_j , respectively. Then the expected costs E_{sc} and E_{dc} to produce an output ρ' of $(F(i, j), v)$ on ρ and an output φ' of $(C_{F(i,j)}, v)$ on $\varphi_i \otimes \varphi_j$ are the same:

$$E_{\text{tc}}(\rho') = E_{\text{dc}}(\varphi'). \quad (7.15)$$

Proof: The proof is quite trivial. We may assume $i = 1$ and $j = 2$ without loss of generality. Then $F(i, j) = C \otimes I^{\otimes n-2}$, where C is a two-qubit Clifford unitary. The rest is easy to see. ■

In the end, the multistep process translates to lower expected cost.

Theorem 8 (Wong, unpublished) *Let φ' be the output of a postselected n -to-one stabilizer circuit $(C, 0^{n-1})$ on an n -qubit state $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$. Suppose $C = F_{n-1} \cdots F_1$ is a binary in-tree Clifford unitary. Then the expected cost to produce $\varphi' \otimes |0^{n-1}\rangle\langle 0^{n-1}|$ under the multistep tree execution strategy is at most the expected cost to produce φ' under the basic strategy.*

Proof: Let d_i be the cost of φ_i . Without loss of generality, assume $n = 3$, since the expected cost calculations are recursive. As such, assume also that $i_1 = 2$ and $j_1 = 3$ for the first two-operator F_1 , and that $j_2 = 2$. Let ρ' be the output of $(F_1, 0)$ on ρ . To improve readability, set $p_1 = Q_0^{\text{tc}}(F_1, \rho)$ and $p_2 = Q_0^{\text{tc}}(F_2, \rho')$. Then

$$E_{\text{tc}}(\varphi' \otimes |0^{n-1}\rangle\langle 0^{n-1}|) = \frac{d_1 + \frac{d_2 + d_3}{p_1}}{p_2} \quad (7.16)$$

$$= \frac{d_1 p_1 + d_2 + d_3}{p_1 p_2} \quad (7.17)$$

$$\leq \frac{d_1 + d_2 + d_3}{p_1 p_2} = E_{\text{sc}}(\varphi') \quad (7.18)$$

since $p_1 p_2 = Q_0^{\text{sc}}(C, \rho)$. This argument extends naturally to larger values of n . ■

By the time we reach the final postselected two-op circuit, $n - 2$ measurements would have passed, meaning the majority of the $\varphi_1 \otimes \cdots \otimes \varphi_n$ input for $(F_{n-1}(1, j_{n-1}), 0)$ are $|0\rangle$ states. The relevant qubits φ_1 and $\varphi_{j_{n-1}}$ are likely outputs of other delegated two-op circuits themselves. As a result, the evaluation of qubit costs d_1 and $d_{j_{n-1}}$ will expand into Equation 7.14, much like in the proof of Theorem 8. Examples of expected cost improvements under the multistep tree policy are provided in Chapter 8.

Chapter 8

Synthesizing Stabilizer Circuits with Binary In-tree Form

With the groundwork in Chapter 7, we aim to synthesize n -qubit stabilizer circuits with binary in-tree form. For that reason, we present a classical algorithm to assemble binary connected sequences of Clifford gates. That is to say, given an input Clifford unitary C , the algorithm returns a Clifford circuit implementing a binary in-tree unitary C_{bt} such that $C_{\text{bt}} \sim_1 C$, if one exists. Otherwise, there is no such equivalent operation. In that case, any previously known algorithm is sufficient to construct a circuit e.g. Aaronson and Gottesman [3] that does not take on the form of a binary in-tree. Our solution is efficient from the computational complexity standpoint: it completes in worst-case polynomial time $O(n^5)$, where n is the number of qubits. Afterwards, we apply the algorithm on some examples to demonstrate the improvements in expected resource cost for a few quantum processes.

8.1 Basic Property of Binary In-tree Unitaries

Unfortunately, not all Clifford unitaries are susceptible to a binary in-tree decomposition. To determine if such a binary in-tree form circuit exists, we can build a test around the following quality.

Lemma 10 *Let $C = F_{n-1} \cdots F_1$ be an n -qubit binary in-tree Clifford unitary. For all indices $t \in \{\mathbf{n} - \mathbf{1}\}$, there is an n -qubit Pauli operator $g \in \langle C_t^\dagger Z^{(2)} C_t, \dots, C_t^\dagger Z^{(n)} C_t \rangle$ such that $1 \leq \mathbf{w}(g) \leq 2$, where $C_t = F_{n-1} \cdots F_t$.*

Proof: Keep in mind that $i_{n-1} = 1$. Since we will postselect on the 0^{n-1} outcome, we follow the stabilizer group $\langle Z^{(2)}, \dots, Z^{(n)} \rangle$. We prove the statement by enumerating in the reverse order $t = n - 1$ to 1, and checking the $C_t^\dagger Z^{(k)} C_t$ generators at each step t . Clearly, we have

$$1 \leq \mathbf{w} \left(F_{n-1}^\dagger Z^{(n-1)} F_{n-1} \right) \leq 2 \quad (8.1)$$

when $t = n - 1$. Suppose this holds for all indices greater than t . Then for the current iteration t , assume without loss of generality that between qubits i_t and j_t of the two-operator F_t , that j_t is the inactive qubit relative to $F_{n-1} \cdots F_{t+1}$. In other words, $j_t \notin \{1, j_{n-1}, \dots, i_{t+1}, j_{t+1}\}$. This means when we arrive at iteration t , the generator $Z^{(j_t)}$ is left untouched by all $C_{t'} = F_{n-1} \cdots F_{t'}$ with $t' > t$. Therefore

$$1 \leq \mathbf{w} \left(C_t^\dagger Z^{(j_t)} C_t \right) = \mathbf{w} \left(F_t^\dagger Z^{(j_t)} F_t \right) \leq 2 \quad (8.2)$$

for every $t \in \{\mathbf{n} - \mathbf{1}\}$. ■

The smallest size n for which there is no equivalent binary in-tree unitary to an interested Clifford operation C is $n = 4$. Figure 8.1 offers a circuit implementing one such instance that performs

$$C^\dagger Z^{(2)} C = Y \otimes X \otimes Y \otimes X, \quad (8.3)$$

$$C^\dagger Z^{(3)} C = Z \otimes I \otimes X \otimes X, \quad (8.4)$$

$$C^\dagger Z^{(4)} C = X \otimes Y \otimes X \otimes Y. \quad (8.5)$$

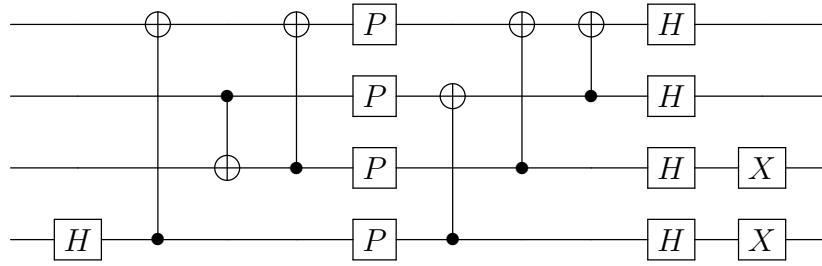


Figure 8.1: The unitary implemented by the Clifford circuit above does not have an equivalent operation with a binary in-tree decomposition.

It is easy to see that each Pauli operator in the stabilizer group generated by these three elements has weight at least three.

8.2 Stabilizer Matrices and Stabilizer Matrix Forms

Lemma 10 provides a basis for an algorithm to synthesize Clifford circuits with binary in-tree form. To that end, we rely on objects which we refer to as *stabilizer matrices*. Recall from Section 2.1 that the n -qubit Pauli group $\mathcal{P}(n)$ contains several stabilizer groups \mathcal{S} . If \mathcal{S} is generated by k independent and commuting n -qubit Pauli operators s_i , then we can store the generators as rows in a $k \times n$ matrix

$$R = \begin{bmatrix} s_{1,1} & \cdots & s_{1,n} \\ \vdots & \ddots & \vdots \\ s_{k,1} & \cdots & s_{k,n} \end{bmatrix} \tag{8.6}$$

which we call a *stabilizer matrix* on $\{s_1, \dots, s_k\}$. We may view R as an alternative to the $2^n \times 2^n$ density matrix of a stabilizer (mixed) state ρ . Such representations enable an efficient simulation of stabilizer circuits with ρ as input. The Controlled-NOT, Hadamard, and Phase gates translate to column operations on R and follow the

conjugation rules in Equations 2.14 and 2.15. Row operations can be an interchange, or component-wise multiplication of Pauli gates from one row to another, both of which leave the group \mathcal{S} – and ρ – unchanged. Throughout our use of stabilizer matrices, the terms row and generator, and column and qubit are interchangeable. The weight function \mathbf{w} accepts as input both rows and columns of Pauli operators from R .

There are efficient procedures to produce various matrix forms on the stabilizer group generators. A familiar one is Audenaert and Plenio’s row echelon form [8], which is effectively the analog of row echelon form for linear algebra matrices.¹ The algorithm to obtain a stabilizer matrix in row echelon form is also similar to Gauss-Jordan elimination. In the next definition, the *leading Pauli gate* of an n -qubit row $g = g_1 \otimes \cdots \otimes g_n$ is the leftmost gate $g_i \in \mathcal{P}_{\pm}(1)$ (i.e. $g_i \neq I$). A *trivial row* means the n -qubit identity.

Definition 25 (Row Echelon Form) *A stabilizer matrix is in row echelon form if*

1. *all nontrivial rows are above all trivial rows*
2. *the leading Pauli gate of a nontrivial row is either*
 - i. *to the right of the leading Pauli gate in the row above, or*
 - ii. *directly below but anticommutes with the leading Pauli gate in the row above.*

An example stabilizer matrix in row echelon form is

$$R = \begin{bmatrix} \boxed{X} & Z & X & Y & Y & Y & Y \\ I & I & \boxed{X} & X & X & Z & Z \\ I & I & I & \boxed{Z} & Z & X & X \\ I & I & I & I & I & I & I \end{bmatrix} \quad (8.7)$$

where the solid boxes indicate the leading Pauli gates from each row.

¹In their paper [8], Audenaert and Plenio use the term *row-reduced echelon form* instead.

For our work, we add another round of row operations after placing a stabilizer matrix in row echelon form. Like in Gauss-Jordan elimination, the second round starts from the bottom-most generator and works its way back up. If the i th row is $g = g_1 \otimes \cdots \otimes g_n \neq I$, and its leading Pauli gate is g_j for some column j , then the second round eliminates all instances of g_j above row i in column j . This is illustrated in an example below, where we use the third row operator to eliminate the Z in the first row (dashed box):

$$\begin{bmatrix} X & X & \boxed{Z} & \cdots \\ I & X & X & \cdots \\ I & I & \boxed{Z} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \implies \begin{bmatrix} X & X & \boxed{I} & \cdots \\ I & X & X & \cdots \\ I & I & \boxed{Z} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (8.8)$$

When the matrix has anticommuting leading Pauli gates g_j and h_j in rows i and $i + 1$, then we clear all nontrivial Pauli gates above row i along column j . For instance, we remove all X , Y and Z characters (dashed box) from column three below:

$$\begin{bmatrix} Z & I & I & \boxed{X} & Z & \cdots \\ I & X & I & \boxed{Z} & X & \cdots \\ I & I & Z & \boxed{Y} & Y & \cdots \\ I & I & I & \boxed{X} & Z & \cdots \\ I & I & I & \boxed{Z} & X & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \implies \begin{bmatrix} Z & I & I & \boxed{I} & I & \cdots \\ I & X & I & \boxed{I} & I & \cdots \\ I & I & Z & \boxed{I} & I & \cdots \\ I & I & I & \boxed{X} & Z & \cdots \\ I & I & I & \boxed{Z} & X & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix}. \quad (8.9)$$

In this manner, we transform the stabilizer matrix to resemble closer to the reduced row echelon form of linear algebra matrices.

Definition 26 (Reduced Row Echelon Form (RREF)) *A stabilizer matrix is said to be in reduced row echelon form (RREF) if*

1. *it is in row echelon form*
2. *every leading Pauli gate g is the only g in its column*
3. *every anticommuting pair of leading Pauli gates g and h from the same column are the only nontrivial Pauli gates in its column.*

The following is an example stabilizer matrix in reduced row echelon form:

$$R = \begin{bmatrix} X & I & I & I & X & X \\ I & Z & I & I & I & I \\ I & I & Z & Z & X & X \\ I & I & X & X & Z & Z \\ I & I & I & I & I & I \end{bmatrix}. \quad (8.10)$$

While row echelon form facilitates the identification of independent generators, RREF facilitates the identification of weight one generators. It is easy to see how the second assertion holds. Suppose there is an n -qubit Pauli operator $g \in \mathcal{S} = \langle s_1, \dots, s_k \rangle$ such that $\mathbf{w}(g) = 1$. Let j be an index such that $g_j \in \mathcal{P}_{\pm}(1)$. When the stabilizer matrix is in row echelon form, there will be one operator g' in row i such that $g'_j = g_j$. If $g' \neq g$, then g' is the product of g along with Pauli operators contained in rows below i . As such, there exists one or more columns $k > j$ such that $g'_k \in \mathcal{P}_{\pm}(1)$. The second round eliminates all such gates g'_k from row i , eventually changing g' to g .

A natural side effect is that at the conclusion of these modifications, when g' has become g , that column j in the stabilizer matrix will also be of weight one. An example can be seen in Equation 8.10, where we have one Z in the second row and column of R . Due to the special role that weight one generators will play in our forthcoming synthesis algorithm, we have the following definition.

Definition 27 (Doubly Weight One (dw-one) Entry) *Let R be a stabilizer matrix. An entry $R_{i,j}$ in row i and column j is doubly weight one (dw-one) if and only if row i and column j are weight one and $R_{i,j} \in \mathcal{P}_{\pm}(1)$.*

This will be useful in identifying other Pauli operators of weight greater than one.

8.3 Synthesis of Binary In-tree Clifford Circuits

We present Algorithm `BinaryTree` to synthesize Clifford circuits with binary in-tree form, using stabilizer matrices to obtain such circuits. In short, we search for a Pauli operator g of weight $\mathbf{w}(g) = 2$ belonging to a particular stabilizer group generated by $n - 1$ independent and commuting n -qubit Pauli operators. If such an element g exists, then we know which two qubits i and j to apply a two-operator $F(i, j)$. If in addition g is dependent on two Pauli operators h_1 and h_2 of the group such that $\mathbf{w}(h_1) = \mathbf{w}(h_2) = 1$, then $F(i, j)$ consists entirely of single qubit Clifford gates. If not, $F(i, j)$ needs at least one Controlled-NOT. Most of the effort goes into deciding whether the sequence requires a two-operator of the latter kind; the details are found in Subroutine `BinConnSeq`. Before we get there, we introduce some smaller procedures that will appear in the main portion of the algorithm. Pseudocode for the majority of subroutines is provided.

8.3.1 Synthesis Algorithm

First we have `Commute`(g, S). It determines whether a Pauli operator g commutes with every Pauli operator in S . Usually S will consist of $n - 1$ stabilizer group generators for an n -qubit system. The next subroutine we have is `Disjoint`(g, \mathbb{I}). We start by creating \mathbb{I}' , the set of locations in $g = g_1 \otimes \cdots \otimes g_n$ with nontrivial Pauli gates, and compare it to \mathbb{I} . We only care whether \mathbb{I}' and \mathbb{I} are disjoint.

