Clifford Simulation: Techniques and Applications

by

Alexander Kerzner

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization (Quantum Information)

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

- Chapter 1 contains my own summary of previous results from [16, 1, 26] (Section 1.1), [6] (Section 1.2), [12, 34] (Section 1.3), and [18, 3] (Section 1.4). The incorporation of phase sensitivity and fast matrix multiplication into Section 1.3 is new, and was aided by discussions with Daniel Grier and Luke Schaeffer.

- Chapter 2, with the exception of Section 2.6, is a summary of joint work with David Gosset, Daniel Grier, and Luke Schaeffer [15], in which Section 2.5 appears verbatim.

- The results of Chapter 3 are based on discussions with David Gosset.

Abstract

Despite the widespread belief that quantum computers cannot be efficiently simulated classically, efficient simulation is known to be possible in certain restricted regimes. In particular, the Gottesman-Knill theorem states that Clifford circuits can be efficiently simulated. We begin this thesis by reviewing and comparing several known techniques for efficient simulation of Clifford circuits: the stabilizer formalism, CH form, affine form, and the graph state formalism. We describe each simulation method and give four different proofs of the Gottesman-Knill theorem.

Next we review a recent work [15], which shows that restricting the geometry of Clifford circuits can lead to a further speedup. We give an algorithm for simulating Pauli basis measurements on a planar graph state in time $\widetilde{O}(n^{\omega/2})$, where $\omega < 2.373$ is the matrix multiplication exponent. This algorithm achieves a quadratic speedup over using Clifford simulation methods directly. As an application of this algorithm, we consider a depth-d Clifford circuit whose two-qubit gates act along edges of a planar graph and describe how to sample from its output distribution or compute an output probability in time $\widetilde{O}(n^{\omega/2}d^{\omega})$. For $d = O(\log n)$, both of these results are quadratic speedups over using Clifford simulation methods directly.

Finally, we extend these simulation algorithms to universal circuits by using stabilizer rank methods. We follow a previously known gadgetization procedure [9] to show that given a depth-d Clifford+T circuit with t T gates and whose two-qubit gates act along edges of a planar graph, we can sample from its output distribution in time $\widetilde{O}(2^{0.7926t}n^{5/2}t^6d^3)$ and can compute output probabilities in time $\widetilde{O}(2^{0.3963t}n^{3/2}t^6d^3)$. Previous work [9, 6], applied to the case $d = O(\log n)$, gives algorithms for sampling in time $O(2^{0.3963t}n^6t^6)$ and computation of output probabilities in time $O(2^{0.3963t}n^3t^3)$. Our sampling algorithm offers improved scaling in n but poorer scaling in the exponential term, while our algorithm for computing output probabilities offers improved scaling in n with identical scaling in the exponential term.

Acknowledgements

This thesis would not have been possible without the guidance of my excellent supervisor, David Gosset. I am especially grateful to have been introduced to the interesting area of Clifford simulation and for invaluable advice on all things relating to research, writing, and presenting. On the nontechnical side, I am also very thankful for his patience and encouragement.

I would also like to express my gratitude to Daniel Grier and Luke Schaeffer for their enthusiasm, many technical discussions, and for teaching me that all problems can be reduced to matrix multiplication with enough persistence.

Feedback from Richard Cleve and Debbie Leung, both of whom generously agreed to read this thesis, was also very helpful and much appreciated.

Although in-person time was limited, working at IQC and in C&O has been a wonderful experience. Many thanks to all of my classmates and professors for interesting classes, lectures, and cookie time discussions.

Over the past few years I have also been lucky to have the support of Anne Broadbent, Bradd Hart, Supartha Podder, Bartosz Protas, and Douglas Stebila. I am deeply indebted to all of them for helping to improve my skills as a student and researcher.

Finally, I would like to thank Avrum, Cathy, and Chloe Kerzner for more things than I have space to list here.

Contents

List of Figures vii

List of Tables viii

Overview 1

1 Clifford circuit simulation 5
- 1.1 The stabilizer formalism . . . 6
- 1.2 CH form . . . 11
- 1.3 Affine form . . . 13
- 1.4 The graph state formalism . . . 17
- 1.5 Discussion . . . 23

2 Graph state simulation 24
- 2.1 Algorithms for graph state simulation . . . 26
- 2.2 Tree decompositions . . . 28
- 2.3 Sampling subroutine . . . 29
- 2.4 Correction subroutine . . . 34
- 2.5 Correction subroutine example . . . 39
- 2.6 Variants of the graph state simulation problem . . . 42
- 2.7 Applications of graph state simulation . . . 43
- 2.8 Discussion . . . 52

3 Stabilizer rank methods 54
- 3.1 Planar Clifford+T circuits . . . 55

References 60

A Fast linear algebra 63

List of Figures

1 A simple quantum circuit 2

1.1 Local complementation 19
1.2 Circuit diagram depiction of using Claim 8 to apply a CZ gate. 20

2.1 Solving the graph state simulation problem on the star graph. 27
2.2 Solving the graph state simulation problem on the $\sqrt{n} \times \sqrt{n}$ grid graph. . . 27
2.3 A graph and tree decomposition. 28
2.4 Gadgets needed to define the circuit $\mathcal{C}$. 30
2.5 A graph, nice tree decomposition, and circuit $\mathcal{C}$. 31
2.6 Commutation relations needed for Lemma 15. 32
2.7 Behaviour of stabilizer Z-components when CNOT and CZ gates are applied. 39
2.8 A graph, nice tree decomposition, and circuit $\mathcal{C}$. 40
2.9 A circuit whose CZ gates act along the edges of a planar graph. 44
2.10 A 5-coarse-graining. 44
2.11 Measurement-based quantum computing gadgets. 45
2.12 Replacing a circuit with a measurement-based computation. 45
2.13 Simulating a circuit using a tensor network contraction algorithm. 47
2.14 Example of contracting an index shared by two tensors. 49
2.15 Replacing a tensor network with a measurement-based computation. 50

3.1 The T gate gadget. 54
3.2 Example of the formation of $C_{\text{gadg}}(I \otimes D^{(j)})$ and G_j. 57
3.3 A small example of the circuit $\mathcal{C}$, defined by Eq. 3.11 58

List of Tables

1.1 Recreated from [26]. Lookup table for updating a stabilizer group after applying the gate $U \in \{H, S, CZ\}$. 7
1.2 Time complexity of performing various operations on stabilizer states when using each of the Clifford simulation techniques of Chapter 1. 23

Overview

The topic of this thesis is classical simulation of quantum computation. It is widely believed that classical computers cannot efficiently simulate quantum computers in general, so why should we pursue simulation in the first place?

The simplest reason is to offer support to this belief. After all, perhaps we just haven't looked hard enough for an efficient simulation algorithm. Just as hardness assumptions in cryptography become more widely accepted after standing the test of time, so too can our confidence in the superiority of quantum computers increase if attempt after attempt to find an efficient simulation algorithm comes up empty.

Another reason is that studying classical simulation can shed light on exactly why quantum computers seem to be more powerful than classical ones. Superposition and entanglement are both typically cited as the source of quantum advantage. However, this is an incomplete characterization, as Clifford circuits display both of these properties yet are still efficiently simulable [16]. Having a more nuanced understanding of the nature of quantum computing's power may help us determine which information processing tasks admit a quantum speedup, and how to maximize the speedup when one exists.

On the practical side, studying both the theory and implementation of classical simulation algorithms can help justify the usefulness of quantum computers for particular tasks. For the foreseeable future, quantum computers will be far more difficult to build than classical computers. Given this difficulty, anyone wanting to build a quantum computer for the purpose of running a particular quantum algorithm will likely want a guarantee that a classical computer could not simulate that algorithm in a comparable amount of time. The magnitude of a quantum speedup may also play a role. Is it worth building a quantum computer for a polynomial speedup? What about an exponential speedup?

We now state some standard definitions. An n-qubit quantum state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$. The conjugate transpose of $|\psi\rangle$ is denoted $\langle\psi|$. The computational basis is the set $\{|z\rangle : z \in \{0,1\}^n\}$. For $k \leq n$, a k-qubit gate is a unitary operator in $\mathbb{C}^{2^n \times 2^n}$ which acts nontrivially only on k qubits of any given state. We often express such a gate as an operator in $\mathbb{C}^{2^k \times 2^k}$ and extend its action to $\mathbb{C}^{2^n \times 2^n}$ by taking a tensor product with the identity operator. A quantum circuit is a product of gates. If a circuit is expressed as the product of several gates, we can express it using a circuit diagram. In Fig. 1, each rectangle represents a gate, and each horizontal line, called a wire, represents a qubit. A state is given as input to the circuit on the left hand side and proceeds in discrete time steps from left to right. When it passes through a gate, that gate is applied to the state. The wires passing through a gate correspond to the qubits on which that gate acts. If two gates act on disjoint subsets of qubits then they commute, and therefore may be drawn as though they act simultaneously.

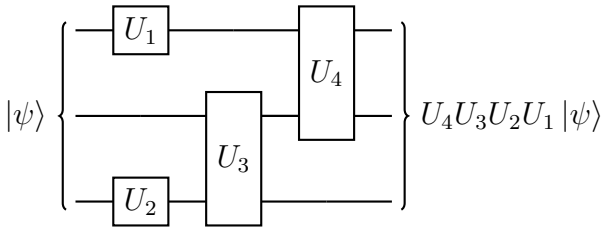


Figure 1: A depth-3 circuit that takes as input the 3-qubit state $|\psi\rangle$, applies the gates U_1, U_2, U_3, and U_4, and outputs $U_4U_3U_2U_1\,|\psi\rangle$.

We say that the depth of a circuit is the number of discrete time steps needed. Because some gates may be applied simultaneously, the depth may be less than the number of gates.

The two most basic simulation tasks are sampling from a circuit's output distribution and computing an output probability. Given an n-qubit circuit C, sampling from its output distribution means sampling $z \in \{0,1\}^n$ from the distribution

$$\Pr[z] = |\langle z|C|0^n\rangle|^2 . \tag{1}$$

Computing an output probability refers to, when given a fixed $z \in \{0,1\}^n$, computing $\Pr[z]$. The sampling task is what a noiseless quantum computer does. Computation of output probabilities is not performed by quantum computers, but can be a useful tool for verifying their correctness: Consider a quantum computer that applies the circuit C to input $|0^n\rangle$ and performs a computational basis measurement. If $\Pr[z]$ is known, then comparing it to the number of times that result z is observed can help us determine whether or not the quantum computer is working correctly.

Both simulation tasks can be performed in exponential time[1] on a classical computer as follows. Suppose $C = U_m \dots U_1$, where each U_j is a one- or two-qubit gate. We start by storing $|0^n\rangle$ in memory as a vector in $\mathbb{C}^{2^n}$. Next we sequentially compute $U_1\,|0^n\rangle$, then $U_2(U_1\,|0^n\rangle)$, and so on until we reach $U_m(U_{m-1}\dots U_1\,|0^n\rangle) = C\,|0^n\rangle$. Each U_j is represented by a matrix with $O(1)$ nonzero entries, so computing each $U_j(U_{j-1}\dots U_1\,|0^n\rangle)$ amounts to a sparse matrix-vector multiplication. Therefore, $C\,|0^n\rangle$ may be computed in time $O(2^n m)$. To compute an output probability we can then look at the z-entry of $C\,|0^n\rangle$ and take the modulus squared. To sample, we can then use $C\,|0^n\rangle$ to learn the entire distribution by computing $\Pr[z]$ for each $z \in \{0,1\}^n$, which takes time $O(2^n)$, and then use any technique for sampling from a discrete distribution.[2]

In contrast to the exponential runtime that seems to be needed in general, there exist certain restricted classes of circuits that admit faster classical simulation.

A classic result of Markov and Shi [25] offers an improved simulation algorithm based on the geometry of C. Consider a graph G with vertex set $[m]$ and an edge ij whenever a wire segment joins U_i and U_j in a circuit diagram representation of C. Markov and Shi show that either simulation task can be performed classically in time $2^{O(\mathrm{tw}(G))}m^{O(1)}$. Here $\mathrm{tw}(G)$ denotes treewidth [30], a graph-theoretic quantity defined formally in Chapter 2 which

[1] By time, we mean the number of arithmetic operations.

[2] For example, if we can generate a uniformly random $r \in (0,1)$, then we can sample from $\Pr[z]$ by selecting $\min\{z : r < \sum_{y \le z} \Pr[y]\}$.

roughly measures how much G resembles a tree. Therefore if G is planar, using the fact that a planar graph on k vertices has treewidth $O(\sqrt{k})$ together with the fact that $m \leq dn$, the algorithm runs in time $2^{O(\sqrt{dn})}(dn)^{O(1)}$. For low-depth circuits, this is an improvement over the naïve algorithm described above, which has runtime $O(2^n dn)$. Finding improved simulation algorithms exploiting geometry is an active area of research [e.g. 23, 17, 13].

We can also obtain an improved simulation algorithm if we restrict the gate set. A Clifford circuit is any circuit that takes $|0^n\rangle$ as input, applies a product of the gates

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \qquad S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}, \qquad \mathrm{CZ} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}, \tag{2}$$

and optionally measures a subset of qubits in computational basis. The Gottesman-Knill theorem [16] says that an n-qubit Clifford circuit with m gates can be simulated classically in time $\mathrm{poly}(m, n)$.

In Chapter 1 we will summarize four previously known approaches to proving the Gottesman-Knill theorem. Although each of the four algorithms run in polynomial time, they all incur slightly different costs depending on the number of gates of each type that are applied, the number of measurements, as well as the order in which these operations occur. Consequently, when simulating a given circuit, the overall runtime may depend on which simulation technique is used. In Section 1.3 we improve the algorithm described in [34] by giving a faster subroutine for simulating the application of Hadamard gates and describing a subroutine for measuring multiple qubits simultaneously. We also pay special attention to storing global phase information, which is necessary for the use of stabilizer rank methods [9], discussed in Chapter 3. Chapter 1 concludes with a discussion of the merits of each simulation technique as well as future research directions.

In Chapter 2 we describe a result of [15], the main application of which can be seen as a combination of the idea of restricting a circuit's geometry with the Gottesman-Knill theorem. Given a graph $G = (V, E)$, its corresponding graph state is defined as

$$|G\rangle = \left(\prod_{ij\in E} \mathrm{CZ}_{ij}\right)|+^{|V|}\rangle\,, \tag{3}$$

where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, and CZ_{ij} acts on qubits i, j. We consider two tasks. (For simplicity we only describe a special case of the tasks here.) In both tasks we are given a planar graph G, and for each $v \in V$ we are given a Pauli operator $P_v \in \{X, Y, Z\}$. Let U_v be the local Clifford operator that maps the P_v eigenbasis to the computational basis: If $P_v = X$ then $U_v = H$, if $P_v = Y$ then $U_v = HS^\dagger$, and if $P_v = Z$ then $U_v = I$. The first task is to sample from the output distribution $\Pr[z] = |\langle z|\bigotimes_v U_v|G\rangle|^2$, for $z \in \{0,1\}^n$. In the second task, we are additionally given a fixed string $z \in \{0,1\}^n$ and must compute $|\langle z|\bigotimes_v U_v|G\rangle|^2$. The matrix-multiplication-time version of Clifford simulation described in [15] can be used directly to solve both problems classically in time $O(n^\omega)$, where $\omega \in [2, 2.37286)$ [2] is the matrix multiplication exponent. We give a quadratic speedup by describing $\widetilde{O}(n^{\omega/2})$-time classical algorithms for both tasks.

The results of Chapter 2 can be used to achieve improved algorithms for simulating certain Clifford circuits. For a depth-d Clifford circuit C on n qubits, let G be the graph with vertex set $[n]$ and an edge ij if and only if CZ_{ij} is present in U. Through a reduction to the two tasks described above, we show that when G is planar, a classical algorithm can sample from $C|0^n\rangle$ in time $\widetilde{O}(n^{\omega/2}d^{\omega})$, and given $z \in \{0,1\}^n$ we can compute $|\langle z|U|0^n\rangle|^2$ in the same runtime. If $d = O(\log n)$ then these are again quadratic speedups over directly applying the Clifford circuit simulation algorithm from [15].

In Chapter 3 we discuss stabilizer rank methods [9, 7, 6]. Let $T = \mathrm{diag}(1, e^{i\pi/4})$ and let $|T\rangle = T|+\rangle$. The set of states produced by Clifford circuits, which are known as stabilizer states, span $\mathbb{C}^{2^n}$, and therefore for any $t \geq 1$ there exists $\chi \geq 1$, $\{\alpha_j\}_j \subset \mathbb{C}$, and stabilizer states $\{|\phi_j\rangle\}_j$ such that

$$|T\rangle^{\otimes t} = \sum_{j=1}^{\chi} \alpha_j |\phi_j\rangle . \tag{4}$$

We say that the minimum possible χ, denoted χ_t, is the stabilizer rank of $|T\rangle^{\otimes t}$. Let C be a depth-d circuit built from the universal gate set $\{H, S, \mathrm{CZ}\} \cup \{T\}$, with input state $|0^n\rangle$ and computational basis measurements. Let G be as defined above, and let t be the number of T gates in C. Using a technique described by [9] together with the results of Chapter 2, we show that if G is planar, then we can sample from the output distribution of $C|0^n\rangle$ in time $\widetilde{O}(\chi_t^2 n^{5/2} t^6 d^3)$, and given $z \in \{0,1\}^n$, we can compute $|\langle z|C|0^n\rangle|^2$ in time $\widetilde{O}(\chi_t n^{3/2} t^6 d^3)$. Substituting in the best known upper bound $\chi_t \leq 2^{0.3963t}$ [27] gives runtimes of $\widetilde{O}(2^{0.3963t} n^{3/2} t^6 d^3)$ and $\widetilde{O}(2^{0.7926t} n^{5/2} t^6 d^3)$, respectively. The techniques used in Chapter 3 closely follow previous uses [9, 7, 6] of stabilizer rank methods. For $d = O(\log n)$ previous work [9, 6] allows sampling to be done in time $O(\chi_t n^6 t^6)$ and computation of output probabilities to be done in time $O(\chi_t n^3 t^3)$. For the sampling task our result offers improved scaling in n, but with poorer scaling in χ_t. For the task of computing output probabilities we improve the scaling in n and leave the scaling in χ_t unchanged.

By considering restricted models of quantum computation in this thesis, we hope to shed light on the relationship between the capabilities of quantum and classical computers. Understanding this relationship has theoretical and practical implications, as simulation can be a useful tool for both understanding the source of quantum advantage, as well as verifying the correctness of actual quantum devices. As technology for physically realizing quantum computers continues to mature, working towards a better characterization of this relationship will become increasingly important.

Chapter 1

Clifford circuit simulation

The Gottesman-Knill theorem says that n-qubit Clifford circuits with m gates can be simulated in time poly(m, n) [16]. In this section we review four different simulation algorithms. A *stabilizer state* is any state that can be produced by a Clifford circuit acting on $|0^n\rangle$. Each simulation algorithm relies on having an efficient classical representation of stabilizer states. Given such a representation, each algorithm then provides rules for efficiently updating that representation when a gate H, S, or CZ is applied. To simulate a single-qubit measurement, every algorithm describes an efficient way to determine the probabilities of observing results 0 and 1, respectively. Using these probabilities we can then sample a measurement outcome $r \in \{0, 1\}$. Finally, each algorithm has a rule for efficiently updating its classical representation upon the application of the projection $|r\rangle \langle r|$.

The Gottesman-Knill theorem allows us to accomplish either of the circuit simulation tasks described in the introduction. To sample from the output distribution of a circuit C, we can begin with an efficient classical representation of the input state $|0^n\rangle$, simulate the application of each gate in C one by one to find an efficient representation of $C|0^n\rangle$, and then simulate measurements on qubits one by one. To compute an output probability $|\langle z|C|0^n\rangle|^2$ for fixed $z \in \{0, 1\}^n$, we again produce a representation of $C|0^n\rangle$ and simulate measurements one by one. However, rather than determining the measurement outcome $r \in \{0, 1\}$ by sampling from a distribution, we simply choose $r = z_j$, where j is the qubit being measured. If at any point the measurement is deterministic with result $1 - z_j$, we conclude $|\langle z|C|0^n\rangle|^2 = 0$ and stop. Conveniently, we will see that every single-qubit measurement on stabilizer states is either deterministic or uniformly random. Therefore if we manage to correctly choose the measurement outcome for each qubit, then we will have $|\langle z|C|0^n\rangle|^2 = 2^{-k}$, where k is the number of nondeterministic measurements encountered.

Throughout this thesis we will often use the fact that $X = HS^2H, Z = S^2$, and CNOT $= \sum_{x\in\{0,1\}^2} |x_1, x_1 + x_2\rangle \langle x_1, x_2| = (I \otimes H)\text{CZ}(I \otimes H)$ are all Clifford operators. For $a \in \{0, 1\}^n$ and $j \in [n]$ we write $X(a) = X^{a_1} \otimes \ldots \otimes X^{a_n}$ and $X_j = I \otimes \ldots \otimes I \otimes X \otimes I \otimes \ldots \otimes I$, with the X in the jth position, and define $Z(a), Z_j, H(a), H_j$, etc. similarly. For example, if $n = 3$ and $a = 101$ then $H(a) = H \otimes I \otimes H$ and $X_1Z_2 = X \otimes Z \otimes I$.

1.1 The stabilizer formalism

The generalized Pauli group $\mathcal{P}_n$ is the multiplicative group

$$\mathcal{P}_n = \{i^\gamma X(a)Z(b) : \gamma \in \{0,1,2,3\}, a,b \in \{0,1\}^n\}. \tag{1.1}$$

In the stabilizer formalism [16] we represent a stabilizer state $|\psi\rangle$ as the subgroup

$$\mathrm{Stab}(|\psi\rangle) = \{P \in \mathcal{P}_n : P|\psi\rangle = |\psi\rangle\}. \tag{1.2}$$

This subgroup is called a *stabilizer group*. We say that Paulis $P^{(1)}, \ldots, P^{(m)} \in \mathcal{P}_n$ are *independent* if $P_j \notin \langle P^{(1)}, \ldots, P^{(j-1)}, P^{(j+1)}, \ldots P^{(m)}\rangle$ for each j, where angled brackets denote the subgroup generated by $P^{(1)}, \ldots, P^{(j-1)}, P^{(j+1)}, \ldots P^{(m)}$. Rather than storing every group element, a stabilizer group can always be described by a set of independent generators, called *stabilizer generators*. For example,

$$\mathrm{Stab}(|00\rangle) = \{I, Z_1, Z_2, Z_1Z_2\} \qquad = \langle Z_1, Z_2\rangle \tag{1.3}$$

$$\mathrm{Stab}(|{+}{+}\rangle) = \{I, X_1, X_2, X_1X_2\} \qquad = \langle X_1, X_2\rangle \tag{1.4}$$

$$\mathrm{Stab}\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \{I, X_1X_2, Z_1Z_2, -Y_1Y_2\} = \langle X_1X_2, Z_1Z_2\rangle \tag{1.5}$$

$$\mathrm{Stab}(|0^n\rangle) = \{Z(a) : a \in \{0,1\}^n\} \qquad = \langle Z_1, \ldots, Z_n\rangle. \tag{1.6}$$

What happens to a stabilizer group when we apply a Clifford gate U to $|\psi\rangle$? For any Pauli P,

$$P|\psi\rangle = |\psi\rangle \iff (UPU^\dagger)U|\psi\rangle = U|\psi\rangle. \tag{1.7}$$

Claim 1. Let $P \in \mathcal{P}_n$ and $j,k \in [n]$ be given. Then $H_jPH_j^\dagger$, $S_jPS_j^\dagger$, and $\mathrm{CZ}_{jk}P\mathrm{CZ}_{jk}^\dagger$ are all in $\mathcal{P}_n$ and may be computed in time $O(1)$.