Subroutine 1:

Input: 1. n -qubit Pauli operator g
 2. set of commuting n -qubit Pauli operators S

Output: True or False

Determine if an n -qubit Pauli operator g commutes with every operator in S .

procedure Commute(g, S)

for $s \in S$ **do**

if $[g, s] \neq 0$ **then return** False

return True

Subroutine 2:

Input: 1. n -qubit Pauli operator $g = g_1 \otimes \cdots \otimes g_n$
 2. set of integers \mathbb{I}

Output: True or False

Determine if there is a single qubit Pauli gate $g_i \in \mathcal{P}_{\pm}(1)$ such that $i \in \mathbb{I}$.

procedure Disjoint($g = g_1 \otimes \cdots \otimes g_n, \mathbb{I}$)

$\mathbb{I}' \leftarrow \{i \in \{\mathbf{n}\} \mid g_i \in \mathcal{P}_{\pm}(1)\}$

return True **if** $\mathbb{I} \cap \mathbb{I}' \neq \emptyset$, **else return** False

We utilize both `Commute` and `Disjoint` in a subroutine `Search(S, \mathbb{I})`, where $S = \{s_1, \dots, s_k\}$ is again a set of stabilizer group generators. Let $\langle S \rangle$ stand for $\langle s_1, \dots, s_k \rangle$, the group generated by the members of S . We first enumerate the Pauli operators $g = g_1 \otimes \cdots \otimes g_n \in \mathcal{P}_{\pm}(n)$ such that $\mathbf{w}(g) = 2$ and filter them into a separate set called *candidates*. Then for each candidate, we ignore it if it does not commute with each member of S , or has a nontrivial Pauli gate g_i at an index $i \in \mathbb{I}$. We shall see later that \mathbb{I} is connected to the matrices $h \in \langle S \rangle$ of weight $\mathbf{w}(h) = 1$. That is, given such an operator $h = h_1 \otimes \cdots \otimes h_n$, if $h_j \in \mathcal{P}_{\pm}(1)$, then $j \in \mathbb{I}$.

Subroutine 3:

Input: 1. set of independent and commuting n -qubit Pauli operators S
 2. set of integers \mathbb{I}

Output: an n -qubit Pauli operator g satisfying (1) and (2) below,
 or null if no such operator exists

Search for an n -qubit Pauli operator $g = g_1 \otimes \cdots \otimes g_n \in \langle S \rangle$ such that:

1. $\mathbf{w}(g) = 2$
2. g is independent of the elements $s \in \langle S \rangle$ such that $\mathbf{w}(s) = 1$.

procedure Search(S, \mathbb{I})

for $g \in \text{candidates} \leftarrow \{g \in \mathcal{P}_{\pm}(n) \mid \mathbf{w}(g) = 2\}$ **do**
 if not Commute(g, S) or not Disjoint(g, \mathbb{I}) **then skip**
 $R' \leftarrow \text{RREF}(\text{StabilizerMatrix}(S \cup \{g\}))$
 if row($R', |S| + 1$) = $I^{\otimes n}$ **then return** g
return null

For those $g \in \text{candidates}$ that do pass the initial **if** guard, we decide if each is dependent only on those operators $h \in \langle S \rangle$ of weight $\mathbf{w}(h) > 1$. We do this by creating a new stabilizer matrix R' on the combined set $S \cup \{g\}$ and placing R' in RREF. We know g is dependent if the last row of R' is the n -qubit identity:

$$\text{RREF} \left(\begin{pmatrix} \begin{bmatrix} s_{1,1} & \cdots & s_{1,n} \\ \vdots & \ddots & \vdots \\ s_{k,1} & \cdots & s_{k,n} \\ g_1 & \cdots & g_n \end{bmatrix} \end{pmatrix} \right) = \begin{bmatrix} s'_{1,1} & \cdots & s'_{1,n} \\ \vdots & \ddots & \vdots \\ s'_{k,1} & \cdots & s'_{k,n} \\ I & \cdots & I \end{bmatrix} \quad (8.11)$$

at which point we return g . Multiple satisfying operators may exist in **candidates**, but we will touch on this more in Lemma 11. We use $\text{row}(R, i)$ to return the i th row of R .

Supposing $\text{Search}(S, \mathbb{I})$ successfully returns an operator $g \in \langle S \rangle$ such that $\mathbf{w}(g) = 2$, let i and j be the indices of g with nontrivial single qubit Pauli gates. Then during $\text{Select}(S, \mathbb{I})$, we simply pick an appropriate two-operator $F(i, j)$ that maps the weight two g to a weight one $Z^{(j)}$ under conjugation: $F(i, j)gF^\dagger(i, j) = Z^{(j)}$. If no such g exists, then we signal an **error**. This will propagate to the calling Subroutine BinConnSeq and consequently causes all activity to halt. The **error** indicates that we are unable to continue and build a binary connected two-operator sequence.

Subroutine 4:

Input: 1. set of independent and commuting n -qubit Pauli operators S
 2. set of integers \mathbb{I}

Output: a two-operator $F(i, j)$, or an **error**

Select an n -qubit two-operator $F(i, j)$ satisfying certain constraints with respect to a set of n -qubit Pauli operators S and a set of integers \mathbb{I} .

procedure $\text{Select}(S, \mathbb{I})$

result $\leftarrow \text{Search}(S, \mathbb{I})$

if **result** = **null** **then** **halt with error**

$g = g_1 \otimes \cdots \otimes g_n \leftarrow \text{result}$

Let $i, j \in \{\mathbf{n}\} \setminus \mathbb{I}$ be indices such that $g_i, g_j \in \mathcal{P}_\pm(1)$ and $i < j$

return $F(i, j)$ such that $F(i, j)gF^\dagger(i, j) = Z^{(j)}$

The procedure $\text{BinConnSeq}(C)$ represents the core of the algorithm. Given an n -qubit Clifford unitary C , the objective is to produce a binary connected circuit sequence $C_1 = F_k \cdots F_1$ with the following property: there are single qubit Clifford gates $C_2 = G_1 \otimes \cdots \otimes G_n$ such that $C_2 C_1 \sim_1 C$. The procedure fails if Select signals an error, which occurs whenever Search is unable to find a weight two Pauli operator fitting our requirements.

In the beginning, we create R , a stabilizer matrix in RREF on the set of Pauli operators $\{C^\dagger Z^{(2)}C, \dots, C^\dagger Z^{(n)}C\}$. We next examine the number of weight one rows in R . This determines whether the binary connected sequence needs a two-operator with a Controlled-NOT. The answer is an affirmative if we enter the **while** loop, at which point **Select** tries to return an appropriate $F(i, j)$. The first argument we pass to **Select** is a set of the $n - 1$ matrix rows, obtained through **rows**, while the second argument is the columns corresponding to the dw-one entries of R .

Subroutine 5:

Input: an n -qubit Clifford unitary C

Output: a binary connected two-operator sequence, or an **error**

Try to build a binary connected two-operator sequence $C_1 = F_k \cdots F_1$ such

that $(G_1 \otimes \cdots \otimes G_n)C_1 \sim_1 C$, where G_i are single qubit Clifford gates.

procedure BinConnSeq(C)

Setup: $C_1 \leftarrow I^{\otimes n}$

$R \leftarrow \text{RREF}(\text{StabilizerMatrix}(\{C^\dagger Z^{(2)}C, \dots, C^\dagger Z^{(n)}C\}))$

while $|\{i \in \{\mathbf{n} - 1\} \mid \mathbf{w}(\text{row}(R, i)) = 1\}| < n - 1$ **do**

$F(i, j) \leftarrow \text{Select}(\text{rows}(R), \{j \in \{\mathbf{n}\} \mid R_{i,j} \text{ is dw-one}\})$

Update sequence: $C_1 \leftarrow F(i, j)C_1$

Update matrix: $R \leftarrow \text{RREF}(\text{ApplyClifford}(R, F(i, j)))$

Let $j \in \{\mathbf{n}\}$ be the column such that $\text{col}(R, j) = I^{\otimes n-1}$

if $j \neq 1$ **then** **Update sequence:** $C_1 \leftarrow \text{SWAP}^{(1,j)}C_1$

return $C_1 = F_k \cdots F_1$

The next step $\text{ApplyClifford}(R, F(i, j))$ performs column operations on R in accordance with $F(i, j)$ from **Select**. Afterwards, we place the updated R in RREF and repeat until all rows have weight one. Since R is a stabilizer matrix on $n - 1$ independent

Pauli operators, there will be one column of I characters. We want this on column one, hence the potential SWAP at the end.

Lemma 11 *Suppose during the execution of Subroutine BinConnSeq on an n -qubit Clifford unitary C , the **while** loop completed k rounds of updates on the stabilizer matrix R . Let $R^{[t]}$ be the state of R at the end of round t , and let $\mathbb{I}^{[t]} = \{j \in \{\mathbf{n}\} \mid R_{i,j}^{[t]} \text{ is dw-one}\}$ be the columns of the dw-one entries of $R^{[t]}$. Then $\mathbb{I}^{[0]} \subset \mathbb{I}^{[1]} \subset \dots \subset \mathbb{I}^{[k]}$.*

Proof: We start with $R^{[0]}$ already in RREF. On the first iteration $t = 1$, suppose we discover a Pauli operator $g = g_1 \otimes \dots \otimes g_n \in \langle \mathbf{rows}(R^{[0]}) \rangle$ matching our demands. Specifically, $\mathbf{w}(g) = 2$, and its two gates $g_{i_1}, g_{j_1} \in \mathcal{P}_{\pm}(1)$ are located at indices $i_1, j_1 \notin \mathbb{I}^0$. We choose a two-operator F_1 to satisfy $F_1 g F_1^\dagger = Z^{(j_1)}$, so only columns i_1 and j_1 are modified during $\mathbf{ApplyClifford}(R^{[0]}, F_1)$. After RREF finishes to produce $R^{[1]}$, we find that not only do the dw-one entries of $R^{[0]}$ carry into $R^{[1]}$, but that $R^{[1]}$ will have an extra dw-one entry that is absent from $R^{[0]}$. More precisely, $Z^{(j_1)}$ is a row operator of $R^{[1]}$, which means $\mathbb{I}^{[0]} \subset \mathbb{I}^{[1]}$. An example of this occurrence is demonstrated below for $n = 4$, with $g = I \otimes Z \otimes Z \otimes I$ in the solid box, $\mathbb{I}^{[0]} = \{4\}$, and $F_1 = \text{CNOT}^{(2,3)}$:

$$R^{[0]} = \begin{bmatrix} I & X & X & I \\ \boxed{I & Z & Z & I} \\ I & I & I & Z \end{bmatrix} \implies R^{[1]} = \begin{bmatrix} I & X & I & I \\ \boxed{I & I & Z & I} \\ I & I & I & Z \end{bmatrix}.$$

There exists the possibility of another operator $h = h_1 \otimes \dots \otimes h_n \in \langle \mathbf{rows}(R^{[0]}) \rangle$ also of weight $\mathbf{w}(h) = 2$ and Pauli gates $h_{i_1}, h_{j_1} \in \mathcal{P}_{\pm}(1)$. If this is the case, then $F_1 h F_1^\dagger \in \{\pm X^{(i_1)}, \pm Y^{(i_1)}, \pm Z^{(i_1)}\}$ since g and h must commute, and another dw-one entry will appear in column i_1 of $R^{[1]}$. Since we repeat this process on each iteration, it becomes clear that on round t , we enact changes on two columns $i_t, j_t \in \{\mathbf{n}\} \setminus \mathbb{I}^{[t-1]}$ of $R^{[t-1]}$ so that $j_t \in \mathbb{I}^{[t]}$. As a result, $\mathbb{I}^{[0]} \subset \mathbb{I}^{[1]} \subset \dots \subset \mathbb{I}^{[k]}$. ■

One peculiarity that we need to better address is the last couple steps of `BinConnSeq`, where we may append a SWAP gate to the running sequence of two-operators.

Lemma 12 *Let $F_k \cdots F_1$ be a two-operator sequence given by Subroutine `BinConnSeq` on an n -qubit Clifford unitary C . If F_k is a SWAP gate, then $k - 1 < n - 1$.*

Proof: Let $R^{[t]}$ be the state of the stabilizer matrix R after round t of the **while** loop, and let $\mathbb{I}^{[t]} = \{j \in \{\mathbf{n}\} \mid R_{i,j}^{[t]} \text{ is dw-one}\}$. If we append a SWAP gate, then $1 \in \mathbb{I}^{[k-1]}$. For this to occur, there must have been a round $t \leq k - 1$ when two Pauli operators g and h of weight $\mathbf{w}(g) = \mathbf{w}(h) = 2$ were impacted by the same two-operator F_t , and for which $i_t = 1$ and $j_t > 1$. Otherwise, $i_t, j_t > 1$, or the round impacted only one operator g . In both cases, the dw-one entries are away from column 1. Therefore $|\mathbb{I}^{[t]}| - |\mathbb{I}^{[t-1]}| \geq 2$, which implies $k - 1 < n - 1$ complete cycles. ■

We now prove that `BinConnSeq` returns a binary connected sequence.

Lemma 13 *Suppose Subroutine `BinConnSeq` finishes without error on an n -qubit Clifford unitary C . Then the two-operator sequence $F_k \cdots F_1$ given by `BinConnSeq` is binary connected with $k \leq n - 1$.*

Proof: Assume C is not realized by SWAP and single qubit Clifford gates only so the **while** loop is not bypassed. Let $R^{[t]}$ be the state of the stabilizer matrix R at the end of round t , and let $\mathbb{I}^{[t]} = \{j \in \{\mathbf{n}\} \mid R_{i,j}^{[t]} \text{ is dw-one}\}$. We show that for each F_t chosen during a round t , that qubit j_t is inactive relative to $F_k \cdots F_{t+1}$ for all $t \in \{\mathbf{k} - \mathbf{1}\}$. This is the same as proving $\mathbb{I}^{[t]}$ is the set of inactive qubits relative to $F_k \cdots F_{t+1}$.

Let us assume the last member F_k is not a SWAP gate first. We begin with the first participant F_1 and the qubits i_1 and $j_1 > i_1$ affected by F_1 . By design, we choose F_1 in such a way that $j_1 \notin \mathbb{I}^{[0]}$ but $j_1 \in \mathbb{I}^{[1]}$. By Lemma 11, column j_1 also belongs to every $\mathbb{I}^{[t]}$ for rounds $t \geq 2$. This means qubit j_1 is inactive relative to F_2 , and every subsequent F_t

because $i_t, j_t \in \{\mathbf{n}\} \setminus \mathbb{I}^{[t-1]}$ for all $t \in \{2, \dots, k\}$. This argument for j_1 extends to j_t for every two-operator F_t in the $F_k \cdots F_1$ sequence. That is because each F_t is picked in a way so that between $i_t \geq 1$ and $j_t > 1$, we add j_t to $\mathbb{I}^{[t]}$. Moreover, $|\mathbb{I}^{[t]}| - |\mathbb{I}^{[t-1]}| \geq 1$ for every round t , which means $k \leq n - 1$ complete cycles by the **while** loop. Lastly, we get $i_{n-1} = 1$ if $k = n - 1$, since $|\{\mathbf{n}\} \setminus \mathbb{I}^{[n-2]}| = 2$, of which one of them is 1.