Proof. Let $U \in \{H, S\}$, and let us write $P = i^\gamma X(a)Z(b)$. Then because

$$U_jPU_j^\dagger = i^\gamma X^{a_1}Z^{b_1} \otimes \ldots \otimes X^{a_{j-1}}Z^{b_{j-1}} \otimes UX^{a_j}Z^{b_j}U^\dagger \otimes X^{a_{j+1}}Z^{b_{j+1}} \otimes \ldots \otimes X^{a_n}Z^{a_n}, \tag{1.8}$$

showing that $U_jPU_j^\dagger \in \mathcal{P}_n$ can be done by showing that $UX^{a_j}Z^{b_j}U^\dagger \in \mathcal{P}_1$. Since there are only eight combinations of U and $X^{a_j}Z^{b_j}$, the fact that $UX^{a_j}Z^{b_j}U^\dagger \in \mathcal{P}_1$ can be checked directly. Moreover, computing $U_jPU_j^\dagger$ amounts to replacing the jth tensor element of P with $UX^{a_j}Z^{b_j}U^\dagger$ (and possibly also changing γ). Because there are $O(1)$ combinations of U, and $X^{a_j}Z^{b_j}$, this update can be done in constant time using a lookup table like the one shown in Table 1.1. The case where $U = \mathrm{CZ}_{jk}$ for some $j,k \in [n]$ is very similar. The only difference is that we must instead compute $\mathrm{CZ}_{jk}(X^{a_j}Z^{b_j} \otimes X^{a_k}Z^{b_k})\mathrm{CZ}_{jk}^\dagger$. □

When combined with Eq. 1.7, Claim 1 tells us that for Paulis $P^{(1)}, \ldots, P^{(m)}$,

$$\mathrm{Stab}(|\psi\rangle) = \langle P^{(1)}, \ldots, P^{(m)}\rangle \iff \mathrm{Stab}(U|\psi\rangle) = \langle UP^{(1)}U^\dagger, \ldots, UP^{(m)}U^\dagger\rangle. \tag{1.9}$$

Therefore, given a set of generators for $\mathrm{Stab}(|\psi\rangle)$, to represent the state $U|\psi\rangle$ it suffices to replace each generator P with $P \leftarrow UPU^\dagger$. Because $Z_1, \ldots, Z_n$ are independent and

U	P	$UPU^\dagger$
H	X	Z
	Y	$-Y$
	Z	X
S	X	Y
	Y	$-X$
	Z	Z
CZ_{ab}	X_a	X_aZ_b
	Y_a	Y_aZ_b
	Z_a	Z_a

Table 1.1: Recreated from [26]. Lookup table for updating a stabilizer group after applying the gate $U \in \{H, S, CZ\}$. The value of $UYU^\dagger$ is completely determined by $UXU^\dagger$ and $UZU^\dagger$ but is included for convenience.

generate $\text{Stab}(|0^n\rangle)$, and every stabilizer state is obtained by a series of Clifford gates acting on $|0^n\rangle$, Eq. 1.9 tells us that every stabilizer group requires exactly n independent generators to describe. Because $UPU^\dagger$ can be computed in time $O(1)$, updating the entire set of generators takes time $O(n)$.

For example, suppose we start with the state $|00\rangle$, with $\text{Stab}(|00\rangle) = \langle Z_1, Z_2\rangle$. If we apply the gate H_1, we update our generators by computing $H_1 Z_1 H_1^\dagger = X_1$ (using the third row of Table 1.1) and $H_1 Z_2 H_1^\dagger = Z_2$, to get $\text{Stab}(H_1 |00\rangle) = \langle X_1, Z_2\rangle$. If we then apply a CZ gate, we compute $\text{CZ}X_1\text{CZ}^\dagger = X_1Z_2$ (using the seventh row) and $\text{CZ}Z_2\text{CZ}^\dagger = Z_2$ (using the last row) to get $\text{Stab}(\text{CZ}H_1 |00\rangle) = \langle X_1Z_2, Z_2\rangle$. Notice that the CZ gate changes the generators, but not the overall stabilizer group, as $\langle X_1, Z_2\rangle = \langle X_1Z_2, Z_2\rangle$. This reflects the fact that $\text{CZ} |+\rangle |0\rangle = |+\rangle |0\rangle$.

Stabilizer generators $P^{(1)}, \ldots, P^{(n)}$ are typically stored in a block matrix, known as a *tableau*, of the form

$$\left[\begin{array}{c|c|c} C & D & \delta \end{array}\right], \tag{1.10}$$

where $C, D \in \{0,1\}^{n\times n}$, $\delta \in \{0,1\}^n$, and $P^{(j)} = (-1)^{\delta_j} i^{C_j D_j^T} X(C_j)Z(D_j)$, where C_j and D_j denote the jth rows of C and D. Note that the form $(-1)^{\delta_j} i^{C_j D_j^T} X(C_j)Z(D_j)$ can only represent the Hermitian elements of $\mathcal{P}_n$. This is not a problem, because all elements of $\text{Stab}(|\psi\rangle)$ must be Hermitian. Suppose for contradiction we had some $P \in \text{Stab}(|\psi\rangle)$ that was not Hermitian, we would then have $P^2 |\psi\rangle = P |\psi\rangle = |\psi\rangle$ since $P \in \text{Stab}(|\psi\rangle)$, but also $P^2 |\psi\rangle \neq (PP^\dagger) |\psi\rangle = |\psi\rangle$. Using row vectors to represent the X- and Z-components of Pauli operators also makes it easy to express commutation relations. If $a, a', b, b' \in \{0,1\}^n$, then $X(a)Z(b)$ and $X(a')Z(b')$ commute if and only if

$$\begin{bmatrix} a^T & b^T \end{bmatrix} \Lambda \begin{bmatrix} a' \\ b' \end{bmatrix} = 0 \mod 2, \quad \text{with } \Lambda = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}. \tag{1.11}$$

We can now prove that $\text{Stab}(|\psi\rangle)$ uniquely identifies $|\psi\rangle$ up to a global phase. The fact that global phase information is lost in the stabilizer formalism reflects the fact that if $P |\psi\rangle = |\psi\rangle$, then $P(\alpha |\psi\rangle) = \alpha |\psi\rangle$ for any α.

Claim 2 ([26])**.** Let $P^{(1)}, \ldots, P^{(n)} \in \mathcal{P}_n$ all commute and be independent. If $-I \notin \langle P^{(1)}, \ldots, P^{(n)} \rangle$ then the subspace $\{|\psi\rangle : P^{(j)} |\psi\rangle = |\psi\rangle \ \ \forall j\}$ is one-dimensional.

Before proving the claim we will show how it allows us to prove that a stabilizer group uniquely identifies a stabilizer state up to a global phase. That is, if $\langle P^{(1)}, \ldots, P^{(n)} \rangle = \mathrm{Stab}(|\psi_1\rangle) = \mathrm{Stab}(|\psi_2\rangle)$ for stabilizer states $|\psi_1\rangle, |\psi_2\rangle$, then $|\psi_1\rangle \propto |\psi_2\rangle$. To do this, we just need to argue that for any stabilizer generators $P^{(1)}, \ldots, P^{(n)}$, the premises of Claim 2 are satisfied. Because $Z_1, \ldots, Z_n$ all commute and generate $\mathrm{Stab}(|0^n\rangle)$, and conjugation by a unitary preserves commutation relations, it is true that all stabilizer groups are abelian. Therefore $P^{(1)}, \ldots, P^{(n)}$ all commute. The condition $-I \notin \langle P^{(1)}, \ldots, P^{(n)} \rangle$ will also always be true for stabilizer generators, since $-I |\psi\rangle \neq |\psi\rangle$.

Proof of Claim 2. We start by claiming that if $[\, C \ D \ \delta \,]$ is the tableau representation of $P^{(1)}, \ldots, P^{(n)}$ then $[\, C \ D \,]$ has full rank. Suppose that $s \in \{0,1\}^n$ satisfies

$$\sum_{j=1}^{n} s_j \begin{bmatrix} C_j & D_j \end{bmatrix} = 0. \tag{1.12}$$

Then because a linear combination of the rows of $[\, C \ D \,]$ corresponds to a product of the $P^{(j)}$ with phases being ignored, we then have

$$\prod_{j=1}^{n} (P^{(j)})^{s_j} \in \{\pm I, \pm iI\}. \tag{1.13}$$

Because $-I \notin \langle P^{(1)}, \ldots, P^{(n)} \rangle$, this product must be equal to $+I$. Since the $P^{(j)}$ are independent, we must have $s = 0^n$, which means that $[\, C \ D \,]$ has full rank.

For each j, the projector onto the $+1$ eigenspace of $P^{(j)}$ is given by $(I + P^{(j)})/2$. For each $x \in \{0,1\}^n$, define the projection operator

$$P_x = \prod_{j=1}^{n} \frac{I + (-1)^{x_j} P^{(j)}}{2}. \tag{1.14}$$

Claim 2 says that P_{0^n}, which projects onto the intersection of all $+1$ eigenspaces, has rank 1. Because $\sum_x P_x = I$ has rank 2^n and rank is subadditive, it suffices to show that each P_x has equal rank. We do this by showing $\mathrm{Rank}(P_x) = \mathrm{Rank}(P_{0^n})$ for all x.

Because $[\, C \ D \,]$ has full rank, for each j there exists a solution $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$ to $[\, C \ D \,] \Lambda \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] = e_j$, where e_j is the usual standard basis vector. In other words, there exists $R^{(j)} \coloneqq X(a)Z(b)$ that commutes with $P^{(k)}$ if and only if $j = k$. Then for any x,

$$\left(\prod_{j : x_j = 1} R^{(j)} \right) P_x \left(\prod_{j : x_j = 1} R^{(j)} \right)^{\dagger} = P_{0^n} \left(\prod_{j : x_j = 1} R^{(j)} \right) \left(\prod_{j : x_j = 1} R^{(j)} \right)^{\dagger} = P_{0^n}. \tag{1.15}$$

Since each $R^{(j)}$ is invertible, taking the rank on both sides gives $\mathrm{Rank}(P_{0^n}) = \mathrm{Rank}(P_x)$. $\square$

We will now describe how to perform single-qubit measurements in $O(n^2)$ time, shown in [1], which improves on an earlier algorithm running in time $O(n^\omega)$ [16, 26]. In order to do so, we must also store another set of n Paulis, known as *destabilizer generators*. We will describe how to update destabilizer generators when a gate is applied and when a measurement is performed. If $P^{(1)}, \ldots, P^{(n)}$ are the stabilizer generators and $Q^{(1)}, \ldots, Q^{(n)}$ the destabilizer generators of a state, then we will also show that the following properties are preserved by our update rules whenever we apply a gate or perform a measurement:

(D1) The set $\{P^{(1)}, Q^{(1)}, \ldots, P^{(n)}, Q^{(n)}\}$ is independent.

(D2) For each j, the operator $Q^{(j)}$ is Hermitian.

(D3) For each j, k, the operators $Q^{(j)}$ and $Q^{(k)}$ commute.

(D4) For each j, k, if $j \neq k$ then $P^{(j)}$ and $Q^{(k)}$ commute, and if $j = k$ then $P^{(j)}$ and $Q^{(k)}$ anticommute.

For the state $|0^n\rangle$, we may take $P^{(j)} = Z_j$ and $Q^{(j)} = X_j$ for each j. To update destabilizer generators after applying a gate U to $|\psi\rangle$, we use the exact same rules as with the stabilizer generators: Using Table 1.1, we set $Q^{(j)} \leftarrow UQ^{(j)}U^\dagger$ for each j. Since there are n destabilizer generators, this takes time $O(n)$ in total. Notice that properties (D1) – (D4) are preserved: Conjugation by a unitary preserves independence, Hermiticity, and commutation relations.

Given that the $Q^{(j)}$ are Hermitian, we can also augment the tableau from Eq. 1.10 to form

$$\left[\begin{array}{c|c|c} A & B & \gamma \\ \hline C & D & \delta \end{array}\right], \tag{1.16}$$

where $Q^{(j)} = (-1)^{\gamma_j} i^{A_j B_j^T} X(A_j) Z(B_j)$.

We can now describe the measurement subroutine. Suppose we wish to measure a single qubit ℓ in the computational basis.

Case 1: Suppose first that $Z_\ell P^{(j)} = -P^{(j)} Z_\ell$ for some stabilizer generator $P^{(j)}$. If $P^{(j)}$ is not unique we can replace all other anticommuting generators with $P^{(k)} \leftarrow P^{(j)} P^{(k)}$ in time $O(n^2)$. We will also replace every anticommuting destabilizer generator with $Q^{(k)} \leftarrow P^{(j)} Q^{(k)}$. A straightforward calculation shows that this preserves properties (D1) – (D4).

Both measurement outcomes can then be seen to occur with probability $\frac{1}{2}$:

$$|\langle\psi| \frac{I + Z_\ell}{2} |\psi\rangle|^2 = |\langle\psi| \frac{I + Z_\ell}{2} P^{(j)} |\psi\rangle|^2 \tag{1.17}$$

$$= |\langle\psi| P^{(j)} \frac{I - Z_\ell}{2} |\psi\rangle|^2 \tag{1.18}$$

$$= |\langle\psi| \frac{I - Z_\ell}{2} |\psi\rangle|^2. \tag{1.19}$$

After flipping a coin to determine the measurement result r we update our tableau by replacing $Q^{(j)} \leftarrow P^{(j)}$ and $P^{(j)} \leftarrow (-1)^r Z_\ell$. The time complexity in this case is dominated by the cost of updating the anticommuting generators, which takes time $O(n^2)$.

Next we show that the updated set of stabilizer generators do in fact stabilize the post-measurement state. For each $P(k)$ with $k \neq j$, we have

$$P^{(k)} \frac{I + (-1)^r Z_\ell}{2} |\psi\rangle = \frac{I + (-1)^r Z_\ell}{2} P^{(k)} |\psi\rangle = \frac{I + (-1)^r Z_\ell}{2} |\psi\rangle . \tag{1.20}$$

Since $(-1)^r Z_\ell$ anticommutes with $P^{(k)}$ for each $k \neq j$, and stabilizer groups are abelian, $(-1)^r Z_\ell$ could not have been in the pre-measurement stabilizer group. Therefore the updated set of stabilizer generators are independent. They also cannot generate $-I$ (since we would then have $-I |\psi\rangle = |\psi\rangle$), so by Claim 2, they do in fact generate $\mathrm{Stab}((I + (-1)^r Z) |\psi\rangle)$.

We will now show that (D1) is preserved. Showing that (D2) – (D4) are preserved is straightforward. By induction, $\{P^{(1)}, Q^{(1)}, \ldots, P^{(n)}, Q^{(n)}\} \setminus \{Q^{(j)}\}$ is independent. By (D4) together with the fact that stabilizer groups are abelian, every element of this set commutes with $P^{(j)}$. But since $(-1)^r Z_\ell$ anticommutes with $P^{(j)}$, it cannot be generated by this set. Therefore the updated set of stabilizer and destabilizer generators are independent, so (D1) is preserved.

Case 2: Suppose instead that $Z_\ell P^{(j)} = P^{(j)} Z_\ell$ for all j. Therefore for each j we have $P^{(j)} Z_\ell |\psi\rangle = Z_\ell P^{(j)} |\psi\rangle = Z_\ell |\psi\rangle$. By Claim 2, together with the fact that Z has eigenvalues ± 1, we then have $Z_\ell |\psi\rangle = (-1)^r |\psi\rangle$ for some $r \in \{0, 1\}$. This tells us that the measurement is deterministic with result r. We just need to determine what r is.

Let us ignore phases for a moment. Suppose we could find a solution $s \in \{0, 1\}^n$ to

$$Z_\ell \propto \prod_{j=1}^{n} (P^{(j)})^{s_j}, \tag{1.21}$$

where the $\propto$ sign hides a power of i. Note that a solution always exists, because $(-1) Z_\ell \in \mathrm{Stab}(|\psi\rangle)$ and $P^{(1)}, \ldots, P^{(n)}$ generate $\mathrm{Stab}(|\psi\rangle)$. A solution can be found by solving the system of linear equations

$$\begin{bmatrix} C \\ D \end{bmatrix} s = \begin{bmatrix} 0 \\ e_\ell \end{bmatrix} . \tag{1.22}$$

Given s we could determine r by explicitly computing the right hand side of Eq. 1.21 and checking its sign. This system can be solved in time $O(n^3)$ using Gaussian elimination or $O(n^\omega)$ using fast matrix multiplication (see Appendix A). Aaronson and Gottesman [1], however, describe an alternative way of finding r in time $O(n^2)$ without resorting to linear algebra.

Recall that for $a, a', b, b' \in \{0, 1\}^n$, the Paulis $X(a)Z(b)$ and $X(a')Z(b')$ commute if and only if $\begin{bmatrix} a^T & b^T \end{bmatrix} \Lambda \begin{bmatrix} a' \\ b' \end{bmatrix} = 0 \mod 2$. Modulo 2, we then have

$$\begin{aligned}
s_j &= s_j \begin{bmatrix} A_j & B_j \end{bmatrix} \Lambda \begin{bmatrix} C_j & D_j \end{bmatrix}^T && (1.23)\\
&= \sum_{k=1}^{n} s_k \begin{bmatrix} A_j & B_j \end{bmatrix} \Lambda \begin{bmatrix} C_k & D_k \end{bmatrix}^T && (1.24)\\
&= \begin{bmatrix} A_j & B_j \end{bmatrix} \Lambda \sum_{k=1}^{n} s_k \begin{bmatrix} C_k & D_k \end{bmatrix}^T && (1.25)\\
&= \begin{bmatrix} A_j & B_j \end{bmatrix} \Lambda \begin{bmatrix} 0 & e_\ell \end{bmatrix}^T && (1.26)\\
&= A_{j\ell}. && (1.27)
\end{aligned}$$

In the first equality, we have used (D4) to multiply s_j by one. In the second equality, we have also used (D4), which tells us that the only nonzero summand is the one where $k = j$. The final equality tells us that s turns out to just be the ℓth column of A. We can now check which of $\pm Z_\ell$ is in the stabilizer group by calculating the right hand side of Eq. 1.21. Doing so takes time $O(n^2)$, and dominates the time complexity of measurement in this case. Since the state is unchanged, no generators are updated, so properties (D1) – (D4) are preserved.

This completes the first proof of the Gottesman-Knill theorem.

Tableau representations can also be used to describe Clifford operators. If U is a Clifford operator, then we can represent U by storing a tableau containing $UX_1U^\dagger, \ldots, UX_nU^\dagger$ and $UZ_1U^\dagger, \ldots, UZ_nU^\dagger$. Given a stabilizer generator P for a state $|\psi\rangle$, one could then compute $UPU^\dagger$ by using the tableau for U. For example, if $P = X_1Z_2Z_3$, we can compute $UPU^\dagger = (UX_1U^\dagger)(UZ_2U^\dagger)(UZ_3U^\dagger)$, where each term may be looked up in the tableau for U. The tableau for U is equivalently the tableau for the state $U|0^n\rangle$.

This idea can be used to frame all operations in the stabilizer formalism in terms of linear algebra. It has been shown [15] that using this framework, any collection of CZ gates can be applied in time $O(n^\omega)$ and measuring k qubits can be done in time $O(n^2k^{\omega-2})$. Using the techniques of this section alone, these operations would take time $O(n^3)$ and $O(kn^2)$ in the worst case.

1.2 CH form

CH form [6] is an adaptation of tableau-based simulation which also stores global phase. Clifford simulators that store global phase information are said to be *phase-sensitive*. To simulate a Clifford circuit, phase-sensitivity is unnecessary, however stabilizer rank methods [9, 6], discussed in Chapter 3, allow universal circuits to be simulated using a phase-sensitive Clifford simulator. In CH form, stabilizer states are represented as

$$|\psi\rangle = e^{wi\pi/4}U_CH(r)|s\rangle, \tag{1.28}$$

where $w \in \{0, 1, \ldots, 7\}$, U_C is a Clifford operator with $U_C|0^n\rangle = |0^n\rangle$, and $r, s \in \{0, 1\}^n$. To store U_C, we store the tableau for $U_C^\dagger$. We call operators that map $|0^n\rangle$ to $|0^n\rangle$, as is the case with U_C, *C-type* or *control-type* operators. The tableau representation of a Clifford operator only identifies that operator up to a global phase, so we must also require $U_C|0^n\rangle = |0^n\rangle$ in order to fix its global phase.

To apply an S gate to qubit j, notice that S_jU_C is a control-type operator. Therefore to apply S_j, it suffices to update $U_C \leftarrow S_jU_C$. The tableau for $U_C^\dagger$ stores $U_C^\dagger X_iU_C$ and $U_C^\dagger Z_iU_C$ for $i \in [n]$. The tableau for $(S_jU_C)^\dagger$ must store $U_C^\dagger S_j^\dagger X_iS_jU_C$ and $U_C^\dagger S_j^\dagger Z_iS_jU_C$. However, for all i, we have $U_C^\dagger S_j^\dagger Z_iS_jU_C = U_C^\dagger Z_iU_C$, and for $i \neq j$ we have $U_C^\dagger S_j^\dagger X_iS_jU_C = U_C^\dagger X_iU_C$, so all that's needed is to compute $U_C^\dagger S_j^\dagger X_jS_jU_C$. To do so, we use the fact that $S^\dagger XS = -Y = -iXZ$. Then

$$U_C^\dagger S_j^\dagger X_jS_jU_C = -iU_C^\dagger X_jZ_jU_C = -i(U_C^\dagger X_jU_C)(U_C^\dagger Z_jU_C). \tag{1.29}$$

We can then use the original tableau for $U_C^\dagger$ to look up $(U_C^\dagger X_jU_C)$ and $(U_C^\dagger Z_jU_C)$. This process takes time $O(1)$. The same idea can also be used to apply the control-type operators CZ and CNOT in time $O(1)$.

Claim 3. In CH form, a Hadamard gate can be applied in time $O(n^2)$.

Proof. Using the decomposition $H = (X+Z)/\sqrt{2}$ we have

$$H_j |\psi\rangle = e^{wi\pi/4} U_C H(r) \left(\frac{P|s\rangle + Q|s\rangle}{\sqrt{2}} \right) \tag{1.30}$$

$$= e^{w'i\pi/4} U_C H(r) \left(\frac{|t\rangle + i^\alpha |t'\rangle}{\sqrt{2}} \right), \tag{1.31}$$

where $P = H(r)U_C^\dagger X_j U_C H(r)$ and $Q = H(r)U_C^\dagger Z_j U_C H(r)$ are both in $\mathcal{P}_n$, and w', α, t, t' are determined by $P|s\rangle, Q|s\rangle$. To compute P and Q, we first use the tableau for $U_C^\dagger$ to look up $U_C^\dagger X_j U_C$ and $U_C^\dagger Z_j U_C$. From there, conjugating by $H(r)$ can be done in time $O(n)$. Computing w', α, t, t' can also be done in time $O(n)$.

If $t = t'$ then $\alpha \in \{1, 3\}$, since otherwise we would have $||H_j|\psi\rangle|| \neq 1$. In this case, referring to Eq. 1.31, we update $s \leftarrow t$, and $w \leftarrow w' + 1 \mod 8$ if $\alpha = 1$ or $w \leftarrow w' - 1 \mod 8$ if $\alpha = 3$. If $t \neq t'$ then we will rewrite the product $H(r)(|t\rangle + i^\alpha |t'\rangle)$ appearing in Eq. 1.31. First, choose any $q \in [n]$ with $t_q \neq t'_q$ and let

$$V_C = H(r) \left(\prod_{\substack{j : t_j \neq t'_j \\ j \neq q}} \mathrm{CNOT}_{qj} \right) H(r). \tag{1.32}$$

Using the identity $\mathrm{CZ} = (I \otimes H)\mathrm{CNOT}(I \otimes H)$, we can see that V_C is a product of CNOT gates (when $r_j = 0$) and CZ gates (when $r_j = 1$), and is therefore a C-type operator. Then we have

$$H(r)(|t\rangle + i^\alpha |t'\rangle) = V_C \left[H(r_{[n]\setminus q}) |t_{[n]\setminus q}\rangle \otimes H(r_q)(|t_q\rangle + i^\alpha |1 + t_q\rangle) \right] \tag{1.33}$$

$$= \sqrt{2} e^{vi\pi/4} V_C \left[H(r_{[n]\setminus q}) |t_{[n]\setminus q}\rangle \otimes S_q^a H_q^b |c\rangle \right] \tag{1.34}$$

for some $a \in \{0, 1, 2, 3\}, b, c \in \{0, 1\}$, and $v \in \{0, 1, \ldots, 7\}$. Substituting this into Eq. 1.31 gives us

$$H_j |\psi\rangle = e^{(w'+v)i\pi/4} U_C V_C \left[H(r_{[n]\setminus q}) |t_{[n]\setminus q}\rangle \otimes S_q^a H_q^b |c\rangle \right]. \tag{1.35}$$

We can now simplify this expression to put it into CH form. Let us first replace $U_C \leftarrow U_C V_C S_q^a$, which is a C-type operator. We will perform this update by writing $V_C S_q^a$ as a product of $O(n)$ CZ and CNOT gates, followed by the S_q^a gate, and update the tableau for $U_C^\dagger$ one gate at a time. We will show how to perform the update for a CZ gate (the CNOT and S gates are similar). Given the tableau for $U_C^\dagger$, which stores products like $U_C^\dagger X_i U_C$ (and similarly for Z_i), we'd like to compute $(U_C \mathrm{CZ})^\dagger X_i (U_C \mathrm{CZ}) = \mathrm{CZ}^\dagger (U_C^\dagger X_i U_C) \mathrm{CZ}$. Using the original tableau for U_C, we look up $(U_C^\dagger X_i U_C)$, and then we simply conjugate by CZ as we did in Section 1.1 in time $O(1)$.[1] To update all $2n$ generators we need time $O(n)$ per gate, leading to the total runtime of $O(n^2)$. After updating $U_C \leftarrow U_C V_C S_q^a$ in time $O(n^2)$, we perform the updates $w \leftarrow w' + v$, $r_q \leftarrow r_q + b$, $s_{[n]\setminus q} \leftarrow t_{[n]\setminus q}$, and $s_q \leftarrow c$ in time $O(n)$. □

[1] To apply the final gate S_q^a, we will need to find $(S_q^a)^\dagger (U_C^\dagger X_i U_C) S_q^a$. However, Table 1.1 only stores information about conjugation by S^1, not by $(S^a)^\dagger$. This can be handled by using the fact that $(S^a)^\dagger = S^{3a}$, and then repeatedly conjugating by S using Table 1.1.