Now suppose F_k is a SWAP gate. Already, $k \leq n - 1$ by Lemma 12. We know $i_k = 1$ and $j_k > 1$ for the last qubit pair of F_k . Recall also that column $j_t > 1$ for all two-operators F_t in $F_{k-1} \cdots F_1$. We need to show $j_t \neq j_k$ for all $t \in \{\mathbf{k} - \mathbf{1}\}$ to show each qubit j_t is inactive relative to F_k . This is easy to see by inspection of $R^{[k-1]}$: before adding $\text{SWAP}^{(1, j_k)}$, column j_k is populated only by I characters and has weight zero, which means $j_k \notin \mathbb{I}^{[k-1]}$. But for all $t \in \{\mathbf{k} - \mathbf{1}\}$, we have $j_t \in \mathbb{I}^{[k-1]}$. Thus $F_k \cdots F_1$ is binary connected for both endings. ■

Subroutine 6:

Input: an n -qubit Clifford unitary C'

Output: n single qubit Clifford gates G_i

Select Clifford gates G_i so that $(G_1 \otimes \cdots \otimes G_n)C' \sim_1 I$.

procedure Finalize(C')

Setup: $g_1 \otimes \cdots \otimes g_n \leftarrow C'X^{(1)}C'^\dagger$

$h_1 \otimes \cdots \otimes h_n \leftarrow C'Z^{(1)}C'^\dagger$

$R \leftarrow \text{RREF}(\text{StabilizerMatrix}(\{C'Z^{(2)}C'^\dagger, \dots, C'Z^{(n)}C'^\dagger\}))$

$G_1 \leftarrow$ single qubit Clifford gate such that $G_1g_1G_1^\dagger = X$ and $G_1h_1G_1^\dagger = Z$

for $i = 2$ **to** n **do**

$r_1 \otimes \cdots \otimes r_{n-1} \leftarrow \text{col}(R, i)$

$G_i \leftarrow$ single qubit Clifford gate such that $G_i r_{i-1} G_i^\dagger = Z$

return $C_2 = G_1 \otimes \cdots \otimes G_n$

With the circuit from `BinConnSeq`, we cap off the algorithm with a layer of single qubit Clifford gates. This last stage is necessary to ensure the final sequence implements a Clifford unitary that is equivalent to the given Clifford operation C . Altogether, Algorithm `BinaryTree` proceeds as follows.

Algorithm 7:

Input: an n -qubit Clifford unitary C

Output: a binary in-tree Clifford unitary C_{bt} , or an **error**

Try to build a Clifford circuit implementing a binary in-tree unitary $C_{\text{bt}} \sim_1 C$.

procedure `BinaryTree`(C)

$C_1 = F_k \cdots F_1 \leftarrow \text{BinConnSeq}(C)$

$C_2 = G_1 \otimes \cdots \otimes G_n \leftarrow \text{Finalize}(C_1 C^\dagger)$

return $C_{\text{bt}} = C_2 C_1$

Before we present the main theorem, we need to examine some details concerning its runtime and input representation.

8.3.2 Runtime Considerations

Algorithm `BinaryTree` is efficient, provided we supply alternative representations of the input Clifford unitary C in place of a $2^n \times 2^n$ matrix. One option are the $2n$ mappings on the Pauli operators $X^{(i)}$ and $Z^{(i)}$, but a more standard one is perhaps any Clifford circuit to C . Aaronson and Gottesman [3] have already demonstrated how to synthesize a Clifford circuit with $O(n^2/\log n)$ total CNOT, H , P gates implementing any such unitary C from the $2n$ mappings. Column operations for the Clifford group generators can be accomplished in $O(n)$ time to obtain the initial stabilizer matrix in $O(n^3/\log n)$. Subroutine `RREF`, like Gaussian elimination, requires $O(n^3)$ operations, implying that the initialization of `BinConnSeq` takes $O(n^3)$ steps.

The most time-consuming part of `BinConnSeq` is `Search`. There are a max of $\binom{n}{2} = O(n^2)$ candidates to enumerate. For each candidate, we need to test its commutation and dependence relation with $n - 1$ other Pauli operators, which takes $O(n^2) + O(n^3)$ and so a worst-case scenario of $O(n^5)$ for `Search`. If `Search` returns successfully, luckily there are only $2 \cdot 3^2 = 18$ rules to remember for the two-operator $F(i, j)$ selection in `Select`. Since `while` may loop $O(n)$ times, the worst-case time complexity of `BinConnSeq` is already estimated at $O(n^6)$. Updating the stabilizer matrix at the end of `while` takes $O(n) + O(n^3)$ steps, considerably less than `Search`.

We can recommend one easy enhancement to improve the runtime of `Search`. In practice, if the reduced row echelon form of stabilizer matrix R is maintained at the end of each iteration for use in the next execution of `Search`, then the commutation and dependence check may be done simultaneously in time $O(n^2)$. We only have to perform pairwise row operations between each generator and the candidate in the last row. If the candidate does not commute with every member nor is dependent on the elements of S , then we move on. Overall, `Search` can complete in $O(n^4)$ steps.

The initialization presented in the pseudocode of `Finalize` is more for clarity so we are able to split and describe the algorithm in terms of smaller pieces. In actuality, we can simply maintain and update all $n + 1$ Pauli operators $Z^{(1)}, \dots, Z^{(n)}$, and $X^{(1)}$ from the start, as opposed to just $Z^{(2)}, \dots, Z^{(n)}$. As a result, the worst-case runtime of Algorithm `BinaryTree` is $O(n^5)$ steps.

8.3.3 Main Result

Assuming that we provide the appropriate representation for the input Clifford unitary C , and that we perform Algorithm `BinaryTree` with all the refinements mentioned earlier, then our theorem states the following.

Theorem 9 (Wong, unpublished) *Upon completion without error on an n -qubit Clifford operation C , Algorithm `BinaryTree` returns a binary in-tree unitary $C_{\text{bt}} \sim_1 C$. Moreover, Algorithm `BinaryTree` finishes in time $O(n^5)$.*

Proof: The time complexity argument is already stated above in Subsection 8.3.2. Let $C_{\text{bt}} = C_2 C_1$, where $C_1 = F_k \cdots F_1$ is a binary connected two-operator sequence from `BinConnSeq`, and $C_2 = G_1 \otimes \cdots \otimes G_n$ consists of single qubit Clifford gates. Lemma 2 confirms $C_{\text{bt}} \sim_1 C$ because

$$\left\langle C_{\text{bt}}^\dagger Z^{(2)} C_{\text{bt}}, \dots, C_{\text{bt}}^\dagger Z^{(n)} C_{\text{bt}} \right\rangle = \left\langle C^\dagger Z^{(2)} C, \dots, C^\dagger Z^{(n)} C \right\rangle \quad (8.12)$$

and the G_1 gate rotates the first qubit to the correct orientation. We need to verify C_{bt} is a binary in-tree unitary. If $k = n - 1$, then C_1 is such by Lemma 13.

If $k < n - 1$, we need $n - 1 - k$ two-operator assignments to complete the sequence. After `BinConnSeq`, we add only single qubit Clifford gates G_1, \dots, G_n , some of which can be absorbed into the earlier two-operators in $C_1 = F_k \cdots F_1$. Without loss of generality, assume that $i_t = n - t$ and $j_t = n - t + 1$ for every two-operator F_t in $C_1 = F_k \cdots F_1$. We need assignments on gates G_1, \dots, G_{n-1-k} covering qubits $1, \dots, n - k$. Here is our solution: for each $k < u \leq n - 1$, we define F_u to consist of G_{n-u} on qubit $i_u = n - u$, an identity gate I on qubit $j_u = n - u + 1$, as well as identity gates on the remaining $n - 2$ qubits. It is possible for $G_{n-u} = I$. This final sequence $F_{n-1} \cdots F_1$ is obviously binary connected and therefore implements a binary in-tree unitary. ■

8.3.4 Final Remark

Because we focus on n -qubit product states $\rho = \varphi_1 \otimes \cdots \otimes \varphi_n$, there are opportunities to trim the input problem size. That is, if $C = C_1 \otimes C_2$, where C_1 acts on the first k

qubits, then we only synthesize a Clifford circuit for C_1 . The separation of C means Z -measurements on qubits $k + 1$ to n have no effect on the first k qubits. This tensor product will manifest in the beginning of Subroutine `BinConnSeq`, when

$$R = \begin{bmatrix} R_1 & I' \\ I'' & R_2 \end{bmatrix} \quad (8.13)$$

where R_1 is a $(k - 1) \times k$ stabilizer matrix and R_2 is an $(n - k) \times (n - k)$ stabilizer matrix. The blocks I' and I'' contain entirely of I symbols.

8.4 Partial Binary In-tree Form Circuit Synthesis

We may insert Algorithm `BinaryTree` as a subroutine within a larger stabilizer circuit synthesis algorithm. The idea is quite simple. We prioritize Clifford circuits with binary in-tree form and will attempt at such a configuration first. If Algorithm `BinaryTree` fails part way, then we keep the running binary connected sequence and invoke a different algorithm to assemble the remaining circuit. The resulting decomposition, given an n -qubit Clifford unitary C , is $C_2 F_k \cdots F_1$, where $F_k \cdots F_1$ comes from `BinaryTree`, and C_2 is some other n -qubit Clifford unitary that acts trivially on k of the n qubits. The implementation of C_2 is handled by a second algorithm e.g. Aaronson and Gottesman [3]. Figure 8.2 displays an example stabilizer circuit where the subcircuit on the last four qubits exhibits a binary in-tree structure.

8.5 Examples

We demonstrate Algorithm `BinaryTree` on two Clifford unitaries described by Duclos-Cianci and Svore [26], then show that following the multistep tree execution strategy of

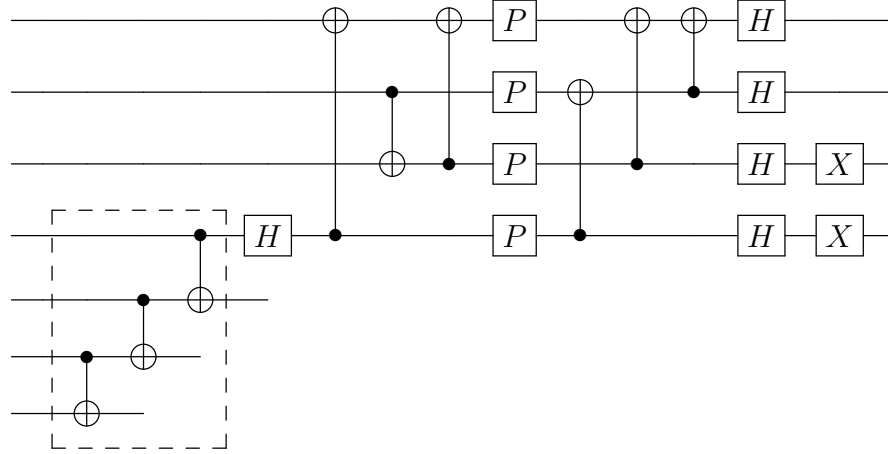


Figure 8.2: We may install Algorithm `BinaryTree` in a larger synthesis algorithm to construct a subcircuit with binary in-tree form that appears in the beginning of the entire stabilizer circuit.

Section 7.5 improves the expected cost to produce various non-stabilizer resource qubits. The stabilizer circuits of the two unitaries have three use cases, the first of which is provided in Figure 8.3a. The goal is to generate a qubit in the non-stabilizer state

$$|\psi_0\rangle = \cos(\phi_0)|0\rangle + \sin(\phi_0)|1\rangle, \quad \cos(2\phi_0) = \frac{6 + 5\sqrt{2}}{6 + 6\sqrt{2}}, \quad 2\phi_0 \approx 0.4456 \quad (8.14)$$

from four magic states $|H\rangle$. Let $C^{[1]}$ be the Clifford unitary implemented by the Clifford circuit of Figure 8.3a. This unitary abides by the transformation

$$\begin{aligned} X \otimes X \otimes Z \otimes I & \quad I \otimes Z \otimes I \otimes I \\ Z \otimes I \otimes X \otimes X & \mapsto I \otimes I \otimes Z \otimes I \\ I \otimes I \otimes Z \otimes Z & \quad I \otimes Z \otimes I \otimes Z. \end{aligned} \quad (8.15)$$

We immediately see that there is a weight two element among the three Pauli operators on the left, which will be the starting point for our synthesis algorithm. If we follow Algorithm `BinaryTree` to completion, then we obtain the circuit in Figure 8.3b, including

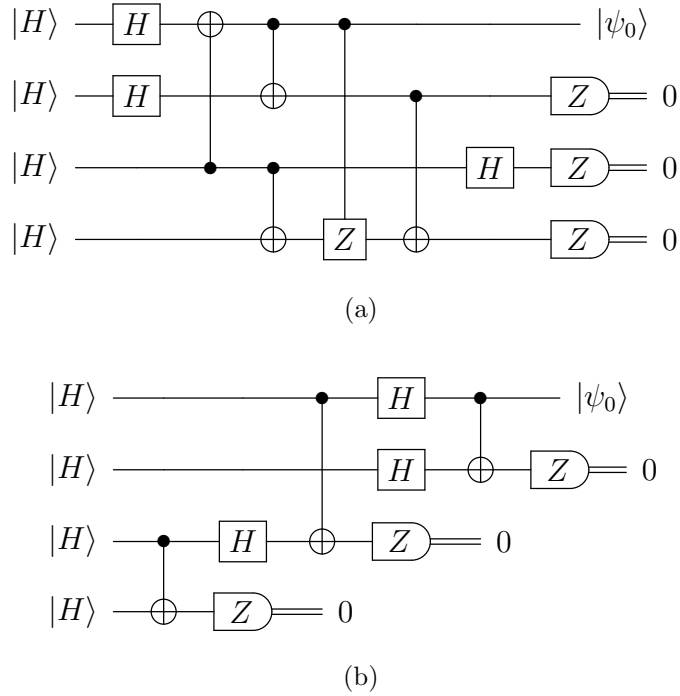


Figure 8.3: (a) Stabilizer circuit to produce a resource qubit $|\psi_0\rangle$ as seen in [26]. (b) Stabilizer circuit with binary in-tree form also producing the same qubit $|\psi_0\rangle$.

measurements. This implements a binary in-tree Clifford unitary $C_{\text{bt}}^{[1]} \sim_1 C^{[1]}$ such that

$$\begin{aligned}
 X \otimes X \otimes Z \otimes I & \quad I \otimes Z \otimes I \otimes I \\
 Z \otimes I \otimes X \otimes X & \mapsto I \otimes I \otimes Z \otimes I \\
 I \otimes I \otimes Z \otimes Z & \quad I \otimes I \otimes I \otimes Z.
 \end{aligned} \tag{8.16}$$

Between $C^{[1]}$ and $C_{\text{bt}}^{[1]}$, we change the mapping of the third Pauli operator $I \otimes I \otimes Z \otimes Z$ while maintaining the same mapping for the other two, yet it leads to completely different gate sequences in the final quantum circuit.