A similar trick can be used to measure a qubit j in the computational basis in time $O(n^2)$. First we choose a random $z \in \{0,1\}$. Then

$$\frac{I + (-1)^z Z_j}{2} e^{wi\pi/4} U_C H(r) |s\rangle = e^{wi\pi/4} U_C H(r) \frac{|s\rangle + Q|s\rangle}{2}, \tag{1.36}$$

where $Q = (-1)^z H(r) U_C^\dagger Z_j U_C H(r)$ can be computed in time $O(n)$ as in the Hadamard gate case. If $Q|s\rangle = |s\rangle$ then the state is unchanged, so the measurement is deterministic with result z. If $Q|s\rangle = -|s\rangle$ then the right hand side of Eq. 1.36 is zero, so the measurement must have been deterministic with result $1-z$. If neither of these are the case, then the measurement is random with result z. To represent the post-measurement state, write $Q = i^\gamma X(a)Z(b)$. To renormalize the state we multiply Eq. 1.36 by $\sqrt{2}$, since z is observed with probability $\frac{1}{2}$. Then taking $w' \leftarrow w, t \leftarrow s, t' \leftarrow s + a$, and $\alpha \leftarrow \gamma + \sum_j b_j s_j$ puts Eq. 1.36 into the form of Eq. 1.31. We can then proceed as in the proof of Claim 3.

1.3 Affine form

Tableau representation and CH form are both somewhat removed from the usual computational basis representation of a state. In this section we describe a simulation technique that tracks the computational basis representation of a stabilizer state [12, 34]. We will present a slightly modified version of the algorithm described by Van den Nest [34]. These modifications allow global phase to be stored, Hadamard gates to be applied in time $O(n^2)$, and k single-qubit measurements to be performed in time $O(n^2 k^{\omega-2})$.

The *affine form* of an n-qubit stabilizer state $|\psi\rangle$ consists of an affine space $\mathbb{A} \subseteq \{0,1\}^n$ of dimension m, linear function $\ell : \{0,1\}^n \to \{0,1,2,3\}$, quadratic function $q : \{0,1\}^n \to \{0,1\}$, and scalar $p \in \{e^{0i\pi/4}, e^{1i\pi/4}, \ldots, e^{7i\pi/4}\}$, such that

$$|\psi\rangle = \frac{p}{\sqrt{2^m}} \sum_{x \in \mathbb{A}} i^{\ell(x)} (-1)^{q(x)} |x\rangle . \tag{1.37}$$

We will see that, roughly speaking, ℓ and q encode information about S and CZ gates that have been applied, since for all $j,k \in [n]$ we have $S_j|x\rangle = i^{x_j}|x\rangle$ and $\mathrm{CZ}_{jk}|x\rangle = (-1)^{x_j x_k}|x\rangle$. The affine space $\mathbb{A}$ will be expressed as $\{Ru + t : u \in \{0,1\}^m\}$, where R is a full rank $n \times m$ binary matrix. We require that R is kept in reduced column echelon form (RCEF),[2] up to a permutation of rows or columns. Throughout this section, when we say "finding an affine space," we will always mean finding a representation of that space in which the generating matrix R has this form.

A few observations will be useful when handling R:

- Permuting columns of R has no effect on $\mathbb{A}$.
- If we swap row a with row b of R, swap t_a with t_b, and swap x_a with x_b whenever they appear in $\ell(x), q(x)$, then we have effectively swapped the indices of qubits a and b of $|\psi\rangle$. This can be done in time $O(n)$.

[2] A matrix R is in RCEF if R^T is in reduced row echelon form.

- Adding column a of R to column b when $a \neq b$ has no effect on $\mathbb{A}$. More generally, replacing $R \leftarrow RW$ when W is an invertible binary matrix has no effect on $\mathbb{A}$.

In light of the first two facts we will often assume that the rows or columns of R are permuted however we like. Any permutation on the n rows of R can be written as a product of $O(n)$ transpositions, which, by the second observation, can be performed in time $O(n^2)$. We will only need to permute rows in the Hadamard and measurement subroutines. Because these take time $O(n^2)$ and $O(n^2 k^{\omega-2})$ anyway, the extra cost of $O(n^2)$ can be ignored.

Because R is in RCEF up to a permutation of rows and columns, for the remainder of this section we will assume that we have gone ahead and performed these row and column permutations, and that R takes the form $R = \left[\begin{smallmatrix} I \\ * \end{smallmatrix}\right]$, where I is the $m \times m$ identity matrix and $*$ is an arbitrary $(n-m) \times m$ matrix. We will also sometimes refer to *pivot rows* of a matrix. If a matrix is in RCEF up to a permutation of rows and columns then a pivot row is a row that contains a leading one. For example, the following matrix is in RCEF up to a permutation of rows and columns, and its pivot rows are rows $2, 4$, and 5:

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.38}$$

We now describe the simulation algorithm. To represent the state $|0^n\rangle$ we use the convention that R is the $n \times 0$ matrix, and set $t = 0^n$, $p = 1$, and $\ell(x) = q(x) \equiv 0$. Applying S_a and CZ_{ab} for $a, b \in [n]$ can both be done in $O(1)$ time. For a computational basis state $|x\rangle$ we have $S_a |x\rangle = i^{x_a} |x\rangle$, so we perform the update $\ell(x) \leftarrow \ell(x) + x_a \mod 4$ to apply S_a. Similarly, we have $\mathrm{CZ}_{ab} |x\rangle = (-1)^{x_a x_b} |x\rangle$, so we update $q(x) \leftarrow q(x) + x_a x_b \mod 2$ to apply CZ_{ab}.

Hadamard gates are more costly. We will show that single Hadamard gate can be applied in time $O(n^2)$. Before doing this, note that the naïve way to apply an X gate is to decompose it as $X = HS^2H$, which would take time $O(n^2)$ to simulate. A shortcut is to instead perform the update $t_a \leftarrow t_a + 1 \mod 2$. Since we also have $Z = S^2$, we can apply any local Pauli in constant time without having to resort to Hadamard gates.

We'll now show to apply a single Hadamard gate H_h to a state $|\psi\rangle$ represented in affine form. We consider the cases $h > m$ (i.e. row h is not a pivot row) and $h \leq m$ (row h is a pivot row) separately.

Case 1: Non-pivot rows. We will assume for simpler notation that $h = n$. If this is not the case we can just permute rows h and n of R. Because row n, which we'll denote R_n is not a pivot row, we can express x_n in terms of $x_1, \ldots, x_m$. Because $R_1, \ldots, R_m$ *are* pivot rows, we have $x_j = u_j + t_j$ for $j = 1, \ldots, m$. Substituting this into $x_n = R_n u + t_n$ gives us

$$x_n = R_n \begin{bmatrix} x_1 & \ldots & x_m \end{bmatrix}^T - R_n \begin{bmatrix} t_1 & \ldots & t_m \end{bmatrix}^T + t_n. \tag{1.39}$$

Using the identity $H |a\rangle = \frac{1}{\sqrt{2}} \sum_{b=0}^{1} (-1)^{ab} |b\rangle$ we have

$$H_n |\psi\rangle = \frac{p}{\sqrt{2^{m+1}}} \sum_{v \in \{0,1\}} \sum_{x \in \mathbb{A}} i^{\ell(x)} (-1)^{q(x) + v x_n} |x_1 \ldots x_{n-1} v\rangle. \tag{1.40}$$

Strings of the form $x_1 \dots x_{n-1} v$ form an affine space $\mathbb{B} = \{\left[\begin{smallmatrix} \tilde{R} & 0 \\ 0 & 1 \end{smallmatrix}\right] u + \left[\begin{smallmatrix} \tilde{t} \\ 0 \end{smallmatrix}\right] : u \in \{0,1\}^m\}$, where tildes denote the first $n-1$ rows of R and t. Notice that $\left[\begin{smallmatrix} \tilde{R} & 0 \\ 0 & 1 \end{smallmatrix}\right]$ is in RCEF (up to a permutation of rows). The only problem is that $\ell(x)$ and $q(x) + vx_n$ still include x_n as a variable.

To remedy this issue, we substitute the expression for x_n from Eq. 1.39 into $\ell(x)$ and $q(x) + vx_n$.[3] This takes time $O(n^2)$. Let us call these new functions $q'(x_1 \dots x_{n-1} v)$ and $\ell'(x_1 \dots x_{n-1} v)$. We now have the affine form expression

$$H_n |\psi\rangle = \frac{p}{\sqrt{2^{m+1}}} \sum_{x_1 \dots x_{n-1} v \in \mathbb{B}} i^{\ell'(x_1 \dots x_{n-1} v)} (-1)^{q'(x_1 \dots x_{n-1} v)} |x_1 \dots x_{n-1} v\rangle . \tag{1.41}$$

Case 2: Pivot rows. We will assume that $h = 1$, since otherwise we just permute rows 1 and h and columns 1 and h. Recall that $R_1 = [1, 0, \dots, 0]$. Because R_1 is a pivot row, it will only be possible to write x_1 in terms of the other x_j if there exists some other R_i with $R_{i1} = 1$. This can be checked in time $O(n)$. We then proceed in two sub-subcases.

Suppose first that such an i exists. We will then add column 1 to column j for each j with $R_{ij} = 1$. This will cause R_i to be a pivot row, and R_1 to become a non-pivot row. Since $R_{21} = \dots = R_{m1} = 0$, rows $R_2, \dots, R_m$ will remain pivot rows. Here is an example with $i = 5$.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{1.42}$$

This takes time $O(n^2)$ and transforms R_1 into a non-pivot row. We can then proceed as in Case 1.

The difficult sub-subcase is when no such i exists. After applying H_1 we have

$$H_1 |\psi\rangle = \frac{p}{\sqrt{2^{m+1}}} \sum_{v \in \{0,1\}} \sum_{x \in \mathbb{A}} i^{\ell(x)} (-1)^{q(x) + vx_1} |v\bar{x}\rangle , \tag{1.43}$$

where $\bar{x} = x_2 \dots x_n$. Strings of the form $v\bar{x}$ form an affine space $\mathbb{B} = \{Ru + \left[\begin{smallmatrix} 0 \\ t \end{smallmatrix}\right] : u \in \{0,1\}^m\}$. The amplitude on $|v\bar{x}\rangle$ in Eq. 1.43 is given by

$$\frac{p}{\sqrt{2^{m+1}}} \sum_{x_1=0}^{1} i^{\ell(x)} (-1)^{q(x) + vx_1} . \tag{1.44}$$

The following lemma, proved in Appendix A, can be used to rewrite this sum as a single term. We give as input to the lemma $N = n+1$, $f(vx) = \ell(x) + 0v$, and $g(vx) = q(x) + x_1 v$.

[3] This may result in some linear terms appearing in q and some constant terms appearing in ℓ. This is not a problem, as those terms can just be transferred to ℓ and p, respectively.

Lemma 4. *Let $N \geq 1$, let $f : \{0,1\}^N \to \{0,1,2,3\}$ be a linear function, let $g : \{0,1\}^N \to \{0,1\}$ be quadratic, and for $y \in \{0,1\}^N$ write $\bar{y} = y_2 \dots y_N$. An $O(N^2)$-time algorithm can find a linear function $f' : \{0,1\}^{N-1} \to \{0,1,2,3\}$, affine function $f'' : \{0,1\}^{N-1} \to \{0,1\}$, quadratic function $g' : \{0,1\}^{N-1} \to \{0,1\}$, and scalar $\alpha \in \mathbb{C}$ such that*

$$\sum_{y_1=0}^{1} i^{f(y)}(-1)^{g(y)} = \alpha i^{f'(\bar{y})}(-1)^{g'(\bar{y})}\delta_{f''(\bar{y}),0}. \tag{1.45}$$

After applying the lemma and writing $y = v\bar{x}$, we have

$$H_1 |\psi\rangle = \frac{p\alpha}{\sqrt{2^{m+1}}} \sum_{\substack{y\in\mathbb{B}: \\ f''(y)=0}} i^{f'(y)}(-1)^{g'(y)} |y\rangle . \tag{1.46}$$

All that remains is to show that $\{y \in \mathbb{B} : f''(y) = 0\}$ can be computed in time $O(n^2)$. To do so we use the following lemma, proved in Appendix A.

Lemma 5. *Let $n > 1$, let $\mathbb{B} = \{Ru + t' : u \in \{0,1\}^m\} \subseteq \{0,1\}^n$ be an affine space with R in RCEF, and let $f'' : \{0,1\}^n \to \{0,1\}$ be an affine function. Then $\{y \in \mathbb{B} : f(y) = 0\}$ can be found in time $O(n^2)$.*

We have now shown that H_h can always be applied in time $O(n^2)$. To complete the proof of the Gottesman-Knill theorem we will describe how to perform measurements on single qubits.

There are three cases to consider when performing measurements. Consider an arbitrary qubit j.

- If $R_j = 0$ then measurement on qubit j is deterministic with result t_j.
- If R_j is a pivot row the measurement result is random and determined entirely by u_j. We therefore choose a random $u_j \in \{0,1\}$ and set our measurement result to $u_j + t_j$. We then fix $x_j = u_j + t_j$ and update ℓ, q, and p accordingly. To update t, we add u_j times the jth column of R and then remove the jth column. As with the measurement subroutines in the previous two sections, we can just as easily choose the value of u_j however we like, rather than choosing randomly. This clearly has no effect on the total runtime.
- If the qubit to be measured corresponds to a nonzero, non-pivot row of R, then we can perform column reduction on R so that that row becomes a pivot row, similarly to what is shown in Eq. 1.42, and then proceed as in the second case.

The first two cases take time $O(n)$, and the third takes time $O(n^2)$.

We now give a subroutine for measuring k qubit simultaneously in time $O(n^2k^{\omega-2})$, an improvement over performing measurements one at a time, which would take time $O(n^2k)$. Because the first two cases take time $O(n)$, it suffices to consider a batched version of the third case, in which all qubits correspond to nonzero, non-pivot rows of R. This case requires the following technical lemma, proved in Appendix A.

Lemma 6. *Let A be an $n \times m$ matrix in RCEF and let $\Gamma \subseteq [n]$. There exists an invertible $m \times m$ matrix W such that AW is in RCEF and can be computed in time $O(|\Gamma|^{\omega-2}n^2)$ and the set $\{(AW)_j : j \in \Gamma \text{ and } (AW)_j \text{ is a pivot row}\}$ is maximal over all possible W.*

Using Lemma 6 with $A = R$ and Γ equal to the set of qubits to be measured, we get a new matrix $R' = RW$. For each $j \in \Gamma$ for which R'_j is a pivot row, we can proceed as already described. Because the set $\{R'_j : j \in \Gamma \text{ and } R'_j \text{ is a pivot row}\}$ is maximal, once the pivot rows have been measured, the remaining rows R'_i for which i is not a pivot row will be identically zero. We can then proceed as already described.

Finally, we mention that if several CNOT gates must be applied successively, we can do better than using the decomposition $\text{CNOT} = (I \otimes H)\text{CZ}(I \otimes H)$. To apply CNOT_{ab}, we replace $R_b \leftarrow R_a + R_b$ and $t_b \leftarrow t_a + t_b$. We will do this for each of CNOT gates to be applied. Unfortunately, this may bring R out of RCEF. To bring R back into RCEF in time $O(n^\omega)$ we apply Lemma 6 with $\Gamma = [n]$. Replacements of the form $R_b \leftarrow R_a + R_b$ and $t_b \leftarrow t_a + t_b$ can be accomplished by multiplying R and t on the left by some $m \times n$ matrix, which takes time $O(n^\omega)$. Therefore, an arbitrary layer of CNOT gates can be applied in time $O(n^\omega)$.

1.4 The graph state formalism

Given any graph $G = (V, E)$ we can define its graph state as

$$|G\rangle = \left(\prod_{ij \in E} \text{CZ}_{ij}\right) |+^{|V|}\rangle . \tag{1.47}$$

It is known [e.g. 3, 18] that for any stabilizer state $|\psi\rangle$, there exists a graph G and local Clifford operators $U_1, \ldots, U_n$ such that

$$|\psi\rangle = (U_1 \otimes \ldots \otimes U_n) |G\rangle . \tag{1.48}$$

In this section we describe the results of [18, 3] to show how to perform Clifford simulation using the representation of Eq. 1.48. The starting state $|0^n\rangle$ has $G = ([n], \varnothing)$ and $U_j = H$ for each j. Applying a local operator W to a qubit j can be done in time $O(1)$ by replacing $U_j \leftarrow WU_j$.

To apply CZ gates we need to make use of the fact that graph state formalism representations need not be unique. Equivalent representations can be characterized using the *local complementation* operation on graphs [11, 35]. For $a \in V(G)$, let $N(a) = \{b \in V(G) : ab \in E(G)\}$ be its neighbourhood, and for two sets A, B let $A \Delta B = (A \setminus B) \cup (B \setminus A)$ denote their symmetric difference. Local complementation at a is an operation $L_a : (V, E) \mapsto (V, E')$, with

$$E' = E \Delta \{bc : b, c \in N(a), b \neq c\}. \tag{1.49}$$

Fig. 1.1 shows an example. In words, local complementation maps the subgraph induced by $N(a)$ to its complement and leaves the rest of G unchanged. We will often make use of the fact that if Q is an adjacency matrix, then L_a maps Q_{ij} to $Q_{ij} + Q_{ai}Q_{aj} \mod 2$. Updating Q in this way takes time $O(n^2)$.

Lemma 7. *For any graph G and vertex a,*

$$|G\rangle = e^{i\pi/4} S_a^3 H_a S_a^3 \prod_{b \in N(a)} S_b^3 \, |L_a(G)\rangle \,. \tag{1.50}$$

We will use this lemma as a way to alter certain local operators without changing $|\psi\rangle$. For example, if we want to replace $U_a \leftarrow U_a S_a^3 H_a S_a^3$, we can rewrite $|\psi\rangle$ as

$$|\psi\rangle = (U_1 \otimes \ldots \otimes U_n) \, |G\rangle = e^{i\pi/4} (U_1 \otimes \ldots \otimes U_n) S_a^3 H_a S_a^3 \prod_{b \in N(a)} S_b^3 \, |L_a(G)\rangle \,. \tag{1.51}$$

Proof. We will prove the equivalent statement

$$S_a H_a S_a \prod_{b \in N(a)} S_b \, |G\rangle = e^{i\pi/4} \, |L_a(G)\rangle \,. \tag{1.52}$$

Let Q be the upper triangular part of the adjacency matrix of G. Then

$$|G\rangle = \sum_{x \in \{0,1\}^n} (-1)^{x^T Q x} \, |x\rangle \,. \tag{1.53}$$

To simplify notation we take $a = 1$. By viewing Eq. 1.53 as an affine form representation of $|G\rangle$, we use the update rules from Section 1.3 to find

$$H_1 S_1 \, |G\rangle = e^{i\pi/4} \sum_{x \in \{0,1\}^n} i^{\ell(x)} (-1)^{q(x)} \, |x\rangle \,, \text{ where} \tag{1.54}$$

$$\ell(x) = 3x_1 + 3 \sum_{j>1} Q_{1j} x_j, \tag{1.55}$$

$$q(x) = \sum_{k>j>1} (Q_{jk} + Q_{1j} Q_{1k}) x_j x_k + \sum_{j>1} Q_{1j} x_1 x_j. \tag{1.56}$$

The function $q(x)$ is equal to $x^T L_a(Q) x$, where $L_a(Q)$ is the upper triangular part of the adjacency matrix for $L_a(G)$. Using the fact that $S_j \, |x\rangle = i^{x_j} \, |x\rangle$, we see that applying the remaining S operators $S_1 \prod_{b \in N(1)} S_b$ eliminates $\ell(x)$, leaving us with

$$S_a H_a S_a \prod_{b \in N(a)} S_b \, |G\rangle = e^{i\pi/4} \sum_{x \in \{0,1\}^n} (-1)^{x^T L_a(Q) x} \, |x\rangle = e^{i\pi/4} \, |L_a(G)\rangle \,. \tag{1.57}$$

□

We now show how for $a, b \in V$, the gate CZ_{ab} can be applied in time $O(n^2)$. Our discussion here follows [3]. Application of CZ_{ab} is broken down into several subcases. The simplest case is if U_a and U_b are both diagonal, in which case CZ_{ab} commutes past them. In this case we just replace $E(G) \leftarrow E(G) \Delta \{ab\}$ in time $O(1)$.

We will now assume that at least one of U_a, U_b is not diagonal. We will try to perform local complementation so as to find an equivalent representation of $|\psi\rangle$ in which U_a and U_b *are* diagonal.

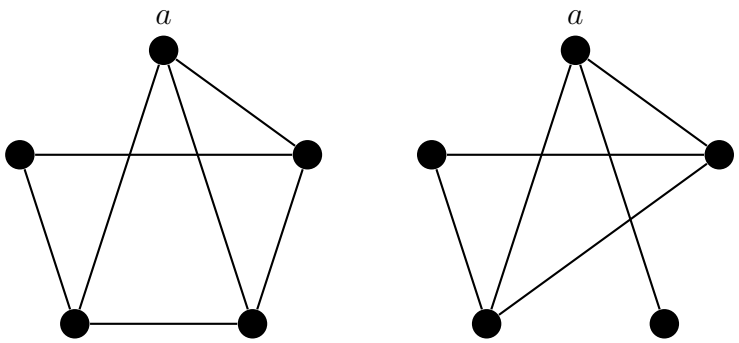


Figure 1.1: *Left:* A graph G. *Right:* The graph $L_a(G)$.

Suppose next that at least one of a and b is not isolated. We'll assume that b has a neighbour c. Using Lemma 7 it is always possible to find a representation of $|\psi\rangle$ with $U_b = I$: Start by decomposing U_b in terms of S and SHS. Such a decomposition can always be found with $O(1)$ terms. To remove a factor of SHS, perform local complementation at b. To remove a factor of S, perform local complementation at c. Because local complementation at a vertex does not change its neighbourhood, c will remain a neighbour of b throughout this process, so local complementation at c will always remove a factor of S. This takes time $O(n^2)$.

Now U_b is diagonal. At this point, a may or may not have a neighbour $d \neq b$. If d exists, we repeat the procedure above using a and d, which brings U_a to the identity. This procedure may alter U_b, but since local complementation *at* b won't be performed, U_b will only accumulate diagonal factors of S^3. Now U_a and U_b are both diagonal, so we can replace $E \leftarrow E\Delta\{ab\}$ as before. If no such d exists, then $N(a)$ is either $\{b\}$ or $\varnothing$. We then set $y = |N(a)|$ and use the following claim. This claim can be proved by enumerating all $O(1)$ possibilities.

Claim 8. Let U_a, U_b be local Clifford operators and $y \in \{0,1\}$. Then in time $O(1)$ we can find local Cliffords U'_a, U'_b and $x \in \{0,1\}$ such that $\mathrm{CZ}(U_a \otimes U_b)\mathrm{CZ}^y |{+}{+}\rangle = (U'_a \otimes U'_b)\mathrm{CZ}^x |{+}{+}\rangle$ and if U_b is diagonal then so is U'_b.