In the second usage case, we see that Duclos-Cianci and Svore employ the same stabilizer circuit from Figure 8.3a to create another non-stabilizer qubit

$$|\psi_1\rangle = \cos(\phi_1)|0\rangle + \sin(\phi_1)|1\rangle, \quad \cos(2\phi_1) = \frac{3 + \sqrt{2}}{1 + 3\sqrt{2}}, \quad 2\phi_1 \approx 0.57. \tag{8.17}$$

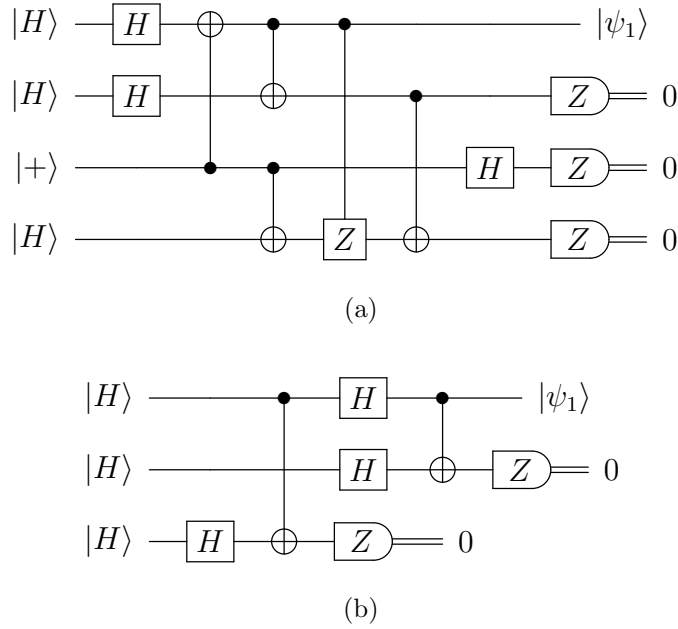


Figure 8.4: (a) Stabilizer circuit to create $|\psi_1\rangle$ from [26]. (b) Stabilizer circuit with binary in-tree form that also outputs $|\psi_1\rangle$. The qubit $|+\rangle$ is not necessary to generate $|\psi_1\rangle$, leading to a simpler circuit on three $|H\rangle$ qubits.

The difference lies in the initial state, and is an example of when the synthesis algorithm helps with circuit simplification. By Lemma 3, we may eliminate $|+\rangle$ entirely from the Figure 8.4a circuit to obtain the three-qubit circuit in Figure 8.4b also producing $|\psi_1\rangle$. Looking at the binary in-tree form circuit in Figure 8.3b, we see that qubit three becomes another $|H\rangle$ after measuring 0 on qubit four. But since this observation occurs with probability $1/2$, the probability of the 0^3 outcome in Figure 8.4a is half the probability of the 0^2 outcome in Figure 8.4b, which doubles the expected cost of $|\psi_1\rangle$.

For the last case, the stabilizer circuit in Figure 8.5a applies a four-qubit Clifford unitary $C^{[2]}$ on four magic $|H\rangle$ states, then measures the last three qubits and postselects on 0^3 to return

$$|\psi_2\rangle = \cos(\phi_2)|0\rangle + \sin(\phi_2)|1\rangle, \quad \cos(2\phi_2) = \frac{6\sqrt{2}}{11}, \quad 2\phi_2 \approx 0.69. \quad (8.18)$$

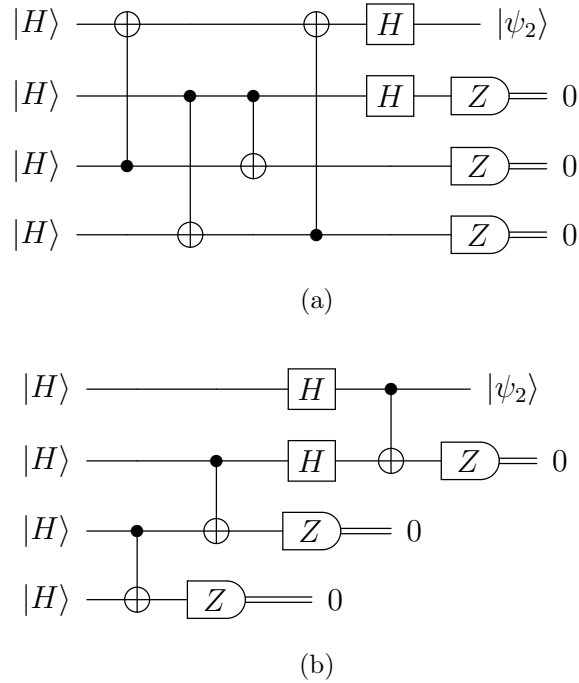


Figure 8.5: (a) Stabilizer circuit to produce $|\psi_2\rangle$ from [26]. (b) Stabilizer circuit with binary in-tree form to produce $|\psi_2\rangle$ from four $|H\rangle$ qubits.

Note that $\frac{\pi}{8} \approx 0.392$ and $\frac{\pi}{4} \approx 0.785$, so the $2\phi_i$ angles of the $|\psi_i\rangle$ qubits are fairly distributed between $\frac{\pi}{8}$ and $\frac{\pi}{4}$. In this instance, $C^{[2]}$ performs the following action:

$$\begin{aligned}
 X \otimes X \otimes X \otimes X & \quad I \otimes Z \otimes I \otimes I \\
 I \otimes Z \otimes Z \otimes I & \mapsto I \otimes I \otimes Z \otimes I \\
 I \otimes I \otimes Z \otimes Z & \quad I \otimes I \otimes Z \otimes Z.
 \end{aligned} \tag{8.19}$$

while the stabilizer circuit in Figure 8.5b implements a binary in-tree Clifford unitary $C_{\text{bt}}^{[2]} \sim_1 C^{[2]}$ that sends

$$\begin{aligned}
 X \otimes X \otimes X \otimes X & \quad I \otimes Z \otimes I \otimes I \\
 I \otimes Z \otimes Z \otimes I & \mapsto I \otimes I \otimes Z \otimes I \\
 I \otimes I \otimes Z \otimes Z & \quad I \otimes I \otimes I \otimes Z.
 \end{aligned} \tag{8.20}$$

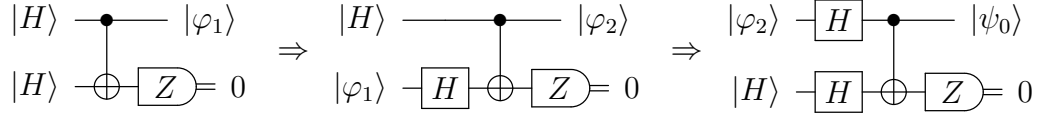


Figure 8.6: Procedure to generate $|\psi_0\rangle$ with three delegated two-op circuits and four $|H\rangle$ states. If we measure 1 at any of the three steps, then we restart from the first circuit on the left with two new $|H\rangle$ copies. Adding recovery for the last two-qubit stabilizer circuit additionally improves the average $|H\rangle$ cost.

According to Theorem 8, we should detect noticeable gains in the expected cost when we execute the stabilizer circuits of Figures 8.3b, 8.4b and 8.5b in accordance with the multistep tree protocol of Section 7.5. We briefly walkthrough the expected cost calculations for $|\psi_0\rangle$ and $|\psi_1\rangle$, as the computation is very similar for $|\psi_2\rangle$. For the postselected four-to-one stabilizer circuit $(C^{[1]}, 0^3)$, the probability of the successful 0^3 outcome is $Q_{0^3}^{\text{sc}}(C^{[1]}, |H\rangle^{\otimes 4}) = 3(2 + \sqrt{2})/32 \approx 0.32$, which implies an average

$$E_{\text{sc}}(|\psi_0\rangle) = \frac{4 \cdot 32}{3(2 + \sqrt{2})} \approx 12.5 \quad (8.21)$$

$|H\rangle$ qubits per $|\psi_0\rangle$ under the basic execution policy. Now for the binary in-tree unitary

$$C_{\text{bt}}^{[1]} = \text{CNOT}^{(1,2)} H^{(2)} H^{(1)} \text{CNOT}^{(1,3)} H^{(3)} \text{CNOT}^{(3,4)}, \quad (8.22)$$

let $(F_1, 0)$, $(F_2, 0)$, $(F_3, 0)$ be postselected two-op circuits. We assign

$$F_1 = \text{CNOT}^{(3,4)} \quad F_2 = \text{CNOT}^{(1,3)} H^{(3)} \quad F_3 = \text{CNOT}^{(1,2)} H^{(2)} H^{(1)}. \quad (8.23)$$

Then we may obtain $|\psi_0\rangle$ with the corresponding delegated two-op circuits by carrying out the procedure in Figure 8.6 (with $C_{F_1} = \text{CNOT}^{(1,2)}$, $C_{F_2} = \text{CNOT}^{(1,2)} H^{(2)}$, etc.).

By Proposition 3, we can simply follow the process in Figure 8.6 to compute the expected cost of $|\psi_0\rangle$ under the multistep tree execution method. To stress the fact

that we are working with delegated two-op circuits, we use “dc” instead of “sc” e.g. E_{dc} for the remainder of this section. The probability of succeeding on the first step is $Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}, |H, H\rangle) = 3/4$, so the expected cost to generate an output $|\varphi_1\rangle$ of $(\text{CNOT}^{(1,2)}, 0)$ on $|H, H\rangle$ is

$$E_{\text{dc}}(|\varphi_1\rangle) = \frac{1 + 1}{Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}, |H, H\rangle)} = \frac{8}{3}. \quad (8.24)$$

Next we have $(\text{CNOT}^{(1,2)}H^{(2)}, 0)$, the delegated two-op circuit of $(F_2, 0)$. The two-qubit input for this circuit is $|H, \varphi_1\rangle$, yielding

$$Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}H^{(2)}, |H, \varphi_1\rangle) = \frac{1 + 3\sqrt{2}}{6\sqrt{2}} \approx 0.6179. \quad (8.25)$$

If $|\varphi_2\rangle$ represents the output of $(\text{CNOT}^{(1,2)}H^{(2)}, 0)$ on $|H, \varphi_1\rangle$, then its expected cost is

$$E_{\text{dc}}(|\varphi_2\rangle) = \frac{1 + 8/3}{Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}H^{(2)}, |H, \varphi_1\rangle)} = \frac{22\sqrt{2}}{1 + 3\sqrt{2}} \approx 5.935. \quad (8.26)$$

For this particular setup, $|\psi_1\rangle = |\varphi_2\rangle$. Lastly, we arrive at $(\text{CNOT}^{(1,2)}H^{(2)}H^{(1)}, 0)$, the delegated two-op circuit of $(F_3, 0)$. The input is similarly $|\varphi_2, H\rangle$. The probability of obtaining $|\psi_0\rangle$ in the final step is therefore

$$Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}H^{(2)}H^{(1)}, |\varphi_2, H\rangle) = \frac{3 + 3\sqrt{2}}{2 + 6\sqrt{2}} \approx 0.6907, \quad (8.27)$$

leading to the new and reduced $|\psi_0\rangle$ expected cost

$$E_{\text{dc}}(|\psi_0\rangle) = \frac{1 + \frac{22\sqrt{2}}{1+3\sqrt{2}}}{Q_0^{\text{dc}}(\text{CNOT}^{(1,2)}H^{(2)}H^{(1)}, |\varphi_2, H\rangle)} = \frac{2 + 50\sqrt{2}}{3 + 3\sqrt{2}} \approx 10.039. \quad (8.28)$$

The other value $E_{dc}(|\psi_2\rangle)$ following the multistep approach is determined in the same fashion. The expected costs for all three qubits are summarized in the table below:

Output State	Previous Cost E_{sc}	New Cost E_{dc}
$ \psi_0\rangle$	12.5 $ H\rangle$	10.04 $ H\rangle$
$ \psi_1\rangle$	12.95 $ H\rangle$	5.94 $ H\rangle$
$ \psi_2\rangle$	11.64 $ H\rangle$	9.82 $ H\rangle$

The biggest improvement is in the creation of $|\psi_1\rangle$, where the new expected cost is less than half of the previous technique by Duclos-Cianci and Svore.

Small additional reductions are possible by incorporating the recovery techniques of Chapter 5. This mainly applies to $|\psi_0\rangle$ and $|\psi_2\rangle$, as we expend enough resources toward the end these processes that recovery might play a supporting role. We perform simulations with a single recovery step (nested recovery protocol of depth $k = 3$) when we fail to create $|\psi_0\rangle$ and $|\psi_2\rangle$ on the last delegated two-op circuit; we also see the same numbers using Theorem 5. The averages lower slightly in both cases:

Output State	No Recovery New Cost	With Recovery New Cost
$ \psi_0\rangle$	10.04 $ H\rangle$	9.45 $ H\rangle$
$ \psi_2\rangle$	9.82 $ H\rangle$	9.64 $ H\rangle$

Ultimately, the $|\psi_i\rangle$ qubits, like $|H\rangle$, are consumed to generate “ladder” resource states that we first referenced in Chapter 4. To create $|\psi_i\rangle$ -type ladder states on top of $|H\rangle$ -type ladder states, we simply start with $|\psi_i\rangle \otimes |H\rangle$ as input for the postselected stabilizer circuit $(\text{CNOT}^{(1,2)}, 0)$. For example, if $|\psi_0^1\rangle$ is the output of $(\text{CNOT}^{(1,2)}, 0)$ on $|\psi_0\rangle \otimes |H\rangle$, then $|\psi_0^2\rangle$ is the output of $(\text{CNOT}^{(1,2)}, 0)$ on $|\psi_0^1\rangle \otimes |H\rangle$. Given $|\psi_0^2\rangle$,

we have a path to create $|\psi_0^3\rangle$. Continuing in this manner, $|\psi_0^{i+1}\rangle$ is then the output of $(\text{CNOT}^{(1,2)}, 0)$ on $|\psi_0^i\rangle \otimes |H\rangle$.

Duclos-Cianci and Svore show that by leveraging such a warehouse of qubits, we can approximate $R_z(\theta)$ for any $\theta \in (0, \pi/2)$ to accuracy $10^{-12} < \epsilon < 10^{-4}$ with better scaling than Solovay-Kitaev's algorithm [26]. We run the same simulation pattern in [26] to observe how the *offline costs* E_{off} are affected by the new numbers in the table above. The offline cost is defined as the number of $|H\rangle$ qubits used in the approximation of $R_z(\theta)$ to error ϵ . To give a sense of how E_{off} is determined, suppose for $\theta = \pi/100$ and $\epsilon = 10^{-10}$, we used 4 $|H\rangle$ -type, 6 $|\psi_0\rangle$ -type, 7 $|\psi_1\rangle$ -type, and 3 $|\psi_2\rangle$ -type ladder states. To generate the appropriate ladder qubits (e.g. $|\psi_1^4\rangle$), suppose we used an additional 280 $|H\rangle$ states. If $|\psi_i\rangle$ are obtained following the basic execution protocol, this leads to

$$E_{\text{off}} = 280 + 1 \cdot 4 + 12.5 \cdot 6 + 7 \cdot 12.95 + 3 \cdot 11.64 = 484.57 \quad (8.29)$$

$|H\rangle$ qubits in this instance. On the other hand, $|\psi_i\rangle$ qubits that are created according to the multistep tree policy with recovery yield

$$E_{\text{off}} = 280 + 1 \cdot 4 + 9.45 \cdot 6 + 7 \cdot 5.94 + 3 \cdot 9.64 = 411.2 \quad (8.30)$$

as the offline cost. Data points for the basic strategy are presented in Figure 8.8a, and data points for the multistep approach with recovery are contained in Figure 8.8b. While the scaling remains practically unchanged, the actual costs to perform the approximations do shift down for the better, as seen in Figure 8.7. The lower line with slope ~ 1.8158 and intercept ~ 0.3361 represents the linear fit of the data in Figure 8.8b. The top line with slope ~ 1.7186 and intercept ~ 0.78745 is the linear fit of the numbers in Figure 8.8a. The average difference in cost turns out to be about 50.4 $|H\rangle$ qubits.