We can then rewrite $\mathrm{CZ}_{ab} |\psi\rangle$ as follows. We give an example of this rewriting procedure in Fig. 1.2. Here we implicitly take each U_i to be acting on qubit i.

$$\mathrm{CZ}_{ab} \prod_i U_i \prod_{jk \in E} \mathrm{CZ}_{jk} |+^n\rangle = \left(\prod_{i \notin \{a,b\}} U_i \prod_{jk \in E \setminus \{ab\}} \mathrm{CZ}_{jk} \right) \mathrm{CZ}_{ab} U_a U_b \mathrm{CZ}^y_{ab} |+^n\rangle \tag{1.58}$$

$$= \left(\prod_{i \notin \{a,b\}} U_i \prod_{jk \in E \setminus \{ab\}} \mathrm{CZ}_{jk} \right) U'_a U'_b \mathrm{CZ}^x_{ab} |+^n\rangle \tag{1.59}$$

$$= \left(\prod_{i \notin \{a,b\}} U_i \right) U'_a U'_b \mathrm{CZ}^x_{ab} \prod_{jk \in E \setminus \{ab\}} \mathrm{CZ}_{jk} |+^n\rangle \tag{1.60}$$

The first equality uses the fact U_b is diagonal, so it commutes past any CZ gates acting on b. It also uses the fact that a has no neighbours other than possibly b to commute U_a

past the CZ gates. The third equality uses the same idea. Eq. 1.60 tells us to perform the update $E \leftarrow E \cup \{ab\}$ if $x = 1$ and $E \leftarrow E \setminus \{ab\}$ otherwise, and to update $U_a, U_b \leftarrow U'_a, U'_b$. The total runtime in this case is $O(n^2)$, because we performed a constant number of local complementations in $O(n^2)$ time, and then used Claim 8 in $O(1)$ time.

The final case to consider is when $N(a) = N(b) = \varnothing$. In this case we use Claim 8 with $y = 0$. Because a and b are isolated, U_a, U_b and U'_a, U'_b commute with $\prod_{jk \in E} \mathrm{CZ}$, so Eqs. 1.58–1.60 will still hold.

Because local complementation induces a global phase (see Eq. 1.50), the representation of $\mathrm{CZ}_{ab} |\psi\rangle$ that this subroutine leaves us with has a global phase $e^{wi\pi/4}$, for some $w \in \{0, \ldots, 7\}$. If $w \neq 0$, then this representation is not technically be in the form of Eq. 1.48. The simplest way to handle this is to just store w as another component of the classical representation. Alternatively, if we insist that Eq. 1.48 has phase 1, then we can choose an arbitrary vertex and perform local complementation $8 - w$ times using Lemma 7.

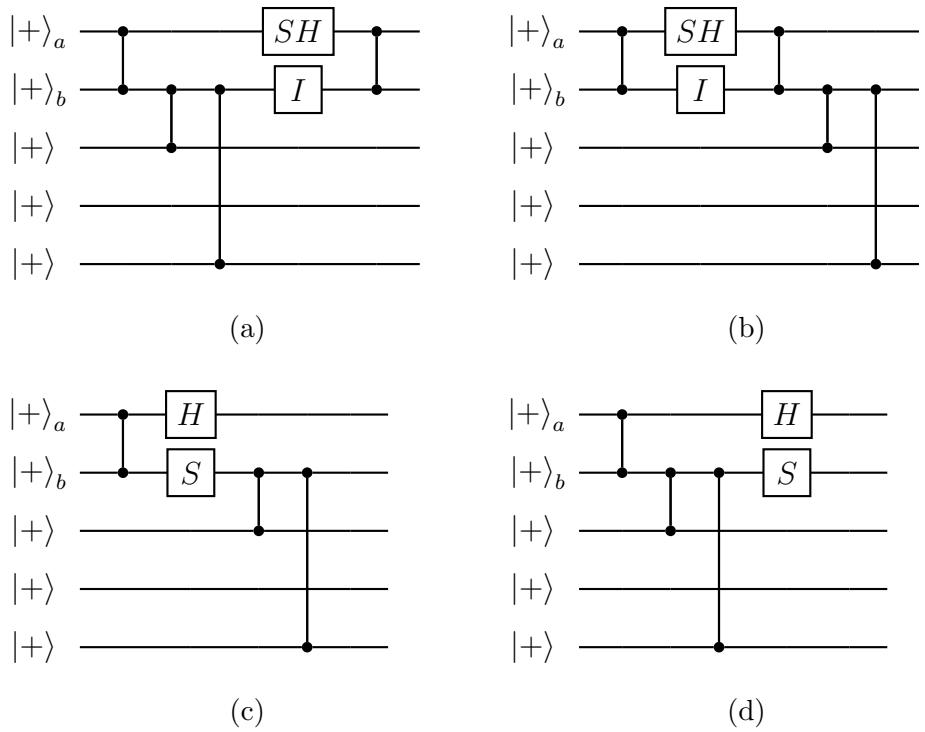


Figure 1.2: Circuit diagram depiction of using Claim 8 to apply a CZ gate. To go from Fig. 1.2b to Fig. 1.2c we have used Claim 8 to produce the identity $(H \otimes S)\mathrm{CZ} |{+}{+}\rangle = \mathrm{CZ}(SH \otimes I)\mathrm{CZ} |{+}{+}\rangle$.

Finally, we show how a computational basis measurement on a qubit a can be performed in time $O(n^2)$. Computational basis measurements can be simplified by using the fact that local Clifford operators map Pauli bases to Pauli bases. Projecting a qubit a with local operator U_a, gives

$$\frac{(I \pm Z)}{2} U_a = U_a \frac{I \pm U_a^\dagger Z U_a}{2}. \tag{1.61}$$

Because $U_a^\dagger Z U_a \in \mathcal{P}_1$, applying U_a and then measuring in the computational basis is equiva-

lent to measuring directly in the $U_a^\dagger Z U_a$ basis and then applying U_a. It's therefore sufficient to describe algorithms for performing X-, Y-, and Z-basis measurements on graph states.

The rules for updating the post-measurement state have previously been shown using the stabilizer formalism [18]. However because the stabilizer formalism is not phase-sensitive, these update rules are only proved up to a global phase. Here we give a phase-sensitive proof using affine form. We also assume that $N(a) \neq \varnothing$. If $N(a) = \varnothing$ then $|\psi\rangle = |\psi\rangle_a \otimes |\psi\rangle_{V\setminus a}$. In this case the measurement on $|\psi\rangle_a$ can be done in time $O(1)$.

Choose an arbitrary $b \in N(a)$. We'll now show that measurement in the P basis is random for any $P \in \{X, Y, Z\}$, and that for a measurement result $r \in \{0, 1\}$ the post-measurement state $\frac{1}{2}(I + (-1)^r P_a)|G\rangle$ is equal to $U(P, r)|r\rangle|G_P\rangle$, where the $U(P, r)$ are defined by:

$$\begin{aligned} U(Z,r) &= \prod_{c\in N(a)} Z_c^r \quad , \quad U(Y,r) = \tfrac{1-i^r}{\sqrt{2}} \prod_{c\in N(a)} S_c, \\ U(X,0) &= H_b \prod_{c\in N(b)\cap N(a)} Z_c \quad , \quad U(X,1) = X_b H_b \prod_{c\in N(b)\setminus N(a)\setminus\{a\}} Z_c. \end{aligned} \tag{1.62}$$

The graphs G_Z, G_Y are given by $G_Z = G - a$, $G_Y = L_a(G) - a$. Finally, the graph G_X has vertex set $V(G) \setminus \{a\}$ and edge set

$$\begin{aligned} E(G_X) = E(G) &\Delta N(a) \times N(b) \\ &\Delta\left(N(a) \cap N(b)\right) \times \left(N(a) \cap N(b)\right) \\ &\Delta\{b\} \times N(a). \end{aligned} \tag{1.63}$$

Applying each $U(P, r)$ takes time $O(n)$, and updating G takes time $O(n^2)$. To simplify notation we will assume that a is the first qubit and b the second. Let us also write $\bar{x} = x_2 \ldots x_n$ for $x \in \{0, 1\}^n$. We will use the affine form representation of $|G\rangle$ from Eq. 1.53, and set $q(x) = x^T Q x$.

Case 1: Z basis. Up to relative phases, each computational basis state in Eq. 1.53, has equal probability amplitude. Therefore both results $\{0, 1\}$ occur with equal probability. To choose the result r we set $x_a \leftarrow r$, leaving the state

$$\frac{1}{\sqrt{2^{n-1}}} \sum_{\bar{x}\in\{0,1\}^{n-1}} (-1)^{q(r\bar{x})} |r\rangle|\bar{x}\rangle = U(Z, r)|r\rangle|G_Z\rangle\,. \tag{1.64}$$

We can write $q(r\bar{x})$ as

$$q(r\bar{x}) = \sum_{j>1} Q_{1j} r x_j + \sum_{k>j>1} Q_{jk} x_j x_k. \tag{1.65}$$

Since $Z_j|x\rangle = (-1)^{x_j}|x\rangle$, the first sum tells us that we have applied $\prod_{j\in N(1)} Z_j^r = U(Z, r)$. Because the adjacency matrix for $G - a$ is just Q with the ath row and column removed, the second sum tells us that the remaining $n - 1$ qubits are in state $|G_Z\rangle$.

Case 2: Y basis. Using the update rules for affine representation, we apply $H_a S_a^\dagger$ to map the Y basis to the computational basis. This leaves

$$H_a S_a^\dagger |G\rangle = \frac{e^{i\pi/4}}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} i^{\ell'(x)}(-1)^{q'(x)} |x\rangle\,, \text{ with} \tag{1.66}$$

$$\ell'(x) = x_1 + 1 + \sum_{j>1} Q_{1j}x_j, \tag{1.67}$$

$$q'(x) = x_1 + 1 + \sum_{k>j>1} (Q_{jk} + Q_{1j}Q_{1k})x_j x_k + \sum_{j>1} Q_{1j}x_1 x_j. \tag{1.68}$$

The double sum in Eq. 1.68 indicates that we have performed local complementation at a. After choosing the result r, the post-measurement state is $U(Y,r)\,|r\rangle\,|G_Y\rangle$.

Case 3: X basis. After changing bases we have

$$H_1 |G\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{\substack{y\in\{0,1\}\\ \bar{x}\in\{0,1\}^{n-1}}} (-1)^{q'(\bar{x})}(1 + (-1)^{\ell'(y\bar{x})}) |y\bar{x}\rangle\,, \text{ where} \tag{1.69}$$

$$q'(\bar{x}) = \sum_{k>j>1} Q_{jk}x_j x_k \tag{1.70}$$

$$\ell'(y\bar{x}) = y + \sum_{j>1} Q_{1j}x_j. \tag{1.71}$$

Using the fact that the summands of Eq. 1.69 vanish whenever ℓ' is odd, we can write

$$H_1 |G\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{\bar{x}\in\{0,1\}^{n-1}} (-1)^{q'(\bar{x})} |\sum_{j>1} Q_{1j}x_j\rangle\,|\bar{x}\rangle\,. \tag{1.72}$$

After choosing a measurement result r we have

$$x_2 = r + \sum_{j>2} Q_{1j}x_j. \tag{1.73}$$

Substituting this into Eq. 1.72 gives the post-measurement state

$$\frac{1}{\sqrt{2^{n-2}}} \sum_{x_3\ldots x_n\in\{0,1\}^{n-2}} (-1)^{q''(\bar{x})} |r\rangle\,|r + \sum_{j>2} Q_{1j}x_j\rangle\,|x_3\ldots x_n\rangle\,, \text{ where} \tag{1.74}$$

$$q''(\bar{x}) = \sum_{k>j>2} (Q_{jk} + Q_{2k}Q_{1j} + Q_{2j}Q_{1k})x_k x_j + \sum_{j>2}(r + Q_{1j})Q_{2j}x_j. \tag{1.75}$$

Using the identity $\mathrm{CNOT}_{j2} = H_2\mathrm{CZ}_{j2}H_2$ this can be rewritten as

$$X_2^r H_2 \frac{1}{\sqrt{2^{n-1}}} \sum_{\bar{x}\in\{0,1\}^{n-1}} (-1)^{q'''(\bar{x})} |r\rangle\,|x_2\ldots x_n\rangle\,, \text{ where} \tag{1.76}$$

$$q'''(\bar{x}) = q''(\bar{x}) + \sum_{j>2} Q_{1j}x_2x_j. \tag{1.77}$$

Applying the remaining Z operators removes any linear terms from q'''. The remaining terms then correspond to the adjacency matrix of G_X.

1.5 Discussion

	S	H	CZ	CNOT	X	Z	Tens. prod.	Measure	Global phase
Tableau	n	n	$\min(kn, n^\omega)$	n	n	n	mn	$n^2k^{\omega-2}$	✗
Affine form	1	n^2	1	$\min(kn^2, n^\omega)$	1	1	mn	$n^2k^{\omega-2}$	✓
Graph states	1	1	n^2	n^2	1	1	mn	n^2	✓
CH form	1	n^2	1	1	n^2	1	mn	n^2	✓

Table 1.2: Time complexity of performing various operations on stabilizer states when using each of the Clifford simulation techniques of Chapter 1. Tensor products refer to taking the tensor product with an m-qubit stabilizer state. Expressions involving k denote "batch operations" in which we either apply k gates or perform k computational basis measurements in succession. The constant $\omega \in [2, 2.37286)$ [2] is the matrix multiplication exponent.

In this chapter we have described four algorithms for simulating Clifford circuits. Table 1.2 summarizes the time complexity of standard operations. Each algorithm uses its own representation of stabilizer states, and consequently incurs different time costs depending on which operation is performed. Depending on the makeup of the circuit to be simulated, one may want to choose one simulation method over another. For example, a tableau-based simulator would be a good choice for a high-depth circuit with many Hadamard and CZ gates, because any other simulation method requires time $O(n^2)$ to apply either H or CZ.

The most obvious open problem is whether any of these simulation techniques can be improved. Each simulation method has at least one type of operation that takes time $\Omega(n^2)$ in the worst case. Must this be true for any Clifford simulator? Or is there a way to perform every type of operation in sub-quadratic time? There are several ways one might hope to do so. One option is to simply find better update rules. For example, perhaps in the graph state formalism there is a way to apply a CZ gate or perform a measurement without requiring local complementation. Another way one might hope to do this is by somehow combining multiple simulation methods. For example, we might hope to represent a state in affine form to apply S and CZ gates quickly, and then convert to a tableau representation to apply Hadamards. This seems improbable, as the cost of converting one representation to another is likely to be quadratic in general. A third way of finding a speedup might be to find a completely new representation of stabilizer states. One technique not discussed in this chapter is that of simulating Clifford circuits by computing the Tutte polynomial of binary matroids [31, 24].

One significant advantage that tableau representation has over its phase-sensitive counterparts is that any elementary gate can be applied in $O(n)$ time. In a high-depth circuit, phase-sensitive simulation may therefore be quite costly. Speeding up the application of single-qubit gates in these three frameworks may help improve the viability of stabilizer rank methods, discussed in Chapter 3, which require phase-sensitivity.

Chapter 2

Graph state simulation

In this chapter we summarize joint work with David Gosset, Daniel Grier, and Luke Schaeffer [15]. We consider a task called the *graph state simulation problem*. Before stating the problem in full, let us give an informal description of a special case to build intuition. The input is a graph $G = (V, E)$ and a Pauli basis $P_v \in \{X, Y, Z\}$ for each vertex $v \in V$. The task is to simulate a measurement of $|G\rangle$ with qubits measured in the P_v bases. In the general version of the graph state simulation problem, which we now state, this special case corresponds to the case where $\mathcal{P} = \varnothing$, and consequently Eq. 2.2 evaluates to 1.

Graph State Simulation Problem. The input to the graph state simulation task is a graph $G = (V, E)$ with $n = |V|$ vertices, a partition $V = \mathcal{S} \cup \mathcal{P}$, a string $m \in \{0,1\}^{|\mathcal{P}|}$, and a Pauli $P_v \in \{X, Y, Z\}$ for each $v \in V$. For each v, let U_v be the local Clifford mapping each P_v basis to the computational basis. That is,

$$U_v = \begin{cases} H, & \text{if } P_v = X \\ HS^\dagger, & \text{if } P_v = Y \\ I, & \text{if } P_v = Z. \end{cases} \tag{2.1}$$

Let

$$p_{\mathcal{P}}(m) = || \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle ||_2^2. \tag{2.2}$$

The task is to output $z \in \{0,1\}^{|\mathcal{S}|}$ sampled from

$$p(z) = \frac{1}{p_{\mathcal{P}}(m)} | \langle z|_{\mathcal{S}} \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle |^2, \tag{2.3}$$

or else report that $p_{\mathcal{P}}(m) = 0$.

We sometimes refer to the cases where $\mathcal{P} \neq \varnothing$ and $\mathcal{P} = \varnothing$ as the graph state simulation problem with and without postselection, respectively. We have already given some intuition about the graph state simulation without postselection. The graph state simulation problem with postselection also corresponds to simulating a measurement of $|G\rangle$ with qubits measured in the P_v bases, but this time we must postselect on the qubits corresponding to $\mathcal{P}$ and require

that they have measurement results given by m.

A classical algorithm can solve the graph state simulation problem without postselection in time $O(n^\omega)$: Using the tableau representation of stabilizer states, we can form $|G\rangle$ by initializing a tableau representation of $|0^n\rangle$ and simulating the application of $(\prod_{e\in E} \mathrm{CZ}_e)H^{\otimes n}$, all in time $O(n^\omega)$. Then we simulate applying $\bigotimes_v U_v$ in time $O(n^2)$ and simulate measurements on all qubits in time $O(n^\omega)$. Postselection can also be incorporated with the same runtime, but we will not prove this. The main theorem of this chapter gives a quadratic improvement on the naïve approach described above when G is planar.

Theorem 9. *Let G be planar. Any instance of the graph state simulation problem on G can be solved classically in time $\widetilde{O}(n^{\omega/2})$.*

A consequence of Theorem 9 is that if $\omega = 2$ then quantum computers have essentially no advantage in terms of gate complexity – which we use synonymously with time complexity – when solving the graph state simulation problem without postselection on planar graphs. A simple quantum algorithm for doing this would be to prepare $|+^n\rangle$, apply $(\bigotimes_v U_v)(\prod_{e\in E} \mathrm{CZ}_e)$, and measure. Because planar graphs have $O(n)$ edges, this algorithm has gate complexity $O(n)$, and because planar graphs may have as many as $3n-6$ edges, any quantum algorithm will need time $\Omega(n)$ in the worst case. On the other hand, if $\omega = 2$ then the classical algorithm from Theorem 9 runs in time $\widetilde{O}(n)$. Our results do, however, leave open the possibility of a gate complexity advantage in the case where G has $\Omega(n)$ edges but is nonplanar. In contrast, the related problem where G is any subgraph of the $\sqrt{n}\times\sqrt{n}$ grid graph, $\mathcal{P} = \varnothing$, and it suffices to output *any* z with $p(z) \neq 0$ (not necessarily sampled from the distribution of Eq. 2.3), has been previously used to show a *depth*-advantage for quantum circuits [8]. The simple quantum algorithm for doing this by preparing $|+^n\rangle$, applying $(\bigotimes_v U_v)(\prod_{e\in E} \mathrm{CZ}_e)$, and then measuring, can be implemented in constant depth by layering CZ gates according to an edge colouring of the grid graph [19], which has constant chromatic index. However, this problem cannot be solved by a classical circuit in NC^0 [8] or even AC^0 [36].

The main applications of graph state simulation are proved in Section 2.7. For an n-qubit Clifford circuit C, let G be the graph with vertex set $[n]$ and an edge ij if and only if CZ_{ij} is a gate in C. We say that the two-qubit gates of C *act along the edges of* G.

Theorem 10. *Let C be an n-qubit depth-d Clifford circuit whose two-qubit gates act along the edges of a planar graph G. There exist classical algorithms for the following tasks.*

a. Sampling z from the output distribution $\Pr[z] = |\langle z|C|0^n\rangle|^2$ in time $\widetilde{O}(n^{\omega/2}d^\omega)$.

b. Given $z \in \{0,1\}^n$, computing $|\langle z|C|0^n\rangle|^2$ in time $\widetilde{O}(n^{\omega/2}d^\omega)$.

c. Given $z \in \{0,1\}^n$, computing $\langle z|C|0^n\rangle$ in time $\widetilde{O}(n^{3/2}d^3)$.

In the case where $d = O(\mathrm{polylog}(n))$ these algorithms each give quadratic improvements over the standard Clifford simulation techniques of Chapter 1. The powers of ω in Theorems 10a and 10b arise from the use of the matrix-multiplication-time version of tableau simulation [15]. On the other hand, the powers of 3 in Theorem 10c arise from the fact that computing

$\langle z|C|0^n\rangle$ requires information about global phase, and therefore we must use a slower phase-sensitive simulator like CH form or affine form.

An outline of this chapter is as follows. In Section 2.1 we build intuition for our solution to the graph state simulation problem by considering two concrete examples. In Section 2.2 we cover some graph-theoretic preliminaries. In Section 2.3 we describe a quantum circuit $\mathcal{C}$. We show that simulating $\mathcal{C}$ with some postselected measurements can help solve the graph state simulation problem, and that if G is planar it can be simulated in time $\widetilde{O}(n^{\omega/2})$ *without* postselection. In Section 2.4 we show how to modify the result obtained in Section 2.3 to account for postselection, which also takes time $\widetilde{O}(n^{\omega/2})$ if G is planar. In Section 2.5 we work through an explicit example of the subroutine from Section 2.4. In Section 2.6 we describe two variants of the graph state simulation problem. In one variant, the task is to compute $|\langle z|\bigotimes_v U_v|G\rangle|^2$ for a given $z \in \{0,1\}^n$, and in the other the task is to compute $\langle z|\bigotimes_v U_v|G\rangle$. We show that these tasks can be performed classically in time $\widetilde{O}(n^{\omega/2})$ and $\widetilde{O}(n^{3/2})$, respectively, with only small modifications to Sections 2.3 and 2.4. Finally, in Section 2.7 we prove Theorems 10a, 10b, and 10c and also give analogous theorems for contracting Clifford tensor networks.

2.1 Algorithms for graph state simulation

The time complexity of simulating a general n-qubit Clifford circuit depends on n. Using a tableau-based simulator, for example, we saw in Section 1.1 that applying any Clifford gate takes time $O(n)$. In order to solve the graph state simulation problem quickly we will try to minimize the number of qubits stored (classically) in memory at any given moment. In this section we illustrate this idea by giving two algorithms for solving the graph state simulation problem without postselection on specific graphs. Both algorithms perform better than the $O(n^\omega)$-time algorithm described at the start of this chapter.

The n-vertex *star graph* has vertex set $\{v_1,\ldots,v_n\}$ and edge set $\{v_1v_2, v_1v_3, \ldots, v_1v_n\}$. Fig. 2.1 shows the 4-vertex star graph and an algorithm for solving the graph state simulation problem without postselection on it, which we now describe. First, initialize a tableau representation of $|+\rangle \otimes |+\rangle$ corresponding to vertices v_1 and v_2. Simulate applying CZ_{v_1,v_2}. Then simulate applying U_{v_2} and measuring v_2, and remove that qubit from our classical representation. Next, initialize a qubit $|+\rangle$ corresponding to v_3, and continue the process. After the qubits corresponding to $v_2,\ldots,v_n$ have all been measured, we also simulate a measurement on v_1. This algorithm does return a solution to the graph state simulation problem. We have used the fact that measuring one qubit (e.g. v_2) commutes with applying an operation to another (e.g. v_3). At any given moment, only the qubits corresponding to v_1 and some other v_j are stored in memory. Therefore all operations on v_j take time $O(1)$. The total runtime of this algorithm is then $O(n)$.

A more complicated example of graph state simulation without postselection is shown by the $\sqrt{n}\times\sqrt{n}$ grid graph and is similar to the nested dissection method for solving systems of linear equations [14]. We begin by partitioning the grid into quadrants, and consider the four subgrids of size $\approx \frac{\sqrt{n}}{2}\times\frac{\sqrt{n}}{2}$ on the *interior* of each quadrant. (See Fig. 2.2a.) For each of these subgrids, we recursively initialize a classical representation of qubits corresponding to the vertices of the subgrid, apply CZ gates for every edge in the subgrid, measure all qubits

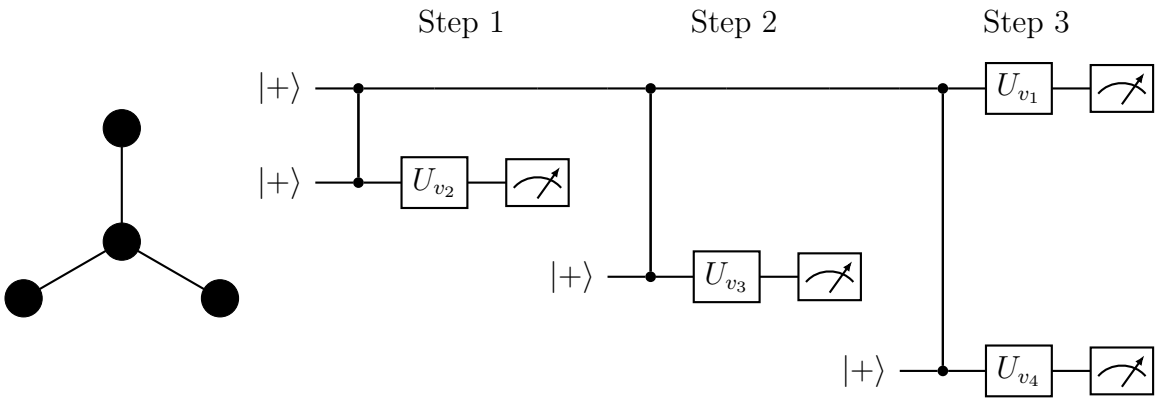


Figure 2.1: *Left:* Four-vertex star graph. *Right:* Circuit diagram depicting the sequence of operations taken in solving the graph state simulation problem.

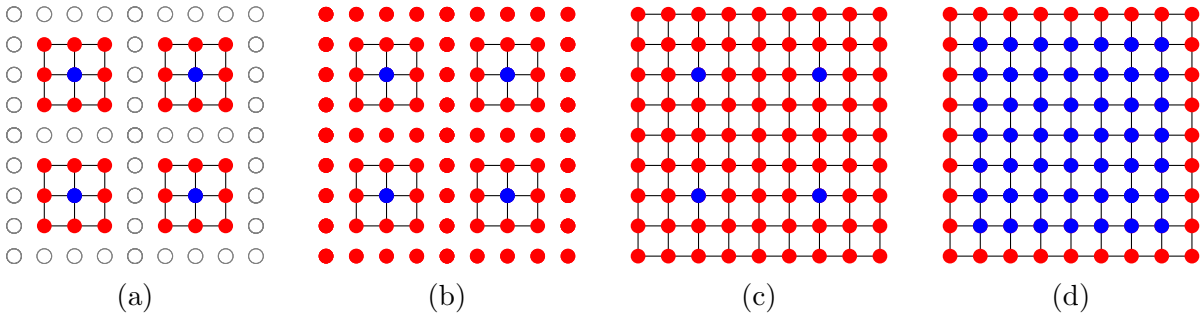


Figure 2.2: Solving the graph state simulation problem on the $\sqrt{n} \times \sqrt{n}$ grid graph. Red qubits are stored in memory as the active set, blue qubits have been measured and are no longer stored in memory, white qubits have not yet been operated on, and edges represent CZ gates that have already been applied. In Fig. 2.2a we use recursion on the interior of each quadrant of the overall grid. In Fig. 2.2b we initialize the qubits that are not on the interior of the quadrants used in the recursive step. In Fig. 2.2c we apply all remaining CZ gates. In Fig. 2.2d we complete the recursion by simulating measurements on all qubits in the interior of the overall grid.

on the *interior* of the subgrid in the P_v bases, and remove the measured qubits from our representation. This step, shown in Fig. 2.2a, leaves $O(\sqrt{n})$ unmeasured qubits being stored in memory. There are also $O(\sqrt{n})$ qubits that are not part of any subgrid (these are the white qubits in Fig. 2.2a), which we then initialize in our classical representation (shown in Fig. 2.2b). As there are now $O(\sqrt{n})$ qubits being stored in memory, we can apply the matrix-multiplication-time version of tableau simulation from [15] to simulate the application of all remaining CZ gates (Fig. 2.2c) and measure the qubits on the interior of the grid (Fig. 2.2d). This takes time $O(n^{\omega/2})$. At the final layer of recursion we can also simulate measurements on the $O(\sqrt{n})$ qubits on the perimeter of the grid in time $O(n^{\omega/2})$ (not shown in Fig. 2.2). If $T(n)$ is the runtime on a grid of size $\sqrt{n} \times \sqrt{n}$, then we have the recurrence relation

$$T(n) = 4T\left(\frac{n}{4}\right) + O(n^{\omega/2}) = \widetilde{O}(n^{\omega/2}). \tag{2.4}$$

Both of the above algorithms are examples of what we call the *active set* approach, first described in [8]. This approach relies on a schedule for initializing qubits, applying gates, and performing measurements, all the while only storing a small set of "active" qubits in memory. We were unable to generalize the active set approach while retaining a runtime of $\widetilde{O}(n^{\omega/2})$. A key feature of our algorithm, which illustrates its deviation from the active set approach, is that there may be many qubits corresponding to the same vertex. To describe this algorithm in full we must define tree decompositions.

2.2 Tree decompositions

Definition 11 ([30])**.** Let $G = (V, E)$ be a graph. A tree decomposition is a set $\{B_i\}_i$ of subsets of V, known as bags, which are arranged at the nodes of a tree graph $(\{B_i\}_i, E')$ such that the following three conditions hold.

- Each vertex appears in a bag: $\bigcup_i B_i = V$.
- Each edge appears in a bag: For each $uv \in E$, there exists i with $u, v \in B_i$.
- For each $v \in V$, the set of bags containing v form a connected subtree of $(\{B_i\}, E')$.

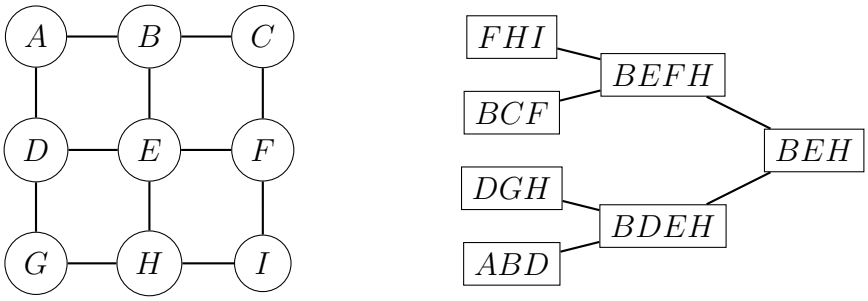


Figure 2.3: *Left:* A graph. *Right:* A tree decomposition of width 3. It can be shown that the graph on the left has treewidth exactly 3.

An example is shown in Fig. 2.3. Every graph admits a trivial tree decomposition by placing all vertices into a single bag, but most applications favour tree decompositions in which each bag is small. The *width* of a tree decomposition is defined as $|T| = \max_i |B_i| - 1$, and the *treewidth* $\mathrm{tw}(G)$ of a graph as the minimum $|T|$ over all tree decompositions T. For $c > 0$ we also use the notation

$$||T||_c = \left(\sum_i |B_i|^c \right)^{1/c}. \tag{2.5}$$

We will insist that our algorithm uses a particular type of tree decomposition.

Definition 12. A *nice* tree decomposition is tree decomposition in which the tree has been rooted, the root bag is empty, and each node B falls into exactly one of three categories:

Introduce nodes: B is a leaf node, or B has exactly one child and $\text{Child}(B) \subseteq B$.

Forget nodes: B has exactly one child and $B \subseteq \text{Child}(B)$.

Merge nodes: B has two children and is equal to their union.

An example of a nice tree decomposition is shown in Fig. 2.5, top. We use without proof a result which finds a nice tree decomposition for planar graphs.

Theorem 13 ([15])**.** *Let G be a planar graph on n vertices. There exists a classical algorithm running in time $\widetilde{O}(n^{\omega/2})$ finding a nice tree decomposition T of G with $||T||_c^c = \widetilde{O}(n^{c/2})$ for all $c \geq 2$.*

Our algorithm for solving the graph state simulation problem actually has a runtime that is upper bounded by $O(||T||_\omega^\omega)$. Theorem 9, which says that the graph state simulation problem can be solved classically in time $\widetilde{O}(n^{\omega/2})$ for planar graphs, can be proved by using Theorem 13 and then the following theorem. The next two sections will prove Theorem 14.

Theorem 14. *Given a tree decomposition T for a graph G, any instance of the graph state simulation problem on G can be solved classically in time $O(||T||_\omega^\omega)$.*

The algorithm of Theorem 13 is the key reason why the simulation algorithms of this chapter are specialized to planar graphs. Partially, this is because finding a tree decomposition of a general graph G with width relatively close to $\text{tw}(G)$ is quite hard [e.g. 5]. However, even given a nice tree decomposition T for G, the best upper bound we were able to find was $||T||_\omega^\omega = O(n|T|^{\omega-1})$. Our algorithms therefore can be used to solve the graph state simulation problem on nonplanar graphs in time $O(n|T|^{\omega-1})$, but only if T is also given as input.

2.3 Sampling subroutine

This section and the next are devoted to proving Theorem 14 by describing an algorithm solving the graph state simulation problem. We will start by using the given tree decomposition T and bases $\{P_v\}_v$ to define a circuit $\mathcal{C} = \mathcal{C}(G, T, \{P_v\}_v)$. The circuit $\mathcal{C}$ will have a structure that is similar to that of T. Qubits of $\mathcal{C}$ will be initialized in the portion corresponding to the leaves of T. The computation described by $\mathcal{C}$ will then proceed from the leaves of T to the root. We will show that if certain qubits are postselected, then $\mathcal{C}$ prepares the state $\bigotimes_v U_v |G\rangle$. We will then show how to simulate $\mathcal{C}$ and also accommodate the necessary postselection.

Each node B of T is mapped to a circuit gadget. That gadget is determined by whether B is an introduce, forget, or merge node. Gadgets are then arranged in the same tree structure as T, so that the input qubits to the gadget for B are the output qubits of the gadget for $\text{Child}(B)$. Note that the gadget for B always contains at least one qubit corresponding to each $v \in B$.

Introduce nodes: If B is a leaf node, the state $|+^{|B|}\rangle$ is initialized. Otherwise let C be the child of B. The gadget acts as the identity on the qubits corresponding to $v \in B \cap C$, and for each $v \in B \setminus C$ a qubit in state $|+\rangle$ is initialized.

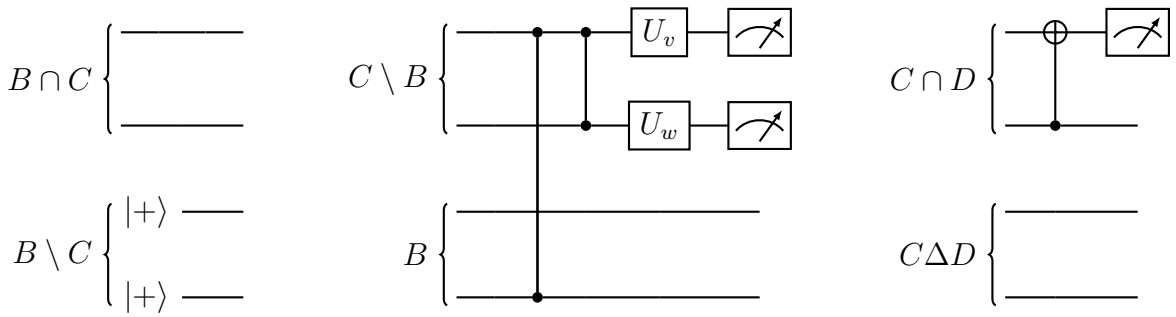


Figure 2.4: *Left:* Introduce node gadget. *Centre:* Forget node gadget when $\{v, w\} = C \setminus B$. The top two qubits are data qubits. *Right:* Merge node gadget ($C \Delta D$ denotes the symmetric difference of C and D). The top qubit is a merge ancilla.

Forget nodes: Let C be the child of B. For each $uv \in E(G)$ with $u, v \in C$ and at least one of u, v not in B, the gadget will receive exactly one qubit corresponding to each of u, v as input. Apply CZ between these qubits. After all such CZ gates have been applied, for each $v \in C \setminus B$ we then apply U_v and measure the qubit corresponding to v. We call the measured qubit a *data qubit*.

Merge nodes: Let C, D be the children of B. For each $v \in C \cap D$, there are two qubits corresponding to v given as input to B's gadget: one from C's gadget and one from D's gadget. For each such v, apply a CNOT gate between them (the choice of which of the two qubits is the control and which is the target is arbitrary) and measure the target qubit. We call the measured qubit a *merge ancilla*.

We will use $\mathcal{C}$ to denote the unitary part of this circuit. Examples of the individual gadgets and a complete circuit are shown in Figs. 2.4 and 2.5, respectively. The presence of merge ancillas highlights how our algorithm differs from the active set approach. In the active set approach, there is a one-to-one correspondence between qubits and vertices. In $\mathcal{C}$, there may be several qubits corresponding to the same vertex. Note however that there is exactly one *data* qubit for each vertex v. The set of bags containing v form a connected subtree of T with some root B. Therefore all qubits corresponding to v must be merged in the portion of $\mathcal{C}$ corresponding to this subtree, except for the qubit corresponding to v that appears in B, which is measured as a data qubit in the gadget corresponding to Parent(B).

We will write n_a for the number of merge ancillas and $n_t \coloneqq n + n_a$ for the total number of qubits in $\mathcal{C}$. Lemma 15 tells us that postselecting merge ancillas leaves the remaining n qubits to in the state $\bigotimes_v |G\rangle$.

Lemma 15. *Let $\mathcal{C}$ be as above and let $\langle 0^{n_a}|$ act on merge ancillas. Then*

$$(\langle 0^{n_a}| \otimes I^{\otimes n})\mathcal{C} |+^{n_t}\rangle = 2^{-n_a/2} \bigotimes_v U_v |G\rangle . \tag{2.6}$$

Proof. For each edge $uv \in E$, there is a unique CZ gate in $\mathcal{C}$ that acts between some pair of qubits corresponding to u and v, respectively. (These qubits could be either merge ancillas or data qubits.) To see that this is true, we use the fact that u and v must appear together

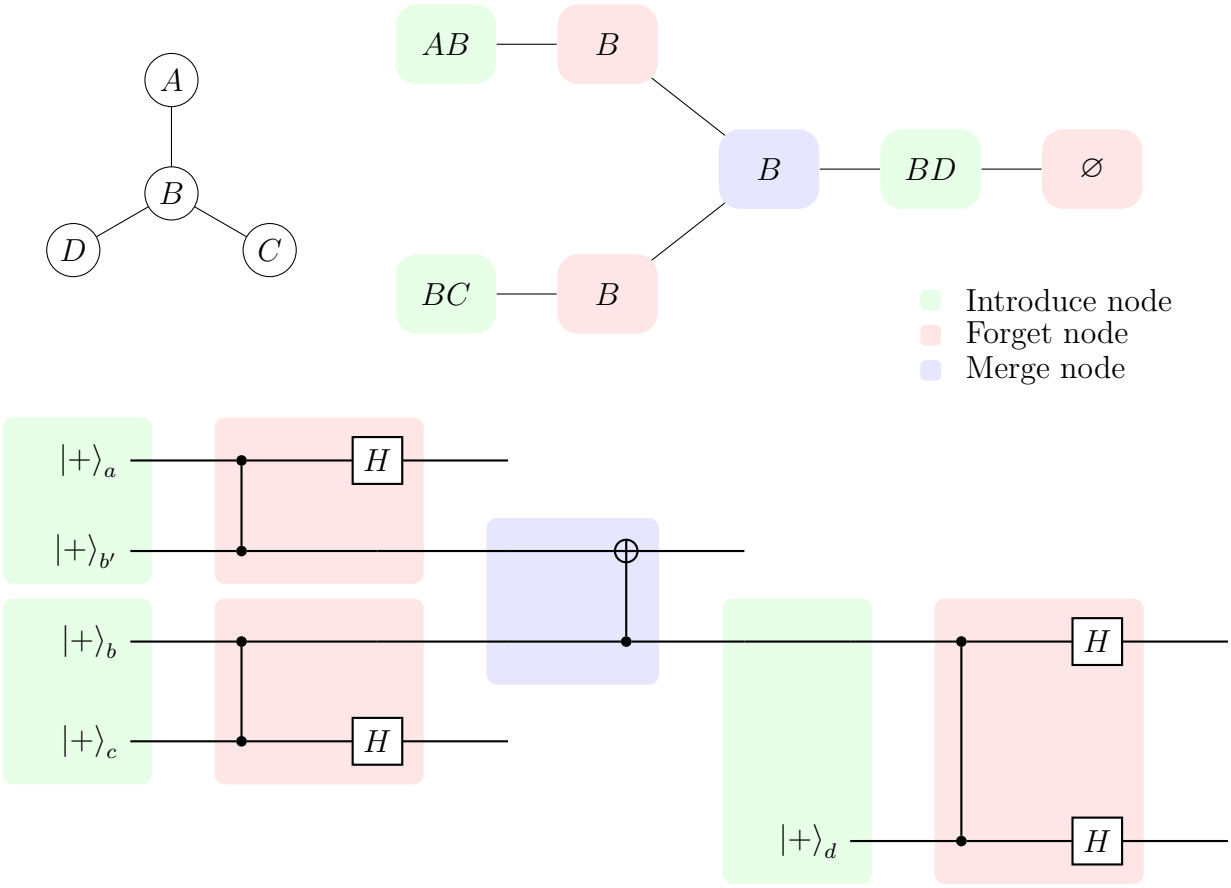


Figure 2.5: *Top:* A graph and a nice tree decomposition. *Bottom:* The circuit $\mathcal{C}$ in the case where $\bigotimes_v U_v = H^{\otimes 4}$ (i.e. all data qubits are to be measured in the X-basis). The qubits a, b, c, d are data qubits, and b' is a merge ancilla.

in some bag of T. Let B be the bag containing both u and v that is nearest to the root of T. By the connectivity property, B is unique. By definition of $\mathcal{C}$, there is a CZ gate between qubits corresponding to u, v in the forget node gadget of Parent(B). Because CZ gates are only applied when a qubit is forgotten, there is only one such CZ gate in all of $\mathcal{C}$. Some of these CZ gates are applied to merge ancillas instead of data qubits. We will show that postselecting a merge ancilla "transfers" the effect of a CZ gate on that merge ancilla to the qubit to which it is merged. One way of understanding this fact is that applying the CNOT gate of a merge gadget and then postselecting the merge ancilla to be in state $|0\rangle$ applies the map $(I \otimes \langle 0|)\text{CNOT} = |0\rangle\langle 00| + |1\rangle\langle 11|$.

We will rewrite $\mathcal{C}$ using the two circuit identities of Fig. 2.6. The circuit $\mathcal{C}$ consists of a layer of CNOT and CZ gates, and then a layer of local operators $\bigotimes_v U_v$. We will rewrite $(\langle 0^{n_a}| \otimes I^{\otimes n})\mathcal{C}\,|+^{n_t}\rangle$ as a layer of CNOT gates, then CZ gates, and then $\bigotimes_v U_v$. Because $\text{CNOT}\,|{+}{+}\rangle = |{+}{+}\rangle$, any CNOT gates can then be ignored. Fig. 2.6 shows how CNOT gates can be moved to the left of CZ gates. In particular, if a, b, c are qubits with CZ_{ab} appearing in $\mathcal{C}$ and b later being merged to c, then we can move the CZ_{ab} gate to the left of CNOT_{cb}

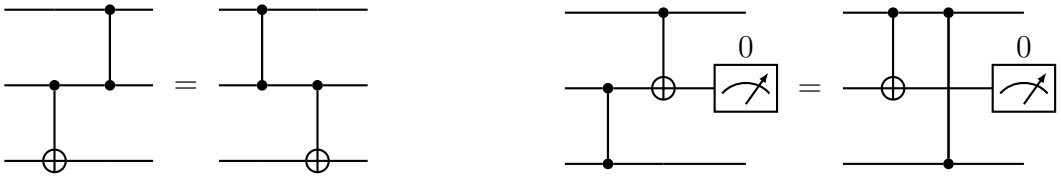


Figure 2.6: Commutation relations needed for Lemma 15. The second equality, which corresponds to Eq. 2.7, depends on the centre qubit being postselected.

using

$$(I_a \otimes \langle 0|_b \otimes I_c)\mathrm{CNOT}_{cb}\mathrm{CZ}_{ab} = (I_a \otimes \langle 0|_b \otimes I_c)\mathrm{CZ}_{ac}\mathrm{CNOT}_{cb}. \tag{2.7}$$

By shifting all CNOT gates to the start of the circuit, we rewrite the circuit such that no CZ gates act on merge ancillas. Because there is one CZ gate in $\mathcal{C}$ for each edge, and all CZ gates have been transferred to data qubits, we have

$$(I \otimes \langle 0^{n_a}|)\mathcal{C}(|+^n\rangle \otimes |+^{n_a}\rangle) = \left(\bigotimes_v U_v \prod_{ij\in E} \mathrm{CZ}_{ij} |+^n\rangle\right) \otimes \langle 0^{n_a}|+^{n_a}\rangle = 2^{-n_a/2} \bigotimes_v U_v |G\rangle . \tag{2.8}$$

□

For a string $y \in \{0,1\}^{n_t}$ indexed by qubits of $\mathcal{C}$, and a vertex $v \in V$, let us write $j(v) \in [n_t]$ to denote the index of y that represents the data qubit corresponding to v. Lemma 15 tells us that if $y \in \{0,1\}^{n_t}$ is chosen uniformly from the subset of $\{0,1\}^{n_t}$ satisfying

- $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 \neq 0$;
- $y_j = 0$ for each merge ancilla j; and
- $y_{j(v)} = m_v$ for each $v \in \mathcal{P}$;

then the $|\mathcal{S}|$ bits of y corresponding to data qubits for vertices in $\mathcal{S}$ form a solution to the graph state simulation problem. The first phase of our algorithm consists of the *sampling subroutine*, in which we find a string y with $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 \neq 0$. The second phase is the *correction subroutine*, in which we show how y can be modified so that it also satisfies the other two criteria above. Both algorithms run in time $O(||T||_\omega^\omega)$, which will prove Theorem 14.

Lemma 16 (Sampling subroutine)**.** *A classical algorithm running in time $O(||T||_\omega^\omega)$ can find $y \in \{0,1\}^{n_t}$ with $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 \neq 0$.*

We set some notation before proving the theorem. For each bag B, let T_B be the subtree of T rooted at B. Let $\mathcal{C}|_B$ denote the portion of $\mathcal{C}$ corresponding to B's gadget, and define $\mathcal{C}|_{T_B}$ similarly. For example, in the circuit $\mathcal{C}$ of Fig. 2.5, if B is the lone merge node, $\mathcal{C}|_B$ consists of two wires and a CNOT gate, and $\mathcal{C}|_{T_B}$ consists of four wires, two CZ gates, two Hadamard gates, and one CNOT gate.

Proof. Recall that in the two examples of Section 2.1 our goal was to store as few qubits in memory as possible. We will employ the same strategy here to simulate $\mathcal{C}$. For any bag B we will show that a recursive classical algorithm running in time $O(||T_B||_\omega^\omega)$ can simulate $\mathcal{C}|_{T_B}|+^{n_t}\rangle$ and output (1) measurement outcomes for qubits appearing in $\mathcal{C}|_{T_B}$ but nowhere else in $\mathcal{C}$; and (2) a classical representation of the qubits of $\mathcal{C}|_{T_B}|+^{n_t}\rangle$ that appear in $\mathcal{C}|_{T_B}$ and also somewhere else in $\mathcal{C}$. Taking B to be the root node then proves the theorem. For example, running the algorithm when B is the merge node in the circuit of Fig. 2.5 would output (1) measurement outcomes on the qubits labelled a, b', and c; and (2) a classical representation of a one-qubit state corresponding to the qubit labelled b.

The base case occurs when B is a leaf node. In this case the algorithm outputs a representation of $|+^{|B|}\rangle$. Otherwise, let C (and D, if B is a merge node) be the child(ren) of B. In every other case we use recursion on T_C, leaving us with the state $|\psi_C\rangle$. The same is done for D if applicable. Then depending on what type of node B is, we do as follows.

Introduce nodes: Output a classical representation of $|\psi_C\rangle \otimes |+^{|B\setminus C|}\rangle$.

Forget nodes: Simulate the application of $\mathcal{C}|_B$ to $|\psi_C\rangle$, simulate measurements on the qubits that do not appear in $\mathcal{C}|_{\text{Parent}(B)}$, and return the state on the unmeasured qubits.

Merge nodes: Form $|\psi_C\rangle \otimes |\psi_D\rangle$, simulate the application of $\mathcal{C}|_B$, simulate measurements on merge ancillas, and return the state on the unmeasured qubits.

We will carry out the simulation by representing all states in tableau form. The runtime in the base case is $O(|B|^2)$ as we initialize $|B|$ qubits. For non-leaf introduce nodes the runtime is $O(|B||C|)$, as we initialize $|B \setminus C|$ qubits. For forget nodes, there are $O(|C|)$ qubits, so all CZ gates can be applied simultaneously in time $O(|C|^\omega)$. Next, we apply the U_v operators to some subset of qubits and measure. Measuring all qubits simultaneously also takes time $O(|C|^\omega)$. For merge nodes there are $O(|B|)$ CNOT gates to apply, which takes time $O(|B|^2)$, and then measuring merge ancillas takes time $O(|B|^\omega)$.