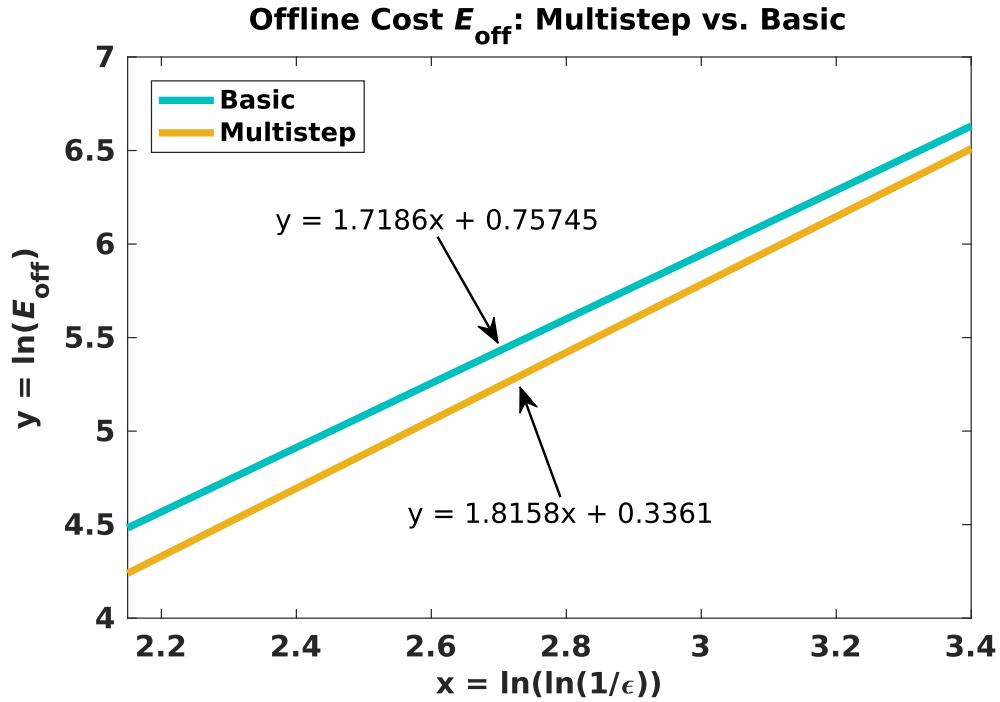


Figure 8.7: Comparison of the offline cost E_{off} fit lines from Figure 8.8. The scaling (slope) between the two approaches are basically the same, but the average shift represents about 50.4 $|H\rangle$ qubits.

8.6 Summary

We studied n -qubit stabilizer circuits with $n - 1$ Pauli measurements, and gave an efficient classical algorithm, `BinaryTree`, to synthesize stabilizer circuits with binary in-tree form. Like the nested recovery protocol, binary in-tree decomposition is bounded to specific stabilizer operations. Nevertheless, when such a configuration exists, we showed that it leads to better expected resource costs, as long as we follow a multistep process to execute the proposed quantum operation. It is not difficult to see that the improvements are a natural consequence of the circuit's increased modularity, potentially allowing parallel subunits of activity. Finally, we demonstrated the utility of the binary in-tree form on concrete examples.

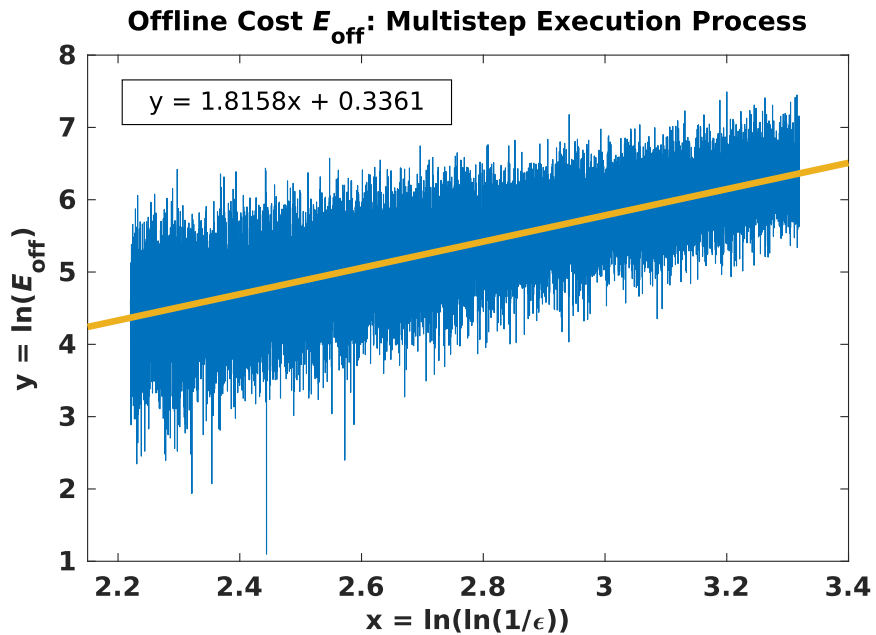
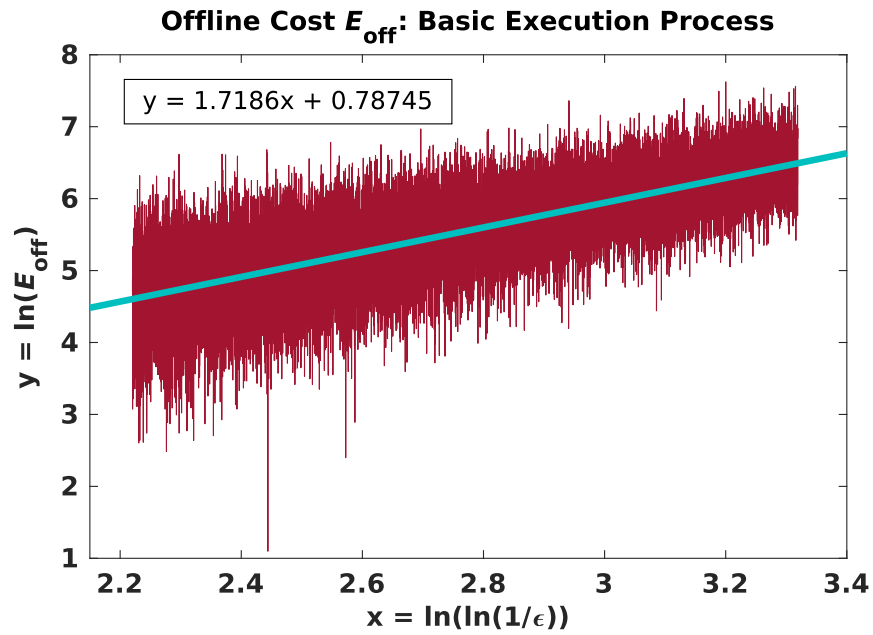


Figure 8.8: Simulation data of the offline cost E_{off} to approximate $R_z(\theta)$ to accuracy $10^{-12} < \epsilon < 10^{-4}$ using $|H\rangle$ -type and $|\psi_i\rangle$ -type ladder qubits. The base qubits $|\psi_i\rangle$ are generated either from (a) the basic execution process or (b) the multistep execution process with recovery.

Chapter 9

Conclusion

In this thesis, we explored two ideas concerning stabilizer operations. First we considered the invertibility of two-qubit stabilizer circuits with one Z -measurement when given inputs $\varphi \otimes \psi$, where ψ is a pure non-stabilizer state. Then we gave an illustration in which we portrayed certain quantum circuits as possessing tree-like qualities. This viewpoint lead to our characterization of stabilizer circuits with binary in-tree form. Investigations into both approaches produced some notable outcomes, namely (1) a probabilistic protocol for recovering qubits, and (2) a multistep process to generate an output φ' of a postselected n -to-one stabilizer circuit $(C, 0^{n-1})$ on an input $\varphi_1 \otimes \cdots \otimes \varphi_n$. The C of interest in the second result is a binary in-tree unitary. Since the recovery and multistep options are based on two-to-one qubit processes, both techniques are readily adoptable in the present moment, when physical control of large quantum systems is difficult.

Over time, we expect hardware developments will lead to better tools to yield near perfect qubits outright. Although state distillation may become practically obsolete in such events, our techniques might be able to withstand this change. It is not hard to see that our methods and analyses are applicable even when we replace the underlying cost unit from non-stabilizer states to arbitrary quantum states or another resource. In this sense, our work may stay relevant even as technology progresses. We offer some final thoughts regarding future work around these two topics.

9.1 Recovery Circuits

To be of greater practical value, one direction of interest is to study the impact of imperfect $|\psi\rangle$ qubits on the recovery process. A numerical experiment with magic states $|\psi\rangle = |H\rangle$ in [26] indicates a decay for certain error rates, but whether this observation is retained for arbitrary non-stabilizer $|\psi\rangle$ states is unknown. A related question is how the optimal depth may be affected by the presence of errors, where we expect the optimal depth k_{opt} to decrease but by what amount.

The more compelling problem is whether something resembling recovery circuits can easily be extended for larger stabilizer circuits, since our two-qubit setting is appropriate for only a limited number of scenarios. This question has been answered to an extent for the Clifford+ T gate set in [12, 13, 58], where we assume $|\psi\rangle = |\frac{\pi}{4}\rangle$. The objective is to construct a multiqubit circuit of Clifford+ T gates and Z -measurements to approximate an arbitrary single qubit unitary U up to some error ϵ . If the measurements are unfavorable, then there is a backup operation that either returns the qubits to the initial state, or directly tries to approximate U with a secondary circuit. It is worth investigating whether recovery processes similar to the one in Chapter 4 exist on general $|\psi\rangle$ resources, and if so, whether the reduction factor can grow beyond two.

9.2 Stabilizer Circuits with In-tree Form

Binary in-tree form is rather prohibitive, and beneficial for only some important cases. In the long run, we want in-trees with greater breadth. Unfortunately, trying to obtain more arbitrary in-tree configurations is more difficult than it may seem. To understand the meaning behind such a claim, we look at a complexity problem that might bear some relevance to synthesizing stabilizer circuits with general in-tree form.

9.2.1 One Possible Generalization of In-Tree Form

For the moment, let us continue the focus on n -qubit stabilizer circuits, with $n - 1$ Pauli Z -measurements, and n -qubit product states $\varphi_1 \otimes \cdots \otimes \varphi_n$. At the very least, we may treat any n -qubit Clifford operation C as an n -ary in-tree unitary, where, like the four-qubit example in Figure 7.1, we draw one root qubit dependent on all n leaf qubits. Depending on its implementation, we may upgrade the classification to an m -ary in-tree unitary, where $m < n$. This means if $C = C_l \cdots C_1$, then each Clifford operator C_t acts nontrivially on at most m qubits. Following each application of C_t on $d \leq m$ qubits, we perform Z -measurements on $d - 1$ qubits that are inactive relative to $C_l \cdots C_{t+1}$. The objective then is to determine an equivalent m' -ary in-tree unitary $C' = C'_l \cdots C'_1 \sim_1 C$ such that m' is as small as possible. The reason being that if m' is small, then l' and the number of in-tree layers is likely to be bigger, so there is greater potential for resource conservation. The generalization for stabilizer circuits with $n - k$ measurements to produce a k -qubit output, on the other hand, is less obvious.

9.2.2 Potential Challenges to Circuit Synthesis

To assemble an n -qubit Clifford circuit $C = C_l \cdots C_1$ with m -ary in-tree form, one option is to use a stabilizer matrix with $n - 1$ rows and n columns like in Algorithm `BinaryTree`. However, the determination of each C_t is more complicated when $m > 2$. In the preceding chapters, when $m = 2$, we only had to find one Pauli operator of weight two because we apply one Z -measurement after every two-operator. If we now want a gate C_t that affects $d \leq m$ qubits, then we need to pick $d - 1$ independent Pauli operators to go with $d - 1$ measurements. Each element must have weight at most d , and the I , X , Y , Z symbols for these $d - 1$ operators must appear in d common places (to influence the same d qubits). For the remaining $n - d$ locations, we require I .

It is not clear how the selection of such $d - 1$ operators will take place. And even though we know we want m to be as small as possible, determining what values d and m to even start such a selection does not appear to be a simple task. Consider the following decision problem:

Minimum Distance Problem with Stabilizer Groups (MD-S): *Given a stabilizer group \mathcal{S} of n -qubit Pauli operators and a positive integer m , is there a nontrivial operator $g \in \mathcal{S}$ such that $\mathbf{w}(g) \leq m$?*

We can store the generators of \mathcal{S} in a stabilizer matrix like we have always done up to this point. However this is essentially the same as the Minimum Distance Problem (MD) from classical coding theory, which is known to be NP-complete [71]:

Minimum Distance Problem (MD): *Given a binary $k \times n$ matrix M and a positive integer w , is there a nonzero vector $x \in \{0, 1\}^n$ of weight $\leq w$ such that $Mx^\top = 0$?*

The reduction from MD to MD-S to prove NP-hardness is straightforward: for any $k \times n$ binary matrix M , we replace all 0 entries with the symbol I , and all 1 entries with X . Then we set $m = w$. Consequently, there is no known efficient solution to obtain the minimum weight Pauli operator of \mathcal{S} for a general integer m . Then again, we are not certain if stabilizer matrices are the best approach to synthesize Clifford circuits with arbitrary in-tree form. Thus we cannot say with guarantee that no such efficient synthesis algorithm exists.

9.2.3 Future Considerations

One ambitious undertaking is to develop classes of Clifford operations, some of which are susceptible to an in-tree implementation with more than one layer. This is similar to the path that classical and quantum coding theory has followed, where there are many different families of codes that have varying properties and designed to fulfill specific

purposes. The difficulty at the moment is there is not much theoretical basis to start such a classification. While this is certainly not ideal, it is an interesting research question whether this type of development is indeed possible for Clifford operations.

Appendix A

Bounded One-Dimensional Random Walk with Difference Equation

In this appendix, we study a particular category of random walks with applications to our nested recovery protocol analysis in Chapter 6. Specifically, we analyze a broad class of bounded one-dimensional random walk instances whose transitions are characterized by a rational difference equation (Definition 10). We determine two quantities regarding these bounded random walks: (1) the success probability P (Definition 28), and (2) the expected number of steps S (Definition 29), so that we may answer some interesting questions about our nested recovery protocol described in Chapter 5. Our solutions for P and S are summarized in Lemmas 14 and 15.

A.1 Bounded One-Dimensional Random Walk

A bounded one-dimensional random walk is a random walk over a finite integer set $\{0, \dots, k\}$, as seen in Figure A.1. For these types of Markov chains, there is a probability $L(i)$ of moving from i to $i - 1$ and a probability $1 - L(i)$ of moving from i to $i + 1$ for all interior points $i \in \{1, \dots, k - 1\}$. On the other hand, if i is a boundary 0 or k , then there is no lateral movement whatsoever, and we remain at that location. We may interpret an arrival on either endpoint to signal the end of the current random walk process. There

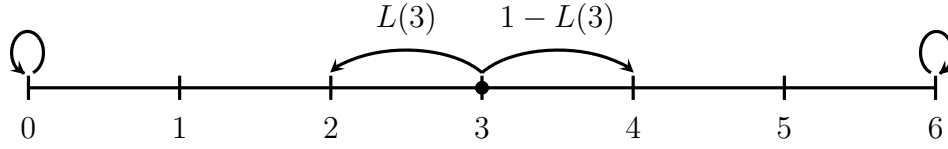


Figure A.1: A one-dimensional random walk over the integers $\{0, \dots, 6\}$. For some position $i \in \{1, \dots, 5\}$, there is a probability $L(i)$ of stepping to $i - 1$ and a probability $1 - L(i)$ of stepping to $i + 1$. We remain stationary if $i \in \{0, 6\}$.

are two quantities from [24] that we want to compute for these random walks, and the details of each are provided in the following definitions.

Definition 28 (Random Walk Success Probability) Consider a random walk over the integers $\{0, \dots, k\}$. Define $P(i)$ to be the probability that the walk, starting at i , successfully reaches 0 before it reaches k . Then the $P(i)$ probabilities are described by

$$P(i) = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{if } i = k \\ L(i)P(i - 1) + (1 - L(i))P(i + 1) & \text{otherwise} \end{cases} \quad (\text{A.1})$$

where $L(i)$ is the probability of a left step from i to $i - 1$.

Definition 29 (Random Walk Expected Number of Steps) Similar to Definition 28, define $S(i)$ to be the expected number of steps that the walk, starting at i , takes to reach 0 or k . Then the $S(i)$ expectations are described by

$$S(i) = \begin{cases} 0 & \text{if } i = 0 \text{ or } i = k \\ L(i)S(i - 1) + (1 - L(i))S(i + 1) + 1 & \text{otherwise} \end{cases} \quad (\text{A.2})$$

where $L(i)$ is the probability of a left step from i to $i - 1$.

According to [24], we have $P(i) = (k - i)/k$ and $S(i) = ki - i^2$ when $L(i) = 1/2$.

A.2 Random Walk with Difference Equation

For this section, we derive closed-form expressions for the success probability $P(i)$ and the expected number of steps $S(i)$ for bounded random walks with a rational difference equation (RDE) as the transition. Due to the nature of our RDEs, we arrive at two sets of formulas which we provide in Lemmas 14 and 15 below. To the best of our knowledge, there are no published works with the kind of results presented here for these types of random walks. As a reminder, the RDEs that we utilize are parameterized by two real numbers (λ, γ) called a probability specification (Definition 8), which also determines a set of intermediate functions (Definition 9).

Lemma 14 *If the left step probabilities of a random walk over $\{0, \dots, k\}$ are determined by an RDE on a restricted probability specification (λ, γ) , then the following are solutions to Equations A.1 and A.2 of the random walk:*

$$P(i) = \frac{A_1(1)A_1(k-i)\gamma\lambda^{i-1}}{A_1(k)B_1(i-1)} \quad (\text{A.3})$$

$$S(i) = \frac{A_1(k-i)(\gamma\lambda^{i-1} - 2\lambda^i)i + (k-i)A_1(i)B_2(k-1)}{A_1(1)A_1(k)B_1(i-1)} \quad (\text{A.4})$$

where $A_1(i)$, $B_1(i)$, and $B_2(i)$ are intermediate functions of (λ, γ) .

Lemma 15 *The solutions to Equations A.1 and A.2 are $P(i) = (k-i)/k$ and $S(i) = ki - i^2$ for a uniform random walk over $\{0, \dots, k\}$.*

Lemma 15 is already covered in [24], so we show how to reproduce it using material from our framework of derivations. Since the constant transition $L(i) = p > 1/2$ and $L(i) = p < 1/2$ is of little importance to us in Chapter 6, most of our activity is dedicated to proving Lemma 14 for RDEs decided by a restricted probability specification.

A.2.1 Intermediate Function Identities

As the algebra may become difficult to manage, the next set of identities will play an important part in the developments to come.

Lemma 16 *Let (λ, γ) be a restricted probability specification. Then we have the following identities on its intermediate functions $A_{1,2}(i)$ and $B_{1,2}(i)$:*

$$i. A_{1,2}(i+1) = A_{1,2}(i) - \lambda A_{1,2}(i-1)$$

$$ii. B_{1,2}(i+1) = B_{1,2}(i) - \lambda B_{1,2}(i-1)$$

$$iii. A_1(j)A_1(i) = A_2(j+i) - \lambda^i A_2(j-i)$$

$$iv. A_2(j)A_1(i) = A_1(j+i) - \lambda^i A_1(j-i)$$

$$v. B_1(i)A_1(i+1)A_1(1) + \lambda B_1(2i) = B_1(2i+2) - 2\lambda^{i+1}A_1(1) + \gamma\lambda^i A_1(1)$$

$$vi. B_1(j-i)A_1(i) = B_2(j) - \lambda^i B_2(j-2i)$$

$$vii. \lambda^i A_2(j-2i)A_1(1) + A_2(j-i+1)A_1(i) = A_2(j-i)A_1(i+1)$$

$$viii. \lambda^i B_2(j-2i-1)A_1(1) + B_2(j-i)A_1(i) = B_2(j-i-1)A_1(i+1)$$

Proof: Note that $\lambda = \alpha\beta$ and $A_2(1) = \alpha + \beta = 1$ for the boundaries (α, β) of a probability specification (λ, γ) . The first item is obvious from

$$A_1(i) - \lambda A_1(i-1) = \alpha^i - \beta^i - \alpha\beta(\alpha^{i-1} - \beta^{i-1}) \quad (\text{A.5})$$

$$= (1-\beta)\alpha^i - (1-\alpha)\beta^i \quad (\text{A.6})$$

$$A_2(i) - \lambda A_2(i-1) = \alpha^i + \beta^i - \alpha\beta(\alpha^{i-1} + \beta^{i-1}) \quad (\text{A.7})$$

$$= (1-\beta)\alpha^i + (1-\alpha)\beta^i \quad (\text{A.8})$$

and the second one follows immediately. The next two are just as easy to show:

$$A_1(j)A_1(i) = \alpha^{j+i} + \beta^{j+i} - \alpha^j\beta^i - \alpha^i\beta^j \quad (\text{A.9})$$

$$= \alpha^{j+i} + \beta^{j+i} - \alpha^i\beta^i (\alpha^{j-i} + \beta^{j-i}) \quad (\text{A.10})$$

$$A_2(j)A_1(i) = \alpha^{j+i} - \beta^{j+i} - \alpha^j\beta^i + \alpha^i\beta^j \quad (\text{A.11})$$

$$= \alpha^{j+i} - \beta^{j+i} - \alpha^i\beta^i (\alpha^{j-i} - \beta^{j-i}). \quad (\text{A.12})$$

The fifth identity looks a little more involved, but we just need to prove

$$B_1(i)A_1(i+1) = A_1(i+1)A_1(i+1) - \gamma A_1(i+1)A_1(i) \quad (\text{A.13})$$

$$= A_2(2i+2) - 2\lambda^{i+1} - \gamma A_2(2i+1) + \gamma\lambda^i \quad (\text{A.14})$$

$$= B_2(2i+1) - 2\lambda^{i+1} + \gamma\lambda^i \quad (\text{A.15})$$

$$B_2(2i+1)A_1(1) = A_2(2i+2)A_1(1) - \gamma A_2(2i+1)A_1(1) \quad (\text{A.16})$$

$$= A_1(2i+3) - \lambda A_1(2i+1) - \gamma A_1(2i+2) + \gamma\lambda A_1(2i) \quad (\text{A.17})$$

$$= B_1(2i+2) - \lambda B_1(2i) \quad (\text{A.18})$$

and the result becomes clear. The following covers vi:

$$B_1(j-i)A_1(i) = A_1(j-i+1)A_1(i) - \gamma A_1(j-i)A_1(i) \quad (\text{A.19})$$

$$= A_2(j+1) - \lambda^i A_2(j-2i+1) - \gamma A_2(j) + \gamma\lambda^i A_2(j-2i) \quad (\text{A.20})$$

$$= B_2(j) - \lambda^i B_2(j-2i) \quad (\text{A.21})$$

while vii is based on iv. That is, observe that the first term expands into

$$\lambda^i A_2(j-2i)A_1(1) = \alpha^i\beta^i (A_1(j-2i+1) - \alpha\beta A_1(j-2i-1)) \quad (\text{A.22})$$

and that the second term similarly converts into

$$A_2(j - i + 1)A_1(i) = A_1(j + 1) - \alpha^i \beta^i A_1(j - 2i - 1). \quad (\text{A.23})$$

This will allow us to smoothly arrive at

$$\lambda^i A_2(j - 2i)A_1(1) + A_2(j - i + 1)A_1(i) = A_2(j - i)A_1(i + 1). \quad (\text{A.24})$$

The last one is a consequence of vii. ■

A.2.2 Fundamental Matrix

We now start the general framework for computing $P(i)$ and $S(i)$ as seen in Lemmas 14 and 15, with Kemény and Snell's work [45] as an initial guide. At the heart of proving these statements is a tool known as the *fundamental matrix*. To define the fundamental matrix, we need to look at the transition matrix of the random walk first. A one-dimensional random walk on integers $\{0, \dots, k\}$ is also called an absorbing Markov chain, where the endpoints 0 and k serve as absorbing states. It has $k - 1$ transient (non-absorbing) states, and we may write down the transition matrix in canonical form as

$$\left[\begin{array}{c|c} \overbrace{\quad}^2 & \overbrace{\quad}^{k-1} \\ \hline I & O \\ \hline W & U \end{array} \right] \begin{array}{l} \} 2 \\ \} k-1 \end{array} \quad (\text{A.25})$$

where O contains all zeroes and I is the 2×2 identity. Each row sums to 1, and the first two rows represent transitions from the left and right boundaries 0 and k . The

block matrices W and U contain transition probabilities from transient to absorbing and transient to transient states, respectively. If $L(i)$ is the probability from i to $i - 1$ and $R(i) = 1 - L(i)$, then W is a matrix of mostly zeroes with the exception of two spots: $W_{1,1} = L(1)$ and $W_{k-1,2} = R(k - 1)$. On the other hand, U is not as sparsely populated as W . To be precise, the rows and columns are arranged such that

$$U_{i,j} = \begin{cases} L(i) & \text{if } j = i - 1 \\ R(i) & \text{if } j = i + 1 \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.26})$$

In other words, U has nonzero entries only at places immediately adjacent to the main diagonal. As an example,

$$\left[\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline L(1) & 0 & 0 & R(1) & 0 \\ 0 & 0 & L(2) & 0 & R(2) \\ 0 & R(3) & 0 & L(3) & 0 \end{array} \right] \quad (\text{A.27})$$

is the canonical transition matrix for a random walk with $k = 4$.

Once we obtain the transition matrix in canonical form, the fundamental matrix is the inverse $V = (I - U)^{-1}$. According to [45], we can use V to obtain $P(i) = (VW)_{i,1}$ and $S(i) = (V\mathbf{1})_i$, where $\mathbf{1}$ is a column vector of ones. If $S(i)$ is an expectation in the number of steps taken, we may even utilize V to calculate the variance $(2V - I)V\mathbf{1} - \text{Sq}(V\mathbf{1})$, where $\text{Sq}(V\mathbf{1})$ squares each entry of $V\mathbf{1}$. The fundamental matrix basically allows us to gather a number of meaningful statistics that we may want when evaluating a finite Markov chain.

The generic form of $V = (I - U)^{-1}$ for a bounded random walk can be found through various derivations, but regardless of which method is used, we find it easiest to write an entry of the fundamental matrix in terms of the following recurrences:

$$F(i) = \begin{cases} F(i-1) - L(i)R(i-1)F(i-2) \\ F(0) = 1 \\ F(-1) = 0 \end{cases} \quad (\text{A.28})$$

$$\bar{F}(i, k) = \begin{cases} F(i+1, k) - R(i)L(i+1)\bar{F}(i+2, k) \\ \bar{F}(k, k) = 1 \\ \bar{F}(k+1, k) = 0. \end{cases} \quad (\text{A.29})$$

The $\bar{F}(i, k)$ function mirrors $F(i)$, with k acting as the base. To give an example, if $k = 4$ and we start with

$$\left[I - U \mid I \right] = \left[\begin{array}{ccc|ccc} 1 & -R(1) & 0 & 1 & 0 & 0 \\ -L(2) & 1 & -R(2) & 0 & 1 & 0 \\ 0 & -L(3) & 1 & 0 & 0 & 1 \end{array} \right] \quad (\text{A.30})$$

then Gaussian elimination eventually yields

$$V = \left[\begin{array}{ccc} \frac{\bar{F}(2, 4)F(0)}{\bar{F}(1, 4)} & \frac{R(1)\bar{F}(3, 4)F(0)}{\bar{F}(1, 4)} & \frac{R(2)R(1)\bar{F}(4, 4)F(0)}{\bar{F}(1, 4)} \\ \frac{L(2)\bar{F}(3, 4)F(0)}{\bar{F}(1, 4)} & \frac{\bar{F}(3, 4)F(1)}{\bar{F}(1, 4)} & \frac{R(2)\bar{F}(4, 4)F(1)}{\bar{F}(1, 4)} \\ \frac{L(3)L(2)\bar{F}(4, 4)F(0)}{\bar{F}(1, 4)} & \frac{L(3)\bar{F}(4, 4)F(1)}{\bar{F}(1, 4)} & \frac{\bar{F}(4, 4)F(2)}{\bar{F}(1, 4)} \end{array} \right] \quad (\text{A.31})$$

as our inverse. Substituting an RDE (Equation 6.5) into V leads to Lemma 19, but to

realize this, we prove some identities on $F(i)$ and $\bar{F}(i, k)$ first so as to make the algebra less difficult to handle later on.

Lemma 17 *Let $F(i) = F(i-1) - \alpha\beta F(i-2)$ with initial conditions $F(-1) = 0$, $F(0) = 1$ and positive real numbers α, β such that $\alpha + \beta = 1$. Then*

$$F(i) = \sum_{j=0}^i \alpha^{i-j} \beta^j = \alpha^i + \alpha^{i-1} \beta + \dots + \alpha \beta^{i-1} + \beta^i. \quad (\text{A.32})$$

Moreover, $(\alpha - \beta) F(i) = \alpha^{i+1} - \beta^{i+1}$.