The total runtime including the recursive step can therefore be upper bounded as

$$O(||T_C||_\omega^\omega) + O(|B|^2 + |C|^\omega) \quad \text{or} \quad O(||T_C||_\omega^\omega + ||T_D||_\omega^\omega) + O(|B|^\omega), \tag{2.9}$$

which are both $O(||T_B||_\omega^\omega)$. Taking B to be the root of T gives us a runtime of $O(||T||_\omega^\omega)$. □

Before continuing we state two facts about the sampling subroutine that are not necessary for solving the graph state simulation problem, but will be useful in Section 2.6.

Lemma 17. *Given $y \in \{0,1\}^{n_t}$, a classical algorithm running in time $O(||T||_\omega^\omega)$ can compute $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2$.*

Proof. We use the algorithm described by the proof of Lemma 16, but this time postselect on measurement outcomes given by y. If we ever encounter a deterministic measurement on a qubit that must give result $1 - y_j$, we can immediately stop and report $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 = 0$. Otherwise, since $\mathcal{C}$ is a Clifford circuit, we have $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 = 2^{-k}$, where k is the number of nondeterministic measurements encountered. □

Lemma 18. *Given $y \in \{0,1\}^{n_t}$, a classical algorithm running in time $O(||T||_3^3)$ can compute $\langle y|\mathcal{C}|+^{n_t}\rangle$.*

Proof. Running the sampling algorithm using CH form or affine form can be done in time $O(||T||_3^3)$. The slowdown comes from the fact that in forget nodes we may need to apply as many as $|B|$ Hadamard gates, which would take time $O(|B|^3)$. If we perform the simulation in this way, we can postselect on y as in Lemma 17 to find $|\langle y|\mathcal{C}|+^{n_t}\rangle|$. After the last qubit has been measured, the global phase given by the simulator will be the phase (as a complex number) of $\langle y|\mathcal{C}|+^{n_t}\rangle$. □

2.4 Correction subroutine

Lemma 16 gives us $y \in \{0,1\}^{n_t}$ with $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 \neq 0$. The core idea of the correction subroutine is the following fact: If $\alpha, \beta \in \{0,1\}^{n_t}$ are such that $i^\gamma X(\alpha)Z(\beta) \in \mathrm{Stab}(\mathcal{C}|+^{n_t}\rangle)$ for some γ (i.e. $X(\alpha)Z(\beta)$ is proportional to a stabilizer of $\mathcal{C}|+^{n_t}\rangle$), then

$$0 \neq |\langle y|\mathcal{C}|+^{n_t}\rangle|^2 = |\langle y|X(\alpha)Z(\beta)\mathcal{C}|+^{n_t}\rangle|^2 = |\langle y+\alpha|\mathcal{C}|+^{n_t}\rangle|^2. \tag{2.10}$$

Lemma 19 tells us that if $i^\gamma X(\alpha)Z(\beta)$ is a *uniformly* random stabilizer of $\mathcal{C}|+^{n_t}\rangle$ then $y+\alpha$ is sampled from the output distribution of $\mathcal{C}|+^{n_t}\rangle$.

Lemma 19 ([15])**.** *Let $|\psi\rangle$ be a stabilizer state, $a \in \{0,1\}^n$ with $\langle a|\psi\rangle \neq 0$, and P a uniformly random stabilizer of $|\psi\rangle$. Let $b \in \{0,1\}^n$ be such that $P|a\rangle \propto |b\rangle$. Then b is drawn from the distribution $\Pr[b] = |\langle b|\psi\rangle|^2$.*

The correction subroutine consists of choosing a Pauli $P_{\mathrm{cor}} := X(\alpha)Z(\beta)$ uniformly from the set of Paulis that (1) are proportional to a stabilizer of $\mathcal{C}|+^{n_t}\rangle$; (2) have $(y+\alpha)_j = 0$ for all merge ancillas j; and (3) have $(y+\alpha)_{j(v)} = m_v$ for all $v \in \mathcal{P}$. Let z be the $|\mathcal{S}|$ bits of $y+\alpha$ corresponding to the data qubits of $\mathcal{S}$. By Lemma 15, we have $p(z) \neq 0$, and by Lemma 19, we have that z is drawn from the distribution $p(z)$.

Proof of Lemma 19. Let $\Omega = \{Q \in \mathrm{Stab}(|\psi\rangle) : Q|a\rangle \propto |b\rangle\}$. Then $\Pr[b] = |\Omega|/2^n$. If $\langle b|\psi\rangle = 0$ the statement holds, because if a stabilizer P satisfying $P|a\rangle \propto |b\rangle$ existed, we would have $\langle b|\psi\rangle \propto \langle a|\psi\rangle \neq 0$, a contradiction. Assume now that $\langle b|\psi\rangle \neq 0$. If $P|a\rangle = c|b\rangle$ for some $c \in \mathbb{C}$ with $|c| = 1$, then c is independent of P, as $\langle b|\psi\rangle = \langle b|P|\psi\rangle = \bar{c}\langle a|\psi\rangle$. Then taking $Q_1, \ldots, Q_n$ as the generators of $\mathrm{Stab}(|\psi\rangle)$, we have

$$\langle b|\psi\rangle\langle\psi|a\rangle = \frac{1}{2^n}\langle b|\prod_{j=1}^{n}(I+Q_j)|a\rangle = \sum_{Q\in\mathrm{Stab}(|\psi\rangle)}\langle b|Q|a\rangle = c\frac{|\Omega|}{2^n}, \tag{2.11}$$

where the first equality comes from Claim 2, which tells us that $|\psi\rangle\langle\psi|$ projects onto the intersection of all $+1$ eigenspaces of the Q_j. Since $|\psi\rangle$ is a stabilizer state and $\langle b|\psi\rangle \neq 0$, we have $|\langle b|\psi\rangle| = |\langle a|\psi\rangle|$. Therefore the left hand side has modulus $\Pr[b]$. Since $|c| = 1$, taking the modulus on both sides completes the proof. □

For $k > 0$, the correction subroutine will represent Paulis $i^c X(a)Z(b) \in \mathcal{P}_k$ as elements $(a, b) \in \{0,1\}^{2k}$. We can safely ignore the phase i^c because it has no impact on Eq. 2.10.

We will often need to search for a Pauli whose X- or Z-components are of a particular form. We can represent any desired X- and Z-components by a *pattern* $\pi = (\pi^{(X)}, \pi^{(Z)}) \in \{0, 1, *\}^{2k}$, where $*$ indicates that an entry may be 0 or 1. The set of all strings agreeing with π on non-$*$ entries is denoted

$$\Pi = \{(a, b) \in \{0,1\}^{2k} : a_i = \pi_i^{(X)} \text{ if } \pi_j^{(X)} \in \{0,1\}, b_j = \pi_j^{(Z)} \text{ if } \pi_j^{(X)} \in \{0,1\}\}. \tag{2.12}$$

We will say that a Pauli $i^c X(a)Z(b)$ *respects* π if $(a, b) \in \Pi$. The main result of this section is the following theorem.

Lemma 20 (Correction subroutine)**.** *Let a pattern $\pi \in \{0,1\}^{2n_t}$ be given and let Stab $\subseteq \{0,1\}^{2n_t}$ be the stabilizer group of $\mathcal{C}|+^{n_t}\rangle$. Then a uniformly random element $P_{cor} \in$ Stab$\cap\Pi$ can be found in time $O(||T||_\omega^\omega)$.*

Before proving Lemma 20, we'll summarize how it allows us to complete the proof of Theorem 14, which we restate for convenience.

Theorem 14. *Given a tree decomposition T for a graph G, any instance of the graph state simulation problem on G can be solved classically in time $O(||T||_\omega^\omega)$.*

Proof of Theorem 14. Using Lemma 16 we find $y \in \{0,1\}^{n_t}$ with $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2 \neq 0$ in time $O(||T||_\omega^\omega)$. For each merge ancilla j, let $\pi_j^{(X)} = y_j$; for each $v \in \mathcal{P}$, let $\pi_{j(v)}^{(X)} = y_{j(v)} + m_v$; for each $v \in \mathcal{S}$, let $\pi_{j(v)}^{(X)} = *$; and let $\pi^{(Z)} = *^{n_t}$. Using Lemma 20 we find a uniformly random Pauli $X(\alpha)Z(\beta) \in$ Stab $\cap\, \Pi$ in time $O(||T||_\omega^\omega)$. Let z be the bits of $y + \alpha$ corresponding to data qubits of $\mathcal{S}$. By Lemma 15, $p(z) \neq 0$, and by Lemma 19, z is drawn from the distribution $p(z)$. □

We now build toward proving Lemma 20. For a given k-qubit stabilizer state $|\psi\rangle$ it will be useful to define certain subsets of its stabilizer group that are given by affine subspaces $\mathcal{A} = \{\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]x + \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] : x \in \{0,1\}^K\}$, where $K \leq k$ and the columns of $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ correspond to independent stabilizers of $|\psi\rangle$. For example, if $\mathcal{A} = \text{Stab}(|\psi\rangle)$, then $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] = 0^{2k}$, and the $K = k$ columns of $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ list the generators of Stab$(|\psi\rangle)$. We call $\mathcal{A}$ an *affine stabilizer subspace*. In particular, the set Stab $\cap\, \Pi$ is an affine stabilizer subspace, since it is the intersection of an affine stabilizer subspace (Stab) and an affine space (Π). Let us now state some facts about the manipulation of affine stabilizer subspaces.

- For any pattern π, the affine stabilizer subspace $\mathcal{A} \cap \Pi$ will be

$$\{\begin{bmatrix} A \\ B \end{bmatrix} x + \begin{bmatrix} a \\ b \end{bmatrix} : \begin{bmatrix} A' \\ B' \end{bmatrix} x = \begin{bmatrix} a' \\ b' \end{bmatrix} + \pi'\}, \tag{2.13}$$

  where A', B', a', b' and π' are obtained by removing row j from $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$, and π whenever $\pi_j = *$. This set is an affine stabilizer subspace and can be found in time $O(k^\omega)$ using a result of Ibarra, Moran, and Hui [20], stated as Theorem 35b in Appendix A. Therefore for any affine stabilizer subspace $\mathcal{A}$ and pattern π, we can find $\mathcal{A} \cap \Pi$ in time $O(k^\omega)$. We call the act of computing $\mathcal{A} \cap \Pi$ from $\mathcal{A}$ *enforcing* the pattern π.

- When a gate U is applied, $\mathcal{A}$ becomes $U\mathcal{A}U^\dagger$, which is also an affine stabilizer subspace. That is, we conjugate each Pauli from $\mathcal{A}$ by U. This can be done by conjugating the Pauli represented by each column of $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ by U and doing the same for $\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]$. Moreover, if all elements of $\mathcal{A}$ respect π and U acts only on qubits j for which $\pi_j^{(X)} = \pi_j^{(Z)} = *$, then all elements of $U\mathcal{A}U^\dagger$ will also respect π. Using the matrix-multiplication time version of the stabilizer formalism described in [15], we can update $\mathcal{A}$ when an arbitrary layer of CZ gates or local Cliffords are applied in time $O(k^\omega)$.

- For two stabilizer states $|\psi_1\rangle, |\psi_2\rangle$, and patterns π_1, π_2, the set of stabilizers of $|\psi_1\rangle \otimes |\psi_2\rangle$ that respect $\pi_1\pi_2$ is given by

$$\{P : P \text{ respects } \pi_1\pi_2\} = \{P_1 \otimes P_2 : P_1 \text{ respects } \pi_1 \text{ and } P_2 \text{ respects } \pi_2\}. \tag{2.14}$$

  Therefore if $\mathcal{A}_1$ and $\mathcal{A}_2$ represent the set of stabilizers respecting patterns π_1, π_2, then the set of stabilizers of $|\psi_1\rangle \otimes |\psi_2\rangle$ respecting $\pi_1\pi_2$ is the direct product $\mathcal{A}_1 \oplus \mathcal{A}_2$. In particular, if $|\psi_2\rangle = |+^\ell\rangle$ and $\pi_2 = *^{2\ell}$, then $\mathcal{A}_2 = \mathcal{I} \coloneqq \{\left[\begin{smallmatrix} I \\ 0 \end{smallmatrix}\right] u : u \in \{0,1\}^\ell\}$.

- Tracing out a qubit j from $\mathcal{A}$ amounts to removing the jth rows of A, B, a, b. This may cause the matrix $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ to no longer have full column rank, so we use another result from [20], stated as Theorem 35a, which also allows to remove any linearly dependent columns in time $O(k^\omega)$.

Proof of Lemma 20. A naïve way to sample an element from $\mathrm{Stab} \cap \Pi$ would be to first compute all of Stab in time $O(n_t^\omega)$ by computing $\mathcal{C}X_j\mathcal{C}^\dagger$ for each j, then compute $\mathrm{Stab} \cap \Pi$ using linear algebra, and finally choose a random element in the intersection. This approach takes time $O(n_t^\omega)$, which is too slow. Instead, we will combine all three steps into one and use the tree structure of T to get the runtime bound $O(||T||_\omega^\omega)$. In Section 2.5 we work through an example of the algorithm described below.

In the first phase of the subroutine we construct an affine stabilizer subspace $\mathcal{A}_B$ for each bag B. Recall that $\mathcal{C}|_B$ and $\mathcal{C}|_{T_B}$ denote the portions of $\mathcal{C}$ corresponding to B and to the subtree of T rooted at B. Let $\mathcal{Y}_B$ be the set of stabilizers of $\mathcal{C}|_{T_B}|+^{n_t}\rangle$ that respect π when π is restricted to the qubits that appear in $\mathcal{C}_{T_B}$ but nowhere else in $\mathcal{C}$. For example, $\mathcal{Y}_{\text{Root}} = \mathrm{Stab} \cap \Pi$, or if B is a leaf node with 2 qubits, then $\mathcal{Y}_B = \{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u : u \in \{0,1\}^2\} \cap \{0,1\}^4$. We define $\mathcal{A}_B$ to be the restriction of $\mathcal{Y}_B$ to the qubits of $\mathcal{C}|_B$. For example, in the circuit of Fig. 2.5, we have $\mathcal{A}_B \subseteq \mathbb{F}_2^4$ for each B, since each gadget contains 2 wires, and each wire contributes an X- and Z-component to $\mathcal{A}_B$.

The $\mathcal{A}_B$ are constructed inductively by moving from leaf nodes to the root. In the base case, we have $\mathcal{A}_B = \mathcal{I}$, corresponding to the stabilizers of $|+^{|B|}\rangle$. Otherwise, let C (and D, if B is a merge node) be the child(ren) of B. The first step will always be to take $\mathcal{A}_C$ and restrict it to coordinates of $\mathcal{C}|_B$, and do the same for D if applicable. This gives two new affine stabilizer subspaces, and reflects the fact that there may be some qubits in $\mathcal{C}|_C$ or $\mathcal{C}|_D$ that do not appear in $\mathcal{C}|_B$. If B is an introduce node, we then take the direct product with $\mathcal{I}$, corresponding to the qubits being introduced. If B is a forget node, we conjugate this affine stabilizer subspace through the gates of $\mathcal{C}|_B$ and enforce π on the qubits being forgotten. If B is a merge node, we take the direct product of the affine spaces inherited from C and D,

conjugate it through the gates of $\mathcal{C}|_B$, and then enforce π on the merge ancillas. Using the tools described earlier in this section, producing all of the $\mathcal{A}_B$ takes time $O(||T||_\omega^\omega)$.

If at any point we have $\mathcal{A}_B = \varnothing$ we can stop and conclude that no stabilizers respecting π exist. Suppose instead that we reach the root node and $\mathcal{A}_{\text{Root}} \neq \varnothing$. For any element $(a, b) \in \mathcal{A}_{\text{Root}}$ there exists a longer string $(\alpha, \beta) \in \text{Stab} \cap \Pi$ that agrees with (a, b) on coordinates in $\mathcal{A}_{\text{Root}}$. All that remains is to reconstruct the other coordinates of α and β.

In the second phase we reconstruct (α, β), working our way back down the tree. We start by choosing a uniformly random $(a, b) \in \mathcal{A}_{\text{Root}}$ and for each qubit j in $\mathcal{C}|_{\text{Root}}$, setting the jth tensor element of P_{cor} to $X_j^{a_j} Z_j^{b_j}$. Next, we conjugate $X(a)Z(b)$ backward through the gates of $\mathcal{C}|_{\text{Root}}$, which leaves us with some new $X(c)Z(d) = (\mathcal{C}|_{\text{Root}})^\dagger X(a)Z(b)(\mathcal{C}|_{\text{Root}})$. Let C be the child of the root node. By definition of $\mathcal{A}_C$, there exists $(e, f) \in \mathcal{A}_C$ that agrees with (c, d) on all coordinates of (c, d).[1] Therefore we enforce the pattern (c, d) onto $\mathcal{A}_C$ (appending some $*$-entries to account for any qubits in $\mathcal{C}|_C$ that are not in $\mathcal{C}|_{\text{Root}}$) and choose a random element (e, f). For each qubit j appearing in $\mathcal{C}|_C$ but not $\mathcal{C}|_{\text{Root}}$, set the jth tensor element of P_{cor} to $X_j^{e_j} Z_j^{f_j}$. We then conjugate $X(e)Z(f)$ backward through $\mathcal{C}|_C$, continuing this process. When we reach a merge node and wish to continue down the tree to its children, we split up the coordinates and work on each child's subtree separately. Each qubit is merged or forgotten at some point in $\mathcal{C}$, so eventually every tensor element of P_{cor} will be reconstructed.

All that remains is to show that P_{cor} is sampled uniformly from $\Pi \cap \text{Stab}$. To do so, we first state some general facts about sampling from affine spaces. Consider affine spaces $\mathcal{V}, \mathcal{V}' \subseteq \{0,1\}^n$, and let $m < n$. Let $\mathcal{L} : \mathcal{V}' \to \mathcal{V}$ be an invertible linear map for which $(\mathcal{L}(y))_1 \dots (\mathcal{L}(y))_m = y_1 \dots y_m$ for each $y \in \{0,1\}^n$. In other words, the first m coordinates of $\{0,1\}^n$ are invariant under $\mathcal{L}$. Suppose we first sample the string $x_{m+1} \dots x_n \in \{0,1\}^{n-m}$ uniformly from $\{y_{m+1} \dots y_n : y \in \mathcal{V}\}$. Next, we sample $x_1 \dots x_m$ uniformly from

$$\{y_1 \dots y_m : y \in \mathcal{V}' \text{ and } (\mathcal{L}(y))_j = x_j \forall j > m\}. \tag{2.15}$$

We claim that $x = x_1 \dots x_n$ is a uniformly random sample from $\mathcal{V}$. First, notice that this procedure does in fact return an element of $\mathcal{V}$: Because $\mathcal{L}$ is invertible, there must be some string in $\mathcal{V}'$ that agrees with $x_{m+1} \dots x_n$ when mapped by $\mathcal{L}$.

For a consistent system of linear equations, the size of the solution space is independent of the right hand side. Therefore $|\{y \in \mathcal{V} : y_{m+1} \dots y_n = x_{m+1} \dots x_n\}|$ is independent of $x_{m+1} \dots x_n$. Because $\mathcal{L}$ is an invertible linear transformation, the size of the set in Eq. 2.15 is also independent of $x_{m+1} \dots x_n$. The probability of choosing x is given by $\Pr[x] = \Pr[x_1 \dots x_m | x_{m+1} \dots x_n] \cdot \Pr[x_{m+1} \dots x_n]$. Since $\Pr[x_1 \dots x_m | x_{m+1} \dots x_n]$ is independent of $x_{m+1} \dots x_n$, and $\Pr[x_{m+1} \dots x_n]$ is uniform, x is uniform, which proves the claim.

More generally, suppose we have some finite number of affine spaces $\mathcal{V}^{(k)} \subseteq \{0,1\}^n$ and invertible linear transformations $\mathcal{L}_k : \mathcal{V}_k \to \mathcal{V}_{k-1}$. Suppose we also have a partition $\bigcup_k P_k = [n]$, and that for each k, the subspace of $\{0,1\}^n$ corresponding to the bits in P_k is invariant under $\mathcal{L}_j$ for all $j \leq k$. Consider the following protocol. We start by sampling $x^{(1)} \in \{0,1\}^{|P_1|}$ uniformly from $\{y_{P_1} : y \in \mathcal{V}^{(1)}\}$. Next, we sample $x^{(2)}$ from

$$\{y_{P_2} : y \in \mathcal{V}^{(2)} \text{ and } (\mathcal{L}_2(y))_{P_1} = x^{(1)}\}. \tag{2.16}$$

[1] The string (e, f) may be longer than (c, d).

Continuing in this way, for each $k > 1$ we sample $x^{(k)}$ from

$$\{y_{P_k} : y \in \mathcal{V}^{(k)} \text{ and } (\mathcal{L}_\ell(\mathcal{L}_{\ell+1}(\dots \mathcal{L}_k(y)\dots)))_{P_{\ell-1}} = x^{(\ell-1)} \forall \ell < k\}. \tag{2.17}$$

A similar argument to the previous case shows that x is a uniformly random element of $\mathcal{V}^{(1)}$.

We claim that the second phase of the sampling subroutine can be described in this way. Fix an order on the bags of T so that $j < k$ whenever B_j is an ancestor of B_k. In our case we have $\mathcal{V}^{(k)} := \mathcal{Y}_k$ for each k. Recall that a qubit in $\mathcal{C}$ corresponds to two bits in each $\mathcal{Y}_k$. The P_k are subsets of $[2n_t]$ in which we put both coordinates corresponding to qubit i if and only if i is merged or forgotten in $\mathcal{C}|_{B_k}$. The map $\mathcal{L}_{k-1}$ corresponds to the action of $\mathcal{C}|_{B_k}$ on stabilizers. Although these linear transformations do not act on $2n_t$ bits, they can be seen as acting on all of $\{0,1\}^{2n_t}$ by taking a direct product with the identity map. Because $\mathcal{C}|_{B_j}$ does not act on any qubits in P_k for $j < k$, we get that the bits of $\{0,1\}^{2n_t}$ in P_k are invariant under $\mathcal{L}_j$. Finally, the condition in Eq. 2.17 corresponds to how each new tensor element of P_{cor} is chosen so as to agree with choices made farther up the tree. □

The following claim is not necessary for running the above correction subroutine, but will be useful in Section 2.6.

Claim 21. Let $\pi \in \{0,1,*\}^{2n}$ be given and let $\text{Stab} \subseteq \{0,1\}^{2n}$ be the stabilizer group of $\mathcal{C}\,|+^{n_t}\rangle$. Then $|\text{Stab} \cap \Pi|$ can be computed classically in time $O(||T||_\omega^\omega)$.

Proof. We start by using the sampling subroutine to find any y with $|\,\langle y|\mathcal{C}|+^{n_t}\rangle\,|^2 \neq 0$. Suppose we then run the first part of the correction subroutine, constructing $\mathcal{A}_B$ for each B. In the proof of Lemma 20 we saw that the second phase of the sampling subroutine can be viewed as having a partition $\bigcup_k P_k = [2n_t]$, and sampling an element $x \in \text{Stab} \cap \Pi$ by sampling bits x_{P_k} for $k = 1, \dots, n$ from

$$\{y_{P_k} : y \in \text{Stab} \cap \Pi \text{ and } y_{P_\ell} = x_{P_\ell} \forall \ell < k\}. \tag{2.18}$$

The set from Eq. 2.18 is obtained by enforcing a pattern on some $\mathcal{A}_B$, which yields another affine stabilizer subspace. If this new subspace has the form $\{\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] u + \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] : u \in \{0,1\}^N\}$ for some N, then assuming $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ has full rank,[2] the total number of choices for x_{P_k} is 2^N. The total number of choices for x, and therefore the size $|\text{Stab} \cap \Pi|$ is equal to the product of the number of choices for each x_{P_k}. This algorithm has the same runtime as the correction subroutine, which is $O(||T||_\omega^\omega)$. □

In the case of the graph state simulation problem without postselection, the correction subroutine can be dramatically simplified. We now present a simplified correction subroutine for this case. In this case, $\pi_j^{(X)}$ is in $\{0,1\}$ if and only if j is a merge ancilla, and $\pi^{(Z)} = *^{n_t}$. Because $\bigotimes_v U_v$ does not act on merge ancillas, finding a stabilizer of $\mathcal{C}\,|+^{n_t}\rangle$ that respects π is equivalent to finding a stabilizer of $\mathcal{C}'\,|+^{n_t}\rangle$ that respects π, where we define $\mathcal{C}'$ to be only the CZ and CNOT gates of $\mathcal{C}$. That is, $\mathcal{C}' = ((\bigotimes_v U_v^\dagger) \otimes I^{\otimes n_a})\mathcal{C}$.

[2] This will always be the case in our algorithm, because any time we enforce a pattern, we use Theorem 35 to remove any linearly dependent columns of the generating matrix for the resulting subspace. (See the paragraph containing Eq. 2.13.)

Lemma 22. *For any $\alpha \in \{0,1\}^{n_t}$, a stabilizer of $\mathcal{C}' |+^{n_t}\rangle$ with X-component α can be found in time $O(n + |E|)$.*

Proof. We start by computing $(\mathcal{C}')^\dagger X(\alpha)\mathcal{C}'$. This yields a Pauli proportional to $X(\gamma)Z(\delta)$ for some $\gamma, \delta \in \{0,1\}^{n_t}$. The claim is that $\mathcal{C}'X(\gamma)(\mathcal{C}')^\dagger$ is the desired stabilizer. Because $X(\gamma)$ stabilizes the initial state $|+^{n_t}\rangle$ no matter what γ is, we know $\mathcal{C}'X(\gamma)(\mathcal{C}')^\dagger$ stabilizes $\mathcal{C}' |+^{n_t}\rangle$. We just need to show that $\mathcal{C}'X(\gamma)(\mathcal{C}')^\dagger$ has X-component α.

Fig. 2.7 shows that any stabilizer Z-components that accumulate when conjugating $X(\gamma)$ forward through $\mathcal{C}'$ will have no effect on the X-component of $\mathcal{C}X(\gamma)(\mathcal{C}')^\dagger$. Therefore by choice of γ, the stabilizer $\mathcal{C}X(\gamma)(\mathcal{C}')^\dagger$ has X-component α. The runtime bound follows from the fact that $\mathcal{C}'$ has $O(n+|E|)$ gates through which we must conjugate $X(\alpha)$ and then $X(\gamma)$. □

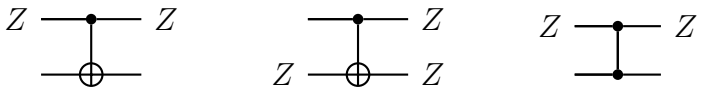


Figure 2.7: Behaviour of stabilizer Z-components when CNOT and CZ gates are applied. *Left:* $\mathrm{CNOT}(Z_1)\mathrm{CNOT}^\dagger = Z_1$. *Centre:* $\mathrm{CNOT}(Z_2)\mathrm{CNOT}^\dagger = Z_1Z_2$. *Right:* $\mathrm{CZ}(Z_1)\mathrm{CZ}^\dagger = Z_1$.

The stabilizer given by Lemma 22 is not uniformly random. By Lemma 19, we can account for this by choosing a random stabilizer of $\bigotimes_v U_v |G\rangle$ and applying it to the data qubits. We can do *that* by choosing a random stabilizer P of $|G\rangle$ and computing $(\bigotimes U_v)P(\bigotimes_v U_v)^\dagger$ in time $O(n)$. It can be shown that $\mathrm{Stab}(|G\rangle)$ is generated by $\{X_v Z_{N(v)} : v \in V\}$. Choosing a random subset of these generators and multiplying them together can be done in time $O(n + |E|)$.

2.5 Correction subroutine example

In this section we work through an example of the correction subroutine described in the proof of Lemma 20. We assume that $P_v = X$ for all v, and thus $U_v = H$ for all v. In Fig. 2.8 we show the input graph, a nice tree decomposition, and $\mathcal{C}$. We also show the state to which each affine stabilizer subspace corresponds. Vertices of the graph are labelled by uppercase letters while qubits of $\mathcal{C}$ are labelled by lowercase letters, with the data qubit and merge ancilla for vertex B being labelled b and b', respectively. Let us take $\pi = (*1**1, *****)$ as the desired pattern, where qubits are ordered $ab'bcd$. That is, we would like P_{cor} to have nonzero X-components acting on qubits b' and d.

We first compute the affine spaces $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5$ as shown in Fig. 2.8. By symmetry, $\mathcal{A}'_1 \cong \mathcal{A}_1$ and $\mathcal{A}'_2 \cong \mathcal{A}_2$, so we omit the calculation of them. The leaf node containing A and B is an introduce node, so we set

$$\mathcal{A}_1 = \{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} x : x \in \mathbb{F}_2^2 \}, \tag{2.19}$$

corresponding to the stabilizer generators X_a and $X_{b'}$ of $|+\rangle_a \otimes |+\rangle_{b'}$. Throughout this example, we will order the rows of matrices first by grouping X- and Z-components together, and then in the same descending order as in π and $\mathcal{C}$. In the matrix above, the first and third rows correspond to qubit a while the second and fourth correspond to b'.

Next we conjugate $\mathcal{A}_1$ by $(H \otimes I)\mathrm{CZ}$. Since qubit a is forgotten, we would in general need to enforce $\pi_a^{(X)}$ and $\pi_a^{(Z)}$, but since $\pi_a^{(X)} = \pi_a^{(Z)} = *$, there is nothing to be done. Therefore we have

$$\mathcal{A}_2 = ((H \otimes I)\mathrm{CZ})\mathcal{A}_1((H \otimes I)\mathrm{CZ})^\dagger = \{\begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} x : x \in \mathbb{F}_2^2\}. \tag{2.20}$$

To compute $\mathcal{A}_3$, we first remove the rows of the matrix in $\mathcal{A}_2$ corresponding to a (i.e. the first and third) to obtain $\{\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)x : x \in \mathbb{F}_2^2\}$. By symmetry, we would obtain the same set

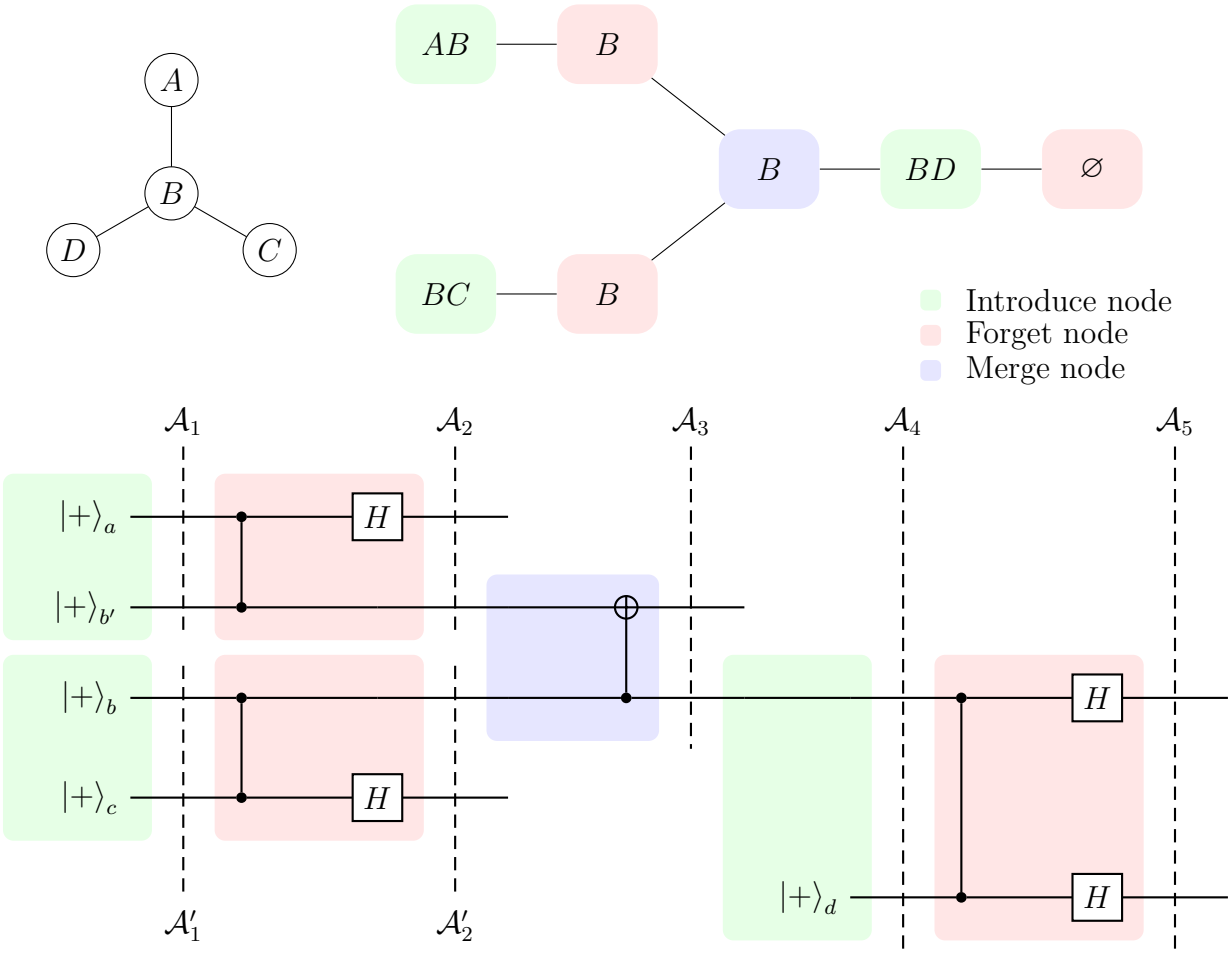


Figure 2.8: *Top:* A graph and a nice tree decomposition. *Bottom:* The resulting circuit $\mathcal{C}$ in the case where $P_v = X$ for all v, which implies that $U_v = H$ for all v. Each affine stabilizer subspace $\mathcal{A}_i$ or $\mathcal{A}'_i$ formed during the correction subroutine is associated with the bag immediately to the left of its dashed line. E.g. $\mathcal{A}_3$ is associated with the lone merge node.

after removing qubit c from $\mathcal{A}_2'$. We then take the direct product

$$\{\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)x : x \in \mathbb{F}_2^2\} \oplus \{\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)x : x \in \mathbb{F}_2^2\} \cong \{\begin{pmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{pmatrix}x : x \in \mathbb{F}_2^4\}. \tag{2.21}$$

We have rearranged the right hand side matrix so that the X-components form the first two rows and the Z-components the last two. Next, we apply the CNOT gate to obtain

$$\{\begin{pmatrix}0&1&0&1\\0&0&0&1\\1&0&0&0\\1&0&1&0\end{pmatrix}x : x \in \mathbb{F}_2^4\}. \tag{2.22}$$

Finally, we must enforce the pattern on the merge ancilla b'. Since $\pi_{b'}^{(X)} = 1$, this means finding the subset of elements in Eq. 2.22 that have the form $(1, *, *, *)^T$. Basic linear algebra tells us that x must take the form $x = e_4 + y_1 e_1 + y_2(e_2 + e_4) + y_3 e_3$, where e_j is the usual standard basis vector and $y_1, y_2, y_3 \in \{0, 1\}$ are free. The desired subset therefore is

$$\mathcal{A}_3 = \{\begin{pmatrix}0&1&0&1\\0&0&0&1\\1&0&0&0\\1&0&1&0\end{pmatrix}(\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&0\end{pmatrix}y + \begin{pmatrix}0\\0\\0\\1\end{pmatrix}) : y \in \mathbb{F}_2^3\} = \{\begin{pmatrix}0&0&0\\0&1&0\\1&0&0\\1&0&1\end{pmatrix}y + \begin{pmatrix}1\\1\\0\\0\end{pmatrix} : y \in \mathbb{F}_2^3\}. \tag{2.23}$$

We then restrict $\mathcal{A}_3$ to coordinates corresponding to b to obtain $\{\left(\begin{smallmatrix}0&1&0\\1&0&1\end{smallmatrix}\right)x + \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) : x \in \mathbb{F}_2^3\}$. Since the matrix has more columns than rows, we search for a maximum linearly independent subset of columns. In general we would apply Theorem 35a, but by inspection, we can simply remove the third column to get $\{\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)x + \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) : x \in \mathbb{F}_2^2\}$. Taking a direct product with the affine space $\mathcal{I}$ for the stabilizer generator of $|+\rangle_d$ gives us

$$\mathcal{A}_4 = \{\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\\0&0&0\end{pmatrix}x + \begin{pmatrix}1\\0\\0\\0\end{pmatrix} : x \in \mathbb{F}_2^3\}. \tag{2.24}$$

To compute $\mathcal{A}_5$, we conjugate the affine space by the CZ and Hadamard gates to obtain

$$\{\begin{pmatrix}1&0&1\\0&1&0\\0&1&0\\0&0&1\end{pmatrix}x + \begin{pmatrix}0\\1\\1\\0\end{pmatrix} : x \in \mathbb{F}_2^3\}, \tag{2.25}$$

and since $\pi_d^{(X)} = 1$, we search for all elements in this space of the form $(*, 1, *, *)^T$. Using the same approach as in the computation of $\mathcal{A}_3$ we arrive at

$$\mathcal{A}_5 = \{\begin{pmatrix}1&1\\0&0\\0&0\\0&1\end{pmatrix}y + \begin{pmatrix}0\\1\\1\\0\end{pmatrix} : y \in \mathbb{F}_2^2\}. \tag{2.26}$$

We are now ready to construct P_{cor}. We first choose a random element from $\mathcal{A}_5$, say, $(0, 1, 1, 0)^T$. This fixes two tensor elements of the Pauli correction: $P_{\text{cor}} = ? \otimes ? \otimes Z \otimes ? \otimes X$. We will move down the tree to fill in the remaining question marks.

Since $((H \otimes H)CZ)^\dagger Z \otimes X((H \otimes H)CZ) = X \otimes I$, pushing the chosen element of $\mathcal{A}_5$ backward through the H and CZ gates gives us $(1, 0, 0, 0)^T \in \mathcal{A}_4$. We then remove qubit d to obtain $(1, 0)^T$ and select a random element in $\mathcal{A}_3$ of the form $(*, 1, *, 0)^T$, say, $(1, 1, 0, 0)^T$. Since b' is measured immediately after the merge node, we can fill in another tensor element in the Pauli correction: $P_{\text{cor}} = ? \otimes X \otimes Z \otimes ? \otimes X$.

Pushing $(1,1,0,0)^T$ back through the CNOT gate gives $(0,1,0,0)^T$. Removing b gives us $(0,0)^T$, so we search for an element of $\mathcal{A}_2$ of the form $(*,0,*,0)^T$. The only such element is $(0,0,0,0)^T$, so we now have $P_{\text{cor}} = I \otimes X \otimes Z \otimes ? \otimes X$. Similarly, we can reconstruct the tensor element corresponding to qubit c and find that $P_{\text{cor}} = I \otimes X \otimes Z \otimes X \otimes X$ is a valid choice; indeed, we have $(01011, 00100) \in \Pi$. We do not need to use $\mathcal{A}_1$ or $\mathcal{A}_1'$ as we now have all tensor elements of P_{cor}.

2.6 Variants of the graph state simulation problem

Our algorithm for the graph state simulation problem can be adapted to solve the following two related problems.

Strong Graph State Simulation Problem. Let $G, \{P_v\}_v, \bigotimes_v U_v, \mathcal{P}, \mathcal{S}$, and m be as in the graph state simulation problem and let $z \in \{0,1\}^{|\mathcal{S}|}$ also be given. The task is to compute

$$p(z) = \frac{1}{p_{\mathcal{P}}(m)} |\langle z|_{\mathcal{S}} \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle|^2. \tag{2.27}$$

Phase-Sensitive Graph State Simulation Problem. Let $G, \{P_v\}_v, \bigotimes_v U_v, \mathcal{P}, \mathcal{S}$, and m be as in the graph state simulation problem and let $z \in \{0,1\}^{|\mathcal{S}|}$ also be given. The task is to compute

$$\frac{1}{\sqrt{p_{\mathcal{P}}(m)}} \langle z|_{\mathcal{S}} \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle . \tag{2.28}$$

Theorem 23. *Given a nice tree decomposition T of G, the strong graph state simulation problem on G can be solved classically in time $O(||T||_\omega^\omega)$.*

Theorem 24. *Given a nice tree decomposition T of G, the phase-sensitive graph state simulation problem on G can be solved classically in time $O(||T||_3^3)$.*

When G is planar we can use Theorem 13 to construct a nice tree decomposition for G. This gives the following corollaries.

Theorem 25. *Let G be a planar graph. Then any instance of the strong graph state simulation problem on G can be solved in time $\widetilde{O}(n^{\omega/2})$.*

Theorem 26. *Let G be a planar graph. Then any instance of the phase-sensitive graph state simulation problem on G can be solved in time $\widetilde{O}(n^{3/2})$.*

Proof of Theorem 23. We will compute $|\langle z|_{\mathcal{S}} \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle|^2$ and $p_{\mathcal{P}}(m)$ separately. Let $\mathcal{C}$ be as defined in Section 2.3. Let $y = zm0^{n_a} \in \{0,1\}^{n_t}$ be the concatenation of z, m, and 0^{n_a}. We will compute $|\langle y|\mathcal{C}|+^{n_t}\rangle|^2$ using Lemma 17 in time $O(||T||_\omega^\omega)$. By Lemma 15, we have

$$|\langle z|_{\mathcal{S}} \langle m|_{\mathcal{P}} \bigotimes_v U_v |G\rangle|^2 = 2^{n_a} |\langle y|\mathcal{C}|+^{n_t}\rangle|^2. \tag{2.29}$$

Now we compute $p_{\mathcal{P}}(m)$. Let $\alpha \in \{0,1,*\}^{n_t}$ have indices associated with qubits of $\mathcal{C}$, and suppose that for all $v \in \mathcal{P}$ we have $\alpha_{j(v)} = 0$, for $v \in \mathcal{S}$ we have $\alpha_{j(v)} = *$, and for merge

ancillas $\alpha_j = 0$. Let $\Pi \subseteq \{0,1\}^{2n_t}$ be the set of strings respecting the pattern $(\alpha, *^{n_t})$, and let $\mathrm{Stab} \subseteq \{0,1\}^{2n}$ represent the stabilizer group of $\mathcal{C}\,|+^{n_t}\rangle$ with phases ignored. Then

$$2^{n_a}\frac{|\Pi \cap \mathrm{Stab}|}{2^{n_t}} = 2^{n_a}\sum_{s\in\{0,1\}^{\mathcal{S}}} |\langle s|\langle m|\langle 0^{n_a}|\,\mathcal{C}\,|+^{n_a}\rangle|^2 \qquad \text{by Lemma 19} \tag{2.30}$$

$$= \sum_{s\in\{0,1\}^{\mathcal{S}}} |\langle s|_{\mathcal{S}}\langle m|_{\mathcal{P}} \bigotimes_v U_v\,|G\rangle|^2 \qquad \text{by Lemma 15} \tag{2.31}$$

$$= p_{\mathcal{P}}(m). \tag{2.32}$$

By Claim 21 we can compute $|\Pi \cap \mathrm{Stab}|$ in time $O(||T||_\omega^\omega)$, which completes the proof. □

The proof of Theorem 24 is identical, except we use Lemma 18 to get $\langle y|\mathcal{C}|+^{n_t}\rangle$ in time $\widetilde{O}(n^{3/2})$, and use $\sqrt{p_{\mathcal{P}}(m)}$ instead of $p_{\mathcal{P}}(m)$.

2.7 Applications of graph state simulation

In this section we describe some problems that can be reduced to graph state simulation. Our main applications are in the simulation of a restricted family of Clifford circuits. We also consider analogous problems involving Clifford tensor networks. The latter problems can be shown to be generalizations of the former, however we present the first applications separately because (1) Clifford circuit simulation is a very common task; (2) our algorithms are simpler when we restrict our attention to Clifford circuits; and (3) we will explicitly use one of the Clifford simulation algorithms in Chapter 3.

We start by proving Theorems 10a, 10b, and 10c, restated here. Fig. 2.9 gives an example of a Clifford circuit whose two-qubit gates act along the edges of a planar graph.

Theorem 10. *Let C be an n-qubit depth-d Clifford circuit whose two-qubit gates act along the edges of a planar graph G. There exist classical algorithms for the following tasks.*

a. Sampling z from the output distribution $\Pr[z] = |\langle z|C|0^n\rangle|^2$ in time $\widetilde{O}(n^{\omega/2}d^\omega)$.

b. Given $z \in \{0,1\}^n$, computing $|\langle z|C|0^n\rangle|^2$ in time $\widetilde{O}(n^{\omega/2}d^\omega)$.

c. Given $z \in \{0,1\}^n$, computing $\langle z|C|0^n\rangle$ in time $\widetilde{O}(n^{3/2}d^3)$.

These three theorems use reductions to the three versions of the graph state simulation problem. We will sometimes need to solve these problems on graphs that are nonplanar, but are in some sense close to being planar. Consider graphs $G' = (V', E')$ and $G = (V, E)$, and a map $f : V(G') \to V(G)$. For $r > 0$ we say that f is an *r-coarse-graining* if (1) for each $u'v' \in E'$ we have either $f(u') = f(v')$ or $f(u')f(v') \in E$; and (2) $|\{u' \in E' : f(u') = v\}| \leq r$ for each $v \in V$. Fig. 2.10 gives an example of a 5-coarse-graining.

Lemma 27. *Let $G' = (V', E')$ and $G = (V, E)$ be graphs and $f : G' \to G$ an r-coarse-graining. Given a nice tree decomposition T for G, a nice tree decomposition T' for G' can be found in time $O(|V|)$ such that for all $c > 0$ we have $||T'||_c^c = O(r^c||T||_c^c)$.*

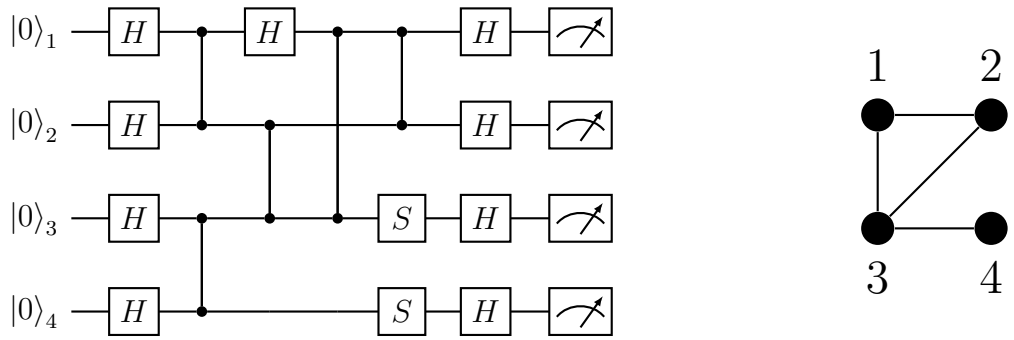


Figure 2.9: The CZ gates in the circuit on the left act along edges of the planar graph on the right.

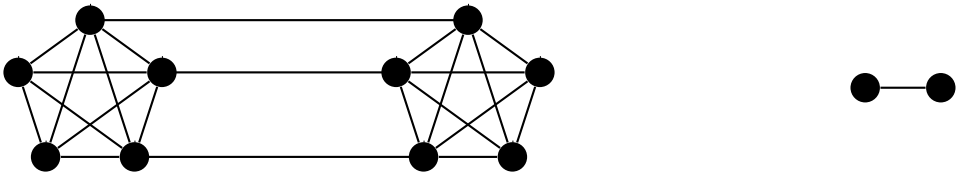


Figure 2.10: The graph on the left admits a 5-coarse-graining to the one on the right.

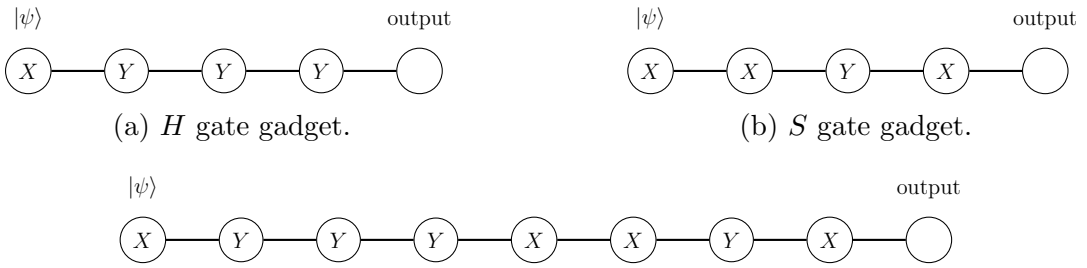
Proof. For each bag B_i in T, define a bag B'_i of T' as $B'_i = \{v' \in V' : f(v') \in B_i\}$. Arranging the nodes B'_i in the same way as the nodes B_i (i.e. B'_i, B'_j share an edge in T' if and only if B_i, B_j share an edge in T) will yield a valid nice tree decomposition. Because $|B'_i| \leq r|B_i|$ for each i, we have $||T'||_c^c = O(r^c||T||_c^c)$. □

Proofs in this section will use measurement-based quantum computing gadgets [28, 29]. Any Clifford operation on a single-qubit state $|\psi\rangle$ can be implemented, up to some Pauli error, by entangling $|\psi\rangle$ with the four-qubit graph state shown in Fig. 2.11 with a CZ gate and measuring all but one qubit in some Pauli bases. Fig. 2.11 also shows the Pauli bases needed for the Hadamard and S gate gadgets. If the measurement results are, from left to right, r_0, r_1, r_2, r_3, then the unmeasured qubit is left in state $X^{r_1+r_3}Z^{r_0+r_2}H|\psi\rangle$ or $X^{r_1+r_3}Z^{r_0+r_2}S|\psi\rangle$ [21]. For example, let $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{v_1v_2, v_2v_3, v_3v_4\}$. Then Fig. 2.11a indicates that

$$(\langle r_0|H \otimes \langle r_1|HS^\dagger \otimes \langle r_2|HS^\dagger \otimes \langle r_3|HS^\dagger \otimes I)\mathrm{CZ}_{\psi,v_1}|\psi\rangle|G\rangle \propto X^{r_1+r_3}Z^{r_0+r_2}H|\psi\rangle\,. \quad (2.33)$$

We have used the fact that H and $HS^\dagger$ map the computational basis to the X- and Y-bases. The output qubit of one gadget can also be used as the input qubit of another gadget to apply products of single-qubit gates, as shown in Fig. 2.11c. By linearity, these gadgets can be used when $|\psi\rangle$ has multiple qubits by applying the CZ gate to the appropriate qubit of $|\psi\rangle$.

Proof of Theorem 10a. An example of the reduction described in this proof is shown in Fig. 2.12. We will consider the circuit $C' = CH^{\otimes n}$ on input $|+^n\rangle$, as for any z we have $\langle z|C'|+^n\rangle = \langle z|C|0^n\rangle$. Each single-qubit gate is replaced by its measurement-based gadget. To apply the gates in succession we take the input qubit of one gadget to be the output qubit



(a) H gate gadget. (b) S gate gadget.

(c) SH gate gadget.

Figure 2.11: Vertices represent qubits and edges represent CZ gates. All qubits other than $|\psi\rangle$ begin in the state $|+\rangle$. Measuring each qubit of Fig. 2.11a or 2.11b in the designated Pauli bases leaves the output qubit in state $QH|\psi\rangle$ or $QS|\psi\rangle$, where Q is a Pauli error determined by measurement outcomes. Gadgets can be joined together. Measuring the qubits of Fig. 2.11c applies the gate $Q'SQH$, where Q, Q' are determined by the measurement outcomes.

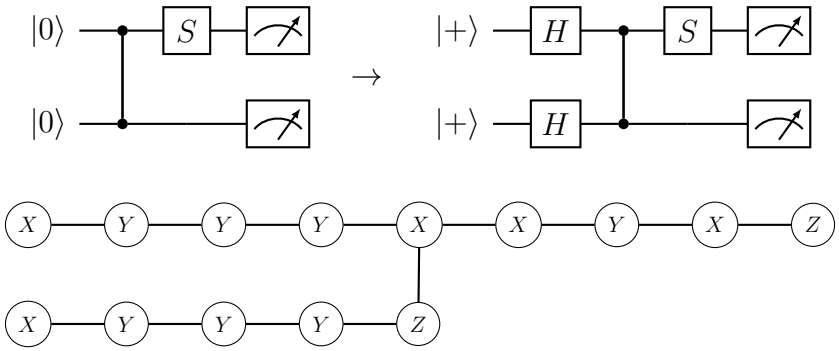


Figure 2.12: *Top:* Replacing C with C'. *Bottom:* Replacing each single-qubit gate of C' with one of the gadgets from Fig. 2.11 to form $|G'\rangle$.

of the previous gadget on its wire. (The first gadget on each wire takes $|+\rangle$ as its input.) The CZ gates of C' are applied between the output qubits of the two gadgets preceding that CZ gate. This procedure leaves us with some graph state $|G'\rangle$. The graph G' has $n+4s = O(nd)$ vertices, where s is the number of single-qubit gates in C'.

All but n qubits of $|G'\rangle$ have Pauli bases in which they are to be measured as part of a gadget. The final n qubits correspond to the output qubits of the final gadget on each wire. We will simulate measurements on all qubits of $|G\rangle$ in the designated Pauli bases, with these last n qubits measured in the computational basis. This is an instance of the weak graph simulation problem on G' without postselection.

To see how a solution can be used to sample from C, let $y \in \{0,1\}^n$ be the measurement results on the final n qubits of $|G'\rangle$. The remaining results indicate the Pauli error $Q^{(j)}$ introduced at each single-qubit gate $U^{(j)}$ of C'. The string y is sampled from $\Pr[y] = |\langle y|C'_{\text{err}}|+^n\rangle|^2$, where C'_{err} is obtained by replacing each single-qubit gate $U^{(j)}$ with $Q^{(j)}U^{(j)}$. Using the results of Section 1.1, we can conjugate each error through to the end of C' in time $O(nd)$ to get a single Pauli error Q with $\langle y|C'_{err}|+^n\rangle = \langle y|QC'|+^n\rangle$. Then for the

$a, b \in \{0,1\}^n$ satisfying $Q \propto X(a)Z(b)$, we have

$$|\langle y|C'_{\text{err}}|+^n\rangle|^2 = |\langle y|QC'|+^n\rangle|^2 = |\langle y \oplus a|C'|+^n\rangle|^2 = |\langle y \oplus a|C|0^n\rangle|^2. \tag{2.34}$$

Therefore $y \oplus a$ is sampled from the output distribution of C.

Finally, we must show that the graph state simulation problem on G' can be solved in time $\widetilde{O}(n^{\omega/2}d^\omega)$. Let $f : G' \to G$ be an $O(d)$-coarse-graining defined as follows: For each vertex $v \in V(G')$ that is part of a gadget acting on wire j, take $f(v) = j$. For each vertex v that corresponds to the input qubit along a wire j, set $f(v) = j$. Because each gadget has constant size and there are at most d gadgets on each wire, f is an $O(d)$-coarse-graining. Next we use Theorem 13 to get a nice tree decomposition T for G with $||T||_\omega^\omega = \widetilde{O}(n^{\omega/2})$, and then use Lemma 27 to transform T into a nice tree decomposition T' for G' with $||T'||_\omega^\omega = \widetilde{O}(n^{\omega/2}d^\omega)$. By Theorem 14 we can solve the graph state simulation problem on G' in time $O(||T'||) = \widetilde{O}(n^{\omega/2}d^\omega)$. □

The proof of Theorems 10b and 10c are very similar, so we will only highlight the differences.

Proof of Theorems 10b and 10c. We will form C' and replace each gate with its measurement-based quantum computing gadget to form $|G'\rangle$ exactly as in the previous proof. For each gadget, if all measurement outcomes are postselected to be 0, the gadget is implemented with zero error. We will solve the strong graph state simulation problem *with* postselection to force all gadgets to be implemented with zero error. Let n_g be the number of vertices of G' that do not correspond to the output of C', and let $\mathcal{P}$ be the set of all such vertices. Let $\mathcal{S}$ be the remaining n vertices. For example, in Fig. 2.12 we have $n_g = 12$ and $\mathcal{S}$ equal to the two vertices labelled with "Z". As in the proof of Theorem 10a, each vertex of G' is labelled with a Pauli basis P_v (either X or Y for gadget qubits, and Z for output qubits).

To prove Theorem 10b, we solve the strong graph state simulation problem on G' with $\{P_v\}_v, \mathcal{P}$ and $\mathcal{S}$ as above, $m = 0^{n_g}$, and z as in the statement of the theorem using Theorem 23. This gives the value $|\langle z|C|0^n\rangle|^2$. Constructing f and forming the tree decomposition T' as in the proof of Theorem 10a gives the runtime bound. To prove Theorem 10c, we solve the phase-sensitive graph state simulation problem on G' with the parameters listed above using Theorem 24, which gives $\langle z|C|0^n\rangle$. Taking f and T' again as described above gives the runtime bound. □

In Chapter 3 we will wish to compute $\langle z|C|0^n\rangle$ in the case where the two-qubit gates of C act along a graph G which is not planar, and for which we might not have an r-coarse-graining for small r. The above proof of Theorem 10c can be modified to yield the following result, which we will not prove. The only modification necessary is to use T directly, rather than using Theorem 13 to compute a nice tree decomposition.

Theorem 28. *Let C be a depth-d Clifford circuit with n qubits, whose two-qubit gates act along the edges of a graph G. Let $z \in \{0,1\}^n$ and a nice tree decomposition T of G be given. Then a classical algorithm can compute $\langle z|C|0^n\rangle$ in time $O(||T||_3^3 d^3)$.*

We now introduce tensor networks, which generalize quantum circuits to allow for nonunitary operations. As a warm-up, consider the task of computing the output probability

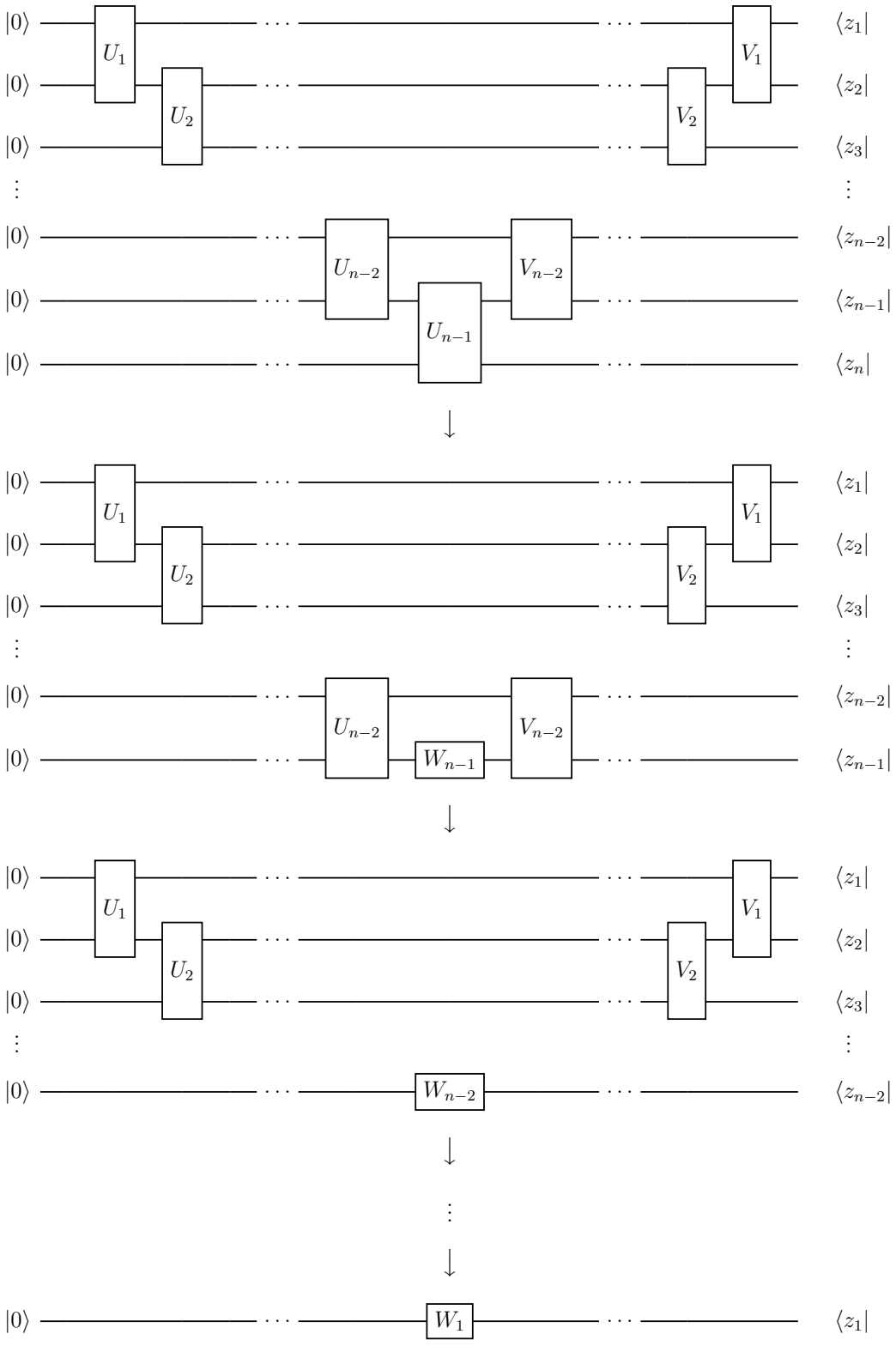


Figure 2.13: Simulation of the circuit at the top can be done in time $O(n)$ using the ordering shown here.

$\Pr[z] \coloneqq |\langle z|U_1 \dots U_{n-1}V_{n-2} \dots V_1|0^n\rangle|^2$ in the circuit shown at the top of Fig. 2.13. We will describe an algorithm that views the circuit as a tensor network, leading to an improved runtime.

A naïve approach would be to initialize a classical representation of $|0^n\rangle$, simulate the applications of each gate one at a time, and then read off the entry of the state vector given by z. Merely writing down the state vector takes time $\Omega(2^n)$, so this algorithm is inefficient. A slightly better approach would be to again proceed from left to right through the circuit, but only initialize qubits when they're needed: First, $|00\rangle$ is initialized and U_1 is applied, then a third qubit is initialized to $|0\rangle$ and the application of U_2 is simulated, and so on. This algorithm fares slightly better than the first one, but once U_{n-1} is applied, we must store all 2^n amplitudes of the state vector.

An improved algorithm shown in Fig. 2.13 yields a better runtime by not simulating the circuit from left to right. First, we replace U_{n-1} with $W_{n-1} \coloneqq (I \otimes \langle z_n|)U_{n-1}(I \otimes |0\rangle)$. Simulation of the resulting $(n-1)$-qubit circuit (which may be nonunitary) will still yield $\Pr[z]$. We then replace $U_{n-2}(I \otimes W_{n-1})V_{n-2}$ with

$$W_{n-2} \coloneqq (I \otimes \langle z_{n-1}|)U_{n-2}(I \otimes W_{n-1})V_{n-2}(I \otimes |0\rangle), \tag{2.35}$$

leaving an $(n-2)$-qubit circuit whose simulation yields $\Pr[z]$. Repeating this process eventually leaves $\Pr[z] = |\langle 0|W_1|z_1\rangle|^2$, which can be computed directly. Since we only ever need to operate on 4×4 matrices, and there are $O(n)$ steps, this algorithm has runtime $O(n)$. The idea of performing a simulation by choosing a favourable ordering in which to carry out operations characterizes tensor network contraction algorithms, which we describe now. The exposition here follows that of [25].

Definition 29. A rank-r tensor M is an array of 2^r complex numbers.[3] Each entry is labelled by r indices $i_1, \dots, i_r \in \{0, 1\}$. We abuse notation slightly and write $M = M_{i_1,\dots,i_r}$ both to make the indices explicit and also to refer to the $(i_1, \dots, i_r)$ entry.

If $M_{i_1,\dots,i_n,j_1,\dots,j_m}$ and $N_{i_1,\dots,i_n,k_1,\dots,k_\ell}$ are tensors sharing the indices $i_1, \dots, i_n$, the *contraction* operation defines a new tensor $(MN)_{j_1,\dots,j_m,k_1,\dots,k_\ell}$ with entries given by

$$(MN)_{j_1,\dots,j_m,k_1,\dots,k_\ell} = \sum_{i_1,\dots,i_n=0}^{1} M_{i_1,\dots,i_n,j_1,\dots,j_m} N_{i_1,\dots,i_n,k_1,\dots,k_\ell}. \tag{2.36}$$

Definition 30. A tensor network is a set of tensors in which no index is shared by more than two elementary tensors.

Contracting every shared index of a tensor network yields a tensor. For this reason, if $\mathcal{T}$ is a tensor network we will often write $\mathcal{T}$ to also represent the tensor that results when all shared indices are contracted. Tensors and tensor networks can be illustrated graphically. In Fig. 2.14 each tensor is depicted by a central node with r "legs," corresponding to indices. Contraction of a shared index is represented by joining the legs corresponding to that index and then contracting the edge.

[3] Most definitions allow the base 2 to be replaced with any integer $d \geq 2$.

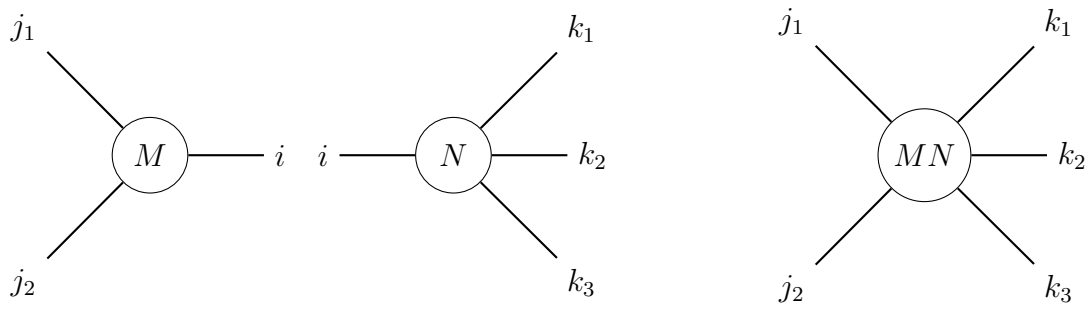


Figure 2.14: *Left:* The rank-3 and rank-4 tensors $M_{j_1,j_2,i}$ and $N_{k_1,k_2,k_3,i}$ form a tensor network. *Right:* The rank-5 tensor $(MN)_{j_1,j_2,k_1,k_2,k_3}$ formed by contracting index i.

The time needed to contract a tensor network using Eq. 2.36 repeatedly is upper bounded by $2^{O(\xi)}s$, where s is the number of tensors in the network and ξ is the maximum rank of any tensor encountered during the contraction process. This sheds some light on why the improved algorithm for the warm-up problem fares better than the naïve ones. Every quantum circuit can be thought of as a tensor network by viewing each m-qubit gate M as a rank-$2m$ tensor with entries $M_{i_1,\ldots,i_m,j_1,\ldots,j_m} = \langle i|M|j\rangle$, wire segments as shared indices, and k-qubit input or output states as rank-k tensors. The three different algorithms for solving the warm-up problem all consisted of different orders for contracting the tensor network described by $\langle z|U_1 \ldots U_{n-1}V_{n-2}\ldots V_1|0^n\rangle$. In the third algorithm, we chose an ordering such that ξ was constant.

Markov and Shi [25] give an algorithm that takes as input a tensor network and outputs an ordering of indices that will keep ξ relatively small. For a tensor network $\mathcal{T}$ made up of tensors $T^{(1)},\ldots,T^{(s)}$, let $G(\mathcal{T})$ be the graph vertex set $[s]$ and an edge ij for each $T^{(i)},T^{(j)}$ that share one or more legs. We say that $G(\mathcal{T})$ is the *underlying graph* of $\mathcal{T}$. Markov and Shi show that the minimum possible ξ is related to the treewidth of $G(\mathcal{T})$ and give a contraction ordering that leads to a relatively small ξ. Many other works [23, 17, 13] consider various techniques for finding a favourable contraction ordering.

Tensors can also be thought of as quantum states. If $M_{i_1,\ldots,i_m}$ is a tensor, we can consider the unnormalized state

$$|M\rangle = \sum_{i_1,\ldots,i_m=0}^{1} M_{i_1,\ldots,i_m}\,|i_1\ldots i_m\rangle\,. \tag{2.37}$$

In this picture, contraction of two tensors can be seen as applying the map $\langle 00| + \langle 11|$ to the shared index in their corresponding states. For example, if $M_{i,j}$ and $N_{i,k}$ are tensors then

$$\sum_{j,k=0}^{1}(MN)_{j,k}\,|jk\rangle = \sum_{j,k=0}^{1}\sum_{i=0}^{1} M_{i,j}N_{i,k}\,|jk\rangle \tag{2.38}$$

$$= \langle 00|_{13} + \langle 11|_{13}\left(\sum_{i,j=0}^{1} M_{i,j}\,|ij\rangle\right)\left(\sum_{i',k=0}^{1} N_{i',k}\,|i'k\rangle\right). \tag{2.39}$$

Let $M_{i_1,\ldots,i_n}$ be a rank-n tensor. We say that $M_{i_1,\ldots,i_n}$ is a *Clifford tensor* if

$$\sum_{i_1,\ldots,i_n=0}^{1} M_{i_1,\ldots,i_n} |i_1 \ldots i_n\rangle \tag{2.40}$$

is proportional to a stabilizer state. A Clifford tensor network is a tensor network made up exclusively of Clifford tensors.

Theorem 31. *Let $\mathcal{T}$ be a Clifford tensor network made up of n tensors $T^{(1)}, \ldots, T^{(n)}$, each of which has rank at most r, let $G(\mathcal{T})$ be the underlying graph, and let $N = \text{rank}(\mathcal{T})$. Then there exist classical algorithms for the following tasks.*

a. Sampling a uniformly random element of $\mathcal{T}$ in time $\widetilde{O}(n^{\omega/2} r^{\omega})$.

b. Given $z \in \{0,1\}^N$, compute $|\mathcal{T}_{z_1,\ldots,z_N}|^2$ in time $\widetilde{O}(n^{\omega/2} r^{\omega})$.

c. Given $z \in \{0,1\}^N$, compute $\mathcal{T}_{z_1,\ldots,z_N}$ in time $\widetilde{O}(n^{3/2} r^3)$.

Similarly to Theorem 10, we will describe a reduction to the three versions of the graph state simulation problem using measurement-based quantum computing gadgets. We will prove Theorem 31a in full and then describe the small modifications needed for Theorems 31b and 31c.

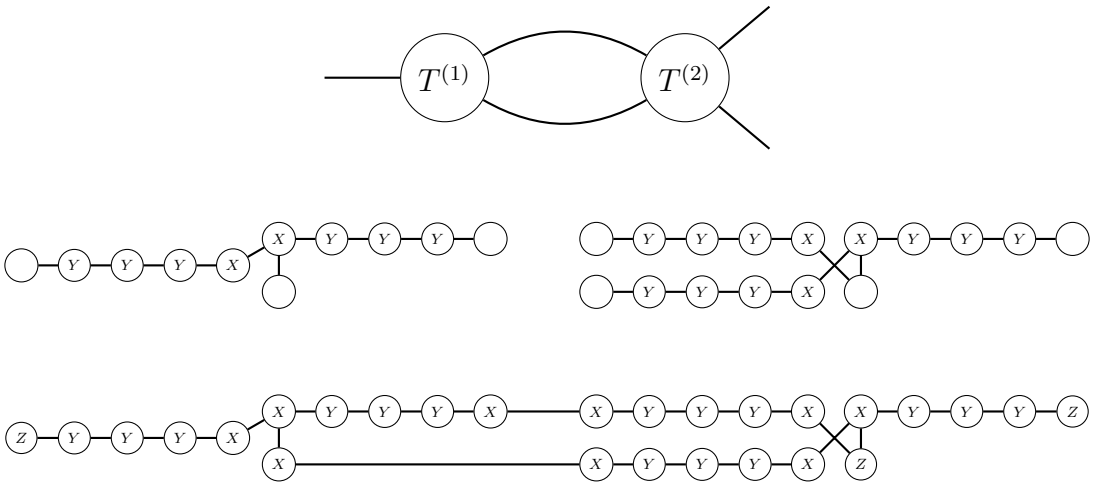


Figure 2.15: Example of the reduction described in the proof of Theorem 31a. *Top:* A tensor network with two shared legs. *Centre:* Each elementary tensor is mapped to a graph state with gadgetized local Cliffords. In this example we assume that all local Cliffords are either Hadamard (see Fig. 2.11) or identity. *Bottom:* Shared legs are contracted by applying $\langle{+}{+}| \, \mathrm{CZ}$. The remaining three output qubits are measured in the computational basis.

Proof of Theorem 31a. We will assume that each $|T^{(k)}\rangle$ is a stabilizer state and not just proportional to one. If this is not the case then $\mathcal{T}$ is just rescaled by a constant factor. We give an example of the reduction to graph state simulation described by this proof in

Fig. 2.15. Our algorithm requires that each $T^{(k)}$ is given to us as $|T^{(k)}\rangle$, expressed as a graph state with local Clifford operators (see Section 1.4). Before