Proof: We prove the lemma by induction on i . The base cases are trivial to see: the first one is an empty sum, and the second one consists of a single term. Assuming $F(l)$ is true for all $l < i$, then

$$F(i) = (\alpha + \beta) \sum_{j=0}^{i-1} \alpha^{i-1-j} \beta^j - \alpha\beta \sum_{j=0}^{i-2} \alpha^{i-2-j} \beta^j \quad (\text{A.33})$$

$$= \sum_{j=0}^{i-1} \alpha^{i-j} \beta^j + \beta \sum_{j=0}^{i-2} \alpha^{i-1-j} \beta^j - \beta \sum_{j=0}^{i-2} \alpha^{i-1-j} \beta^j + \beta^i \quad (\text{A.34})$$

$$= \sum_{j=0}^{i-1} \alpha^{i-j} \beta^j + \beta^i = \sum_{j=0}^i \alpha^{i-j} \beta^j. \quad (\text{A.35})$$

For the second identity,

$$(\alpha - \beta) F(i) = (\alpha - \beta) \sum_{j=0}^i \alpha^{i-j} \beta^j \quad (\text{A.36})$$

$$= \alpha^{i+1} + \sum_{j=0}^{i-1} \alpha^{i-j} \beta^{j+1} - \sum_{j=0}^{i-1} \alpha^{i-j} \beta^{j+1} - \beta^{i+1} \quad (\text{A.37})$$

$$= \alpha^{i+1} - \beta^{i+1} \quad (\text{A.38})$$

which finishes the proof. ■

Lemma 18 *Let α, β be positive real numbers such that $\alpha + \beta = 1$. Let $k \geq 2$ be an integer. Then the two recurrences*

$$F(i) = F(i-1) - \alpha\beta F(i-2), \quad F(0) = 1, \quad F(-1) = 0 \quad (\text{A.39})$$

$$\bar{F}(i, k) = F(i+1, k) - \alpha\beta \bar{F}(i+2, k), \quad \bar{F}(k, k) = 1, \quad \bar{F}(k+1, k) = 0 \quad (\text{A.40})$$

are related by $\bar{F}(i, k) = F(k-i)$.

Proof: The induction goes in decreasing values of i . Immediately, we see $\bar{F}(k+1, k) = F(-1) = 0$ and $\bar{F}(k, k) = F(0) = 1$. Assuming $\bar{F}(j, k) = F(k-j)$ holds for all $j > i$, then

$$\bar{F}(i, k) = \bar{F}(i+1, k) - \alpha\beta \bar{F}(i+2, k) \quad (\text{A.41})$$

$$= F(k-(i+1)) - \alpha\beta F(k-(i+2)) \quad (\text{A.42})$$

$$= F(k-i-1) - \alpha\beta F(k-i-2) = F(k-i). \quad (\text{A.43})$$

This completes the proof. ■

Lemmas 19 and 20 describe what the fundamental matrix V will be in our use cases.

Lemma 19 *Let $L(i)$ be an RDE on a restricted probability specification (λ, γ) . If $L(i)$ determines the left step probabilities of a random walk over $\{0, \dots, k\}$, then the following describes the entries of the fundamental matrix V :*

$$V_{i,j} = \begin{cases} \frac{\lambda^{i-j} A_1(k-i) A_1(j) B_1(j-1)}{A_1(1) A_1(k) B_1(i-1)} & \text{if } i \geq j \\ \frac{A_1(k-j) A_1(i) B_1(j-1)}{A_1(1) A_1(k) B_1(i-1)} & \text{otherwise} \end{cases} \quad (\text{A.44})$$

where $A_1(i)$ and $B_1(i)$ are intermediate functions of (λ, γ) .

Proof: Let (α, β) be the boundaries of the probability specification (λ, γ) . After we adapt the example matrix in Equation A.31 with Equation 6.5, we check if what we get for V is in fact the inverse of $I - U$ (the block matrix U comes from the canonical representation of the transition matrix).

For starters, the non-recursive formulas of the left step $L(i)$ and right step $R(i)$ probability functions are

$$L(i) = \frac{\lambda B_1(i-2)}{B_1(i-1)} \qquad R(i) = \frac{B_1(i)}{B_1(i-1)}. \quad (\text{A.45})$$

Looking at Equation A.31, the pattern suggests

$$V_{i,j} = \begin{cases} \frac{\bar{F}(j+1, k) F(i-1)}{\bar{F}(1, k)} \prod_{l=i}^{j-1} R(l) & \text{if } i < j \\ \frac{\bar{F}(i+1, k) F(i-1)}{\bar{F}(1, k)} & \text{if } i = j \\ \frac{\bar{F}(i+1, k) F(j-1)}{\bar{F}(1, k)} \prod_{l=j+1}^i L(l) & \text{if } i > j. \end{cases} \quad (\text{A.46})$$

If we combine $L(i)R(i-1) = \lambda = \alpha\beta$, Lemma 17, Lemma 18, along with

$$\prod_{l=j+1}^i L(l) = \frac{\lambda B_1(j-1)}{B_1(j)} \frac{\lambda B_1(j)}{B_1(j+1)} \cdots \frac{\lambda B_1(i-2)}{B_1(i-1)} \quad (\text{A.47})$$

$$= \frac{\lambda^{i-j} B_1(j-1)}{B_1(i-1)} \quad (\text{A.48})$$

$$\prod_{l=i}^{j-1} R(l) = \frac{B_1(i)}{B_1(i-1)} \frac{B_1(i+1)}{B_1(i)} \cdots \frac{B_1(j-1)}{B_1(j-2)} \quad (\text{A.49})$$

$$= \frac{B_1(j-1)}{B_1(i-1)} \quad (\text{A.50})$$

then we obtain Equation A.44 above.

We validate $V_{i,j}$ as the last step in our proof. All rows and columns of $I - U$ have at most three non-zero entries, all lying near the main diagonal. When we examine row i of $I - U$ and column j of V such that $i < j$, we get

$$\begin{aligned} ((I - U)V)_{i,j} &= -\frac{\lambda B_1(i-2)}{B_1(i-1)} \frac{A_1(k-j)A_1(i-1)B_1(j-1)}{A_1(1)A_1(k)B_1(i-2)} \\ &\quad + \frac{A_1(k-j)A_1(i)B_1(j-1)}{A_1(1)A_1(k)B_1(i-1)} \\ &\quad - \frac{B_1(i)}{B_1(i-1)} \frac{A_1(k-j)A_1(i+1)B_1(j-1)}{A_1(1)A_1(k)B_1(i)} = 0 \end{aligned} \quad (\text{A.51})$$

by Lemma 16i. The special case $i = 1 < j$ involves only two terms, but the result remains the same since $A_1(2) = A_1(1)$. The other situations follow similarly, where $((I - U)V)_{i,j} = 1$ when $i = j$ and $((I - U)V)_{i,j} = 0$ when $i > j$. The same logic applies if we also look at $V(I - U)$. ■

Lemma 20 *The fundamental matrix V for a uniform random walk over $\{0, \dots, k\}$ is*

$$V_{i,j} = \begin{cases} \frac{2(k-i)j}{k} & \text{if } i \geq j \\ \frac{2(k-j)i}{k} & \text{otherwise.} \end{cases} \quad (\text{A.52})$$

Proof: We have $F(i) = (i+1)/2^i$ as a consequence of $\alpha = \beta = 1/2$ by Lemma 8.

For $i < j$, the entries are

$$V_{i,j} = \frac{\bar{F}(j+1, k)F(i-1)}{\bar{F}(1, k)} \prod_{l=i}^{j-1} R(l) \quad (\text{A.53})$$

$$= \frac{k-j}{2^{k-j-1}} \frac{i}{2^{i-1}} \frac{2^{k-1}}{k} \frac{1}{2^{j-i}} = \frac{(k-j)i}{2^{-1}k}. \quad (\text{A.54})$$

The other case is similar. ■

A.2.3 Identities on Fundamental Matrix Entries

Given the fundamental matrix of a random walk on $\{0, \dots, k\}$, we sum across row i to compute the expected steps $S(i)$. We can separate the summation into two parts, one that goes from column 1 to column i and another from $i + 1$ to $k - 1$. We show a couple identities on these two smaller sums before the final proof.

Lemma 21 *Let $A_1(i)$ and $B_1(i)$ be intermediate functions of a restricted probability specification (λ, γ) . Then for all integers $i \geq 0$,*

$$J_1(i) = \sum_{j=0}^i \lambda^{i-j} B_1(j) A_1(j+1) \quad (\text{A.55})$$

$$= \frac{B_1(2i+2)}{A_1(1)} - ((2i+3)\lambda - (i+1)\gamma) \lambda^i. \quad (\text{A.56})$$

Proof: Recognizing $A_1(3) = A_2(2)A_1(1) + \lambda A_1(1)$ and $A_1(2) = A_1(1)$, we show

$$J_1(0) = \frac{A_1(3) - \gamma A_1(2)}{A_1(1)} - 3\lambda + \gamma \quad (\text{A.57})$$

$$= \frac{A_1(1)(A_2(2) + \lambda - \gamma)}{A_1(1)} - 3\lambda + \gamma \quad (\text{A.58})$$

$$= A_2(2) - 2\lambda = A_1(1)A_1(1). \quad (\text{A.59})$$

This acts as a base case for an induction on $J_1(i) = \lambda J_1(i-1) + B_1(i)A_1(i+1)$. If we continue forward, then

$$J_1(i) = B_1(i)A_1(i+1) + \frac{\lambda B_1(2i)}{A_1(1)} - ((2i+1)\lambda - i\gamma) \lambda^i \quad (\text{A.60})$$

$$= \frac{B_1(2i+2)}{A_1(1)} - 2\lambda^{i+1} + \gamma\lambda^i - (2i+1)\lambda^{i+1} + i\gamma\lambda^i \quad (\text{A.61})$$

as a result of Lemma 16 v. ■

Lemma 22 *Let $A_1(i)$, $B_1(i)$, and $B_2(i)$ be intermediate functions of a restricted probability specification (λ, γ) . Let $k \geq 2$ be an integer. Then for all integers $i \geq 1$,*

$$J_2(i) = \sum_{j=1}^i B_1(k-j-1)A_1(j) \quad (\text{A.62})$$

$$= (i+1)B_2(k-1) - \frac{A_1(i+1)}{A_1(1)}B_2(k-i-1). \quad (\text{A.63})$$

Proof: Again, we give a proof by induction. Starting with $i = 1$,

$$J_2(1) = B_1(k-1-1)A_1(1) + B_2(k-1) - B_2(k-1) \quad (\text{A.64})$$

$$= 2B_2(k-1) - \frac{A_1(1)(B_2(k-1) + \lambda B_2(k-3))}{A_1(1)} \quad (\text{A.65})$$

$$= 2B_2(k-1) - \frac{A_1(2)}{A_1(1)}B_2(k-2) \quad (\text{A.66})$$

using Lemma 16. Assuming $J_2(j)$ is true for all $j < i$, let us look at

$$J_2(i) = B_1(k-i-1)A_1(i) + J_2(i-1) \quad (\text{A.67})$$

$$= B_1(k-i-1)A_1(i) + iB_2(k-1) - \frac{A_1(i)}{A_1(1)}B_2(k-i). \quad (\text{A.68})$$

By Lemma 16 vi, we end up with

$$J_2(i) = (i+1)B_2(k-1) - \lambda^i B_2(k-2i-1) - \frac{A_1(i)}{A_1(1)}B_2(k-i). \quad (\text{A.69})$$

After gathering the last two terms under a common denominator, the numerator becomes

$$-\lambda^i B_2(k-2i-1)A_1(1) - B_2(k-i)A_1(i) = -B_2(k-i-1)A_1(i+1) \quad (\text{A.70})$$

due to Lemma 16 viii. ■

A.2.4 Proofs of Lemmas 14 and 15

We are ready to solve Equations A.1 and A.2 for a restricted (λ, γ) and $\lambda = 1/4$.

Proof: (Lemma 14) More formally,

$$S(i) = \sum_{j=1}^{k-1} V_{i,j} = \frac{A_1(k-i)J_1(i-1) + A_1(i)J_2(k-i-1)}{A_1(1)A_1(k)B_1(i-1)} \quad (\text{A.71})$$

where $J_1(i-1)$ and $J_2(k-i-1)$ are defined in Lemmas 21 and 22. Note that $J_2(k-i-1)$ starts the summation index from the right end of the fundamental matrix V and moves inward. With the help of

$$A_1(i)B_2(i) = A_1(i)A_2(i+1) - \gamma A_1(i)A_2(i) \quad (\text{A.72})$$

$$= A_1(2i+1) - \gamma A_1(2i) - \lambda^i A_1(1) \quad (\text{A.73})$$

$$= B_1(2i) - \lambda^i A_1(1) \quad (\text{A.74})$$

and Lemma 16, we arrive at

$$A_1(i)J_2(k-i-1) = (k-i)A_1(i)B_2(k-1) \quad (\text{A.75})$$

$$- \frac{A_1(k-i)}{A_1(1)}B_1(2i) + \lambda^i A_1(k-i). \quad (\text{A.76})$$

Then combining it with

$$A_1(k-i)J_1(i-1) = \frac{A_1(k-i)}{A_1(1)}B_1(2i) \quad (\text{A.77})$$

$$- A_1(k-i)((2i+1)\lambda - i\gamma)\lambda^{i-1} \quad (\text{A.78})$$

we see that a couple terms cancel out, leaving Equation A.4 as desired.

The derivation of $P(i)$ from fundamental matrix V is easier by comparison. Recall that $P(i) = (VW)_{i,1}$, where W is a $(k-1) \times 2$ matrix with $W_{1,1} = \gamma$ and 0 for the rest of column 1. As such,

$$P(i) = \gamma V_{i,1} = \frac{A_1(k-i)A_1(1)B_1(0)\gamma\lambda^{i-1}}{A_1(1)A_1(k)B_1(i-1)} \quad (\text{A.79})$$

$$= \frac{A_1(1)A_1(k-i)\gamma\lambda^{i-1}}{A_1(k)B_1(i-1)} \quad (\text{A.80})$$

since $B_1(0) = A_1(1)$. ■

Proof: (Lemma 15) The solutions are already discussed in [24], but we reach the same conclusion by way of Lemma 20. Accordingly,

$$S(i) = \sum_{j=1}^i \frac{2(k-i)j}{k} + \sum_{j=i+1}^{k-1} \frac{2(k-j)i}{k} \quad (\text{A.81})$$

$$= \frac{2(k-i)(i+1)i}{k} + \frac{2i(k-i)(k-i-1)}{2} \quad (\text{A.82})$$

$$= \frac{(i+1+k-i-1)(k-i)i}{k} = ki - i^2. \quad (\text{A.83})$$

The $P(i)$ solution is simpler to derive. ■

Lemmas 14 and 15 play a significant part in achieving our nested recovery protocol analysis in Chapter 6, but whether they also have value in other areas of mathematics is unclear. It might be interesting to see if the generic matrix form and recurrence equations here can be adapted according to some other transition function to possibly reveal interesting applications unbeknownst to us at the moment.

Appendix B

Index of Terms

Page numbers indicate where concept is introduced or defined.

basic execution, 101

binary connected, 99

binary in-tree, 92

binary in-tree Clifford unitary, 99

binary in-tree decomposition, 97

Bloch sphere, 6

Bloch vector, 9

Clifford circuit, 25

Clifford equivalent postselected stabilizer circuits, 40

Clifford operator, 23

Clifford two-operator, 95

delegated two-op circuit, 97

density matrix, 8

doubly weight-one entry, 111

entangled quantum state, 8

equivalent Clifford unitaries, 98

equivalent postselected stabilizer circuits, 40

fundamental matrix, 143

global phase, 10

Gottesman-Knill theorem, 26

in-tree, 92

inactive qubit, 98

interacting postselected stabilizer circuit, 53

leading Pauli gate, 108

magic state, 34

mixed quantum state, 9

multistep tree execution, 101

nested recovery protocol, 65

nested recovery protocol expected cost, 78

nested recovery protocol expected demand, 78

nested recovery protocol startup cost, 78

nested recovery protocol success probability, 78

Pauli group, 20

postselected stabilizer circuit, 38

postselected stabilizer circuit expected cost (n -to-one), 102

postselected stabilizer circuit output, 38

postselected stabilizer circuit probability, 38

postselected two-op circuit, 96

postselected two-op circuit expected cost, 102

postselected two-op circuit output, 96

postselected two-op circuit probability, 96

probability specification, 76

probability specification boundaries, 76

probability specification intermediate functions, 77

product quantum state, 8

pure stabilizer state, 22

pure quantum state, 8

qubit, 5

random walk, 138

random walk expected number of steps, 139

random walk success probability, 139

rational difference equation, 77

recovery circuit, 54

reduced row echelon form, 109

restricted probability specification, 76

row echelon form, 108

stabilizer circuit, 25

stabilizer formalism, 18

stabilizer group, 21

stabilizer matrix, 107

stabilizer mixed state, 22

stabilizer operation, 26

stabilizer state, 22

stabilizer state, pure, see *pure stabilizer state*

unitary stabilizer circuit, see *Clifford circuit*

unitary transformation (operator), 10

universal quantum computation, 28

Bibliography

- [1] “Applications of quantum computing.” <https://www.research.ibm.com/ibm-q/learn/quantum-computing-applications>. Accessed: 2018-04-01.
- [2] “UK National Quantum Technologies Programme.” <http://uknqt.epsrc.ac.uk>. Accessed: 2018-04-01.
- [3] S. Aaronson and D. Gottesman, *Improved simulation of stabilizer circuits*, *Phys. Rev. A* **70** (Nov, 2004) 052328.
- [4] D. S. Abrams and S. Lloyd, *Simulation of many-body fermi systems on a universal quantum computer*, *Phys. Rev. Lett.* **79** (Sep, 1997) 2586–2589.
- [5] D. Aharonov and M. Ben-Or, *Fault-tolerant quantum computation with constant error*, in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, (New York, NY, USA), pp. 176–188, ACM, 1997.
- [6] A. Ambainis, *Understanding Quantum Algorithms via Query Complexity*, in *Proceedings of the International Congress of Mathematicians*, ICM '18, (Berlin, Germany), International Mathematical Union, 2018.
- [7] S. Anders and H. J. Briegel, *Fast simulation of stabilizer circuits using a graph-state representation*, *Phys. Rev. A* **73** (Feb, 2006) 022334.
- [8] K. M. R. Audenaert and M. B. Plenio, *Entanglement on mixed stabilizer states: normal forms and reduction procedures*, *New Journal of Physics* **7** (2005) 170.
- [9] C. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing*, in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, (New York, NY, USA), pp. 175–179, IEEE, 1984.
- [10] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels*, *Phys. Rev. Lett.* **70** (Mar, 1993) 1895–1899.
- [11] A. Bocharov, Y. Gurevich, and K. M. Svore, *Efficient decomposition of single-qubit gates into v basis circuits*, *Phys. Rev. A* **88** (Jul, 2013) 012313.

- [12] A. Bocharov, M. Roetteler, and K. M. Svore, *Efficient synthesis of probabilistic quantum circuits with fallback*, *Phys. Rev. A* **91** (May, 2015) 052317.
- [13] A. Bocharov, M. Roetteler, and K. M. Svore, *Efficient synthesis of universal repeat-until-success quantum circuits*, *Phys. Rev. Lett.* **114** (Feb, 2015) 080502.
- [14] B. M. Boghosian and W. Taylor, *Simulating quantum mechanics on a quantum computer*, *Physica D: Nonlinear Phenomena* **120** (1998), no. 1 30 – 42.
Proceedings of the Fourth Workshop on Physics and Consumption.
- [15] S. Bravyi and J. Haah, *Magic-state distillation with low overhead*, *Phys. Rev. A* **86** (Nov, 2012) 052329.
- [16] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal clifford gates and noisy ancillas*, *Phys. Rev. A* **71** (Feb, 2005) 022316.
- [17] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. Sloane, *Quantum error correction via codes over $gf(4)$* , *IEEE Trans. Inf. Theor.* **44** (July, 1998) 1369–1387.
- [18] E. Campbell and D. Browne, *On the Structure of Protocols for Magic State Distillation*, in *Proceedings of the Fourth Workshop on Theory of Quantum Computation, Communication and Cryptography, TQC '09*, (Heidelberg, Germany), pp. 20–32, Springer-Verlag Berlin Heidelberg, 2009.
- [19] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost*, *Phys. Rev. A* **95** (Feb, 2017) 022316.
- [20] E. T. Campbell and M. Howard, *Unifying gate synthesis and magic state distillation*, *Phys. Rev. Lett.* **118** (Feb, 2017) 060501.
- [21] E. T. Campbell and J. O’Gorman, *An efficient magic state approach to small angle rotations*, *Quantum Science and Technology* **1** (2016), no. 1 015007.
- [22] A. Das and B. K. Chakrabarti, *Colloquium: Quantum annealing and analog quantum computation*, *Rev. Mod. Phys.* **80** (Sep, 2008) 1061–1081.
- [23] C. M. Dawson and M. A. Nielsen, *The solovay-kitaev algorithm*, *Quantum Info. Comput.* **6** (Jan., 2006) 81–95.
- [24] P. G. Doyle and J. L. Snell, *Random walks and Electric Networks*. Mathematical Association of America, 1984.
- [25] G. Duclos-Cianci and D. Poulin, *Reducing the quantum-computing overhead with complex gate distillation*, *Phys. Rev. A* **91** (Apr, 2015) 042315.

- [26] G. Duclos-Cianci and K. M. Svore, *Distillation of nonstabilizer states for universal quantum computation*, *Phys. Rev. A* **88** (Oct, 2013) 042325.
- [27] B. Eastin and E. Knill, *Restrictions on transversal encoded quantum gate sets*, *Phys. Rev. Lett.* **102** (Mar, 2009) 110502.
- [28] A. K. Ekert, *Quantum cryptography based on bell's theorem*, *Phys. Rev. Lett.* **67** (Aug, 1991) 661–663.
- [29] R. P. Feynman, *Simulating Physics with Computers*, *International Journal of Theoretical Physics* **21** (June, 1982) 467–488.
- [30] S. Forest, D. Gosset, V. Kliuchnikov, and D. McKinnon, *Exact synthesis of single-qubit unitaries over clifford-cyclotomic gate sets*, *Journal of Mathematical Physics* **56** (2015), no. 8 082201.
- [31] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface codes: Towards practical large-scale quantum computation*, *Phys. Rev. A* **86** (Sep, 2012) 032324.
- [32] B. Giles and P. Selinger, *Exact synthesis of multiqubit clifford+t circuits*, *Phys. Rev. A* **87** (Mar, 2013) 032332.
- [33] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997. <http://arxiv.org/abs/quant-ph/9705052>.
- [34] D. Gottesman, *The Heisenberg representation of quantum computers*, in *Group theoretical methods in physics. Proceedings, 22nd International Colloquium, Group22, ICGTMP'98, Hobart, Australia, July 13-17, 1998*, pp. 32–43, 1998. [quant-ph/9807006](http://arxiv.org/abs/quant-ph/9807006).
- [35] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (New York, NY, USA), pp. 212–219, ACM, 1996.
- [36] J. Haah, M. B. Hastings, D. Poulin, and D. Wecker, *Magic State Distillation with Low Space Overhead and Optimal Asymptotic Input Count*, *Quantum* **1** (Oct., 2017) 31.
- [37] M. B. Hastings, D. Wecker, B. Bauer, and M. Troyer, *Improving quantum algorithms for quantum chemistry*, *Quantum Info. Comput.* **15** (Jan., 2015) 1–21.
- [38] M. Howard, J. Wallman, V. Veitch, and J. Emerson, *Contextuality supplies the 'magic' for quantum computation*, *Nature* **510** (Jun, 2014).
- [39] C. Jones, *Multilevel distillation of magic states for quantum computing*, *Phys. Rev. A* **87** (Apr, 2013) 042305.

- [40] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, *Layered architecture for quantum computing*, *Phys. Rev. X* **2** (Jul, 2012) 031007.
- [41] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, *Faster quantum chemistry simulation on fault-tolerant quantum computers*, *New Journal of Physics* **14** (2012) 115023.
- [42] S. Jordan, “Quantum Algorithm Zoo.” <https://math.nist.gov/quantum/zoo>. Accessed: 2018-04-01.
- [43] R. Jozsa and N. Linden, *On the role of entanglement in quantum-computational speed-up*, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **459** (2003), no. 2036 2011–2032.
- [44] J. Kelly, “A Preview of Bristlecone, Google’s New Quantum Processor.” <https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html>. Accessed: 2018-04-01.
- [45] J. Kemény and J. Snell, *Finite markov chains*. University series in undergraduate mathematics. Springer-Verlag New York, 1976.
- [46] M. M. Khan, M. Murphy, and A. Beige, *High error-rate quantum key distribution for long-distance communication*, *New Journal of Physics* **11** (2009) 063043.
- [47] V. Kliuchnikov, D. Maslov, and M. Mosca, *Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits using a constant number of ancillary qubits*, *Phys. Rev. Lett.* **110** (May, 2013) 190502.
- [48] V. Kliuchnikov, D. Maslov, and M. Mosca, *Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates*, *Quantum Information and Computation* **13** (2013), no. 7-8 607–630.
- [49] E. Knill, *Quantum computing with realistically noisy devices*, *Nature* **434** (Mar, 2005).
- [50] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O’Brien, *Quantum computers*, *Nature* **464** (Mar, 2010).
- [51] S.-K. Liao, W.-Q. Cai, J. Handsteiner, B. Liu, J. Yin, L. Zhang, D. Rauch, M. Fink, J.-G. Ren, W.-Y. Liu, Y. Li, Q. Shen, Y. Cao, F.-Z. Li, J.-F. Wang, Y.-M. Huang, L. Deng, T. Xi, L. Ma, T. Hu, L. Li, N.-L. Liu, F. Koidl, P. Wang, Y.-A. Chen, X.-B. Wang, M. Steindorfer, G. Kirchner, C.-Y. Lu, R. Shu, R. Ursin, T. Scheidl, C.-Z. Peng, J.-Y. Wang, A. Zeilinger, and J.-W. Pan, *Satellite-relayed intercontinental quantum network*, *Phys. Rev. Lett.* **120** (Jan, 2018) 030501.

- [52] A. Meier, B. Eastin, and E. Knill, *Magic-state distillation with the four-qubit code*, *Quantum Information and Computation* **13** (2013) 195–209.
- [53] J. A. Miszczak, *Models of quantum computation and quantum programming languages*, *Bulletin of the Polish Academy of Sciences, Technical Sciences* **59** (Nov, 2011) 305–324.
- [54] Y. Mu, J. Seberry, and Y. Zheng, *Shared cryptographic bits via quantized quadrature phase amplitudes of light*, *Optics Communications* **123** (1996), no. 1 344 – 352.
- [55] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. Das Sarma, *Non-abelian anyons and topological quantum computation*, *Rev. Mod. Phys.* **80** (Sep, 2008) 1083–1159.
- [56] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [57] A. Paetznick and B. W. Reichardt, *Universal fault-tolerant quantum computation with only transversal gates and error correction*, *Phys. Rev. Lett.* **111** (Aug, 2013) 090505.
- [58] A. Paetznick and K. M. Svore, *Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries*, *Quantum Info. Comput.* **14** (Nov., 2014) 1277–1301.
- [59] C. J. Pugh, S. Kaiser, J.-P. Bourgoin, J. Jin, N. Sultana, S. Agne, E. Anisimova, V. Makarov, E. Choi, B. L. Higgins, and T. Jennewein, *Airborne demonstration of a quantum key distribution receiver payload*, *Quantum Science and Technology* **2** (2017), no. 2 024009.
- [60] H. Ramesh and V. Vinay, *String matching in $O(\sqrt{n} + \sqrt{m})$ quantum time*, *Journal of Discrete Algorithms* **1** (2003), no. 1 103 – 110. Combinatorial Algorithms.
- [61] B. Reichardt, *Quantum universality by state distillation*, *Quantum Information and Computation* **9** (2009) 1030–1052.
- [62] N. J. Ross, *Optimal ancilla-free Clifford+V approximation of z-rotations*, *Quantum Information and Computation* **15** (2015), no. 11-12 932–950.
- [63] N. J. Ross and P. Selinger, *Optimal ancilla-free Clifford+T approximation of z-rotations*, *Quantum Information and Computation* **16** no. 11-12 901–953.
- [64] P. Selinger, *Efficient Clifford+T approximation of single-qubit operators*, *Quantum Information and Computation* **15** (2015), no. 1-2 159–180.
- [65] P. Selinger, *Generators and relations for n-qubit Clifford operators*, *Logical Methods in Computer Science* **Volume 11, Issue 2** (June, 2015).

- [66] Y. Shi, *Both toffoli and controlled-not need little help to do universal quantum computing*, *Quantum Info. Comput.* **3** (Jan., 2003) 84–92.
- [67] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, *SIAM Journal on Computing* **26** (1997), no. 5 1484–1509.
- [68] W. T. Tutte, *Graph Theory*. Cambridge University Press, 2001.
- [69] W. van Dam and R. Wong, *Two-qubit Stabilizer Circuits with Recovery I: Existence*, in *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)* (S. Jeffery, ed.), vol. 111 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 7:1–7:15, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
- [70] W. van Dam and R. Wong, *Two-qubit Stabilizer Circuits with Recovery II: Analysis*, in *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)* (S. Jeffery, ed.), vol. 111 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 8:1–8:21, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
- [71] A. Vardy, *Algorithmic complexity in coding theory and the minimum distance problem*, in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, (New York, NY, USA), pp. 92–109, ACM, 1997.
- [72] V. Veitch, S. A. H. Mousavian, D. Gottesman, and J. Emerson, *The resource theory of stabilizer quantum computation*, *New Journal of Physics* **16** (2014) 013009.
- [73] N. Wiebe and V. Kliuchnikov, *Floating point representations in quantum circuit synthesis*, *New Journal of Physics* **15** (2013) 093041.
- [74] W. K. Wootters and W. H. Zurek, *A single quantum cannot be cloned*, *Nature* **299** (October, 1982).
- [75] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, G.-B. Li, Q.-M. Lu, Y.-H. Gong, Y. Xu, S.-L. Li, F.-Z. Li, Y.-Y. Yin, Z.-Q. Jiang, M. Li, J.-J. Jia, G. Ren, D. He, Y.-L. Zhou, X.-X. Zhang, N. Wang, X. Chang, Z.-C. Zhu, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, R. Shu, C.-Z. Peng, J.-Y. Wang, and J.-W. Pan, *Satellite-based entanglement distribution over 1200 kilometers*, *Science* **356** (2017), no. 6343 1140–1144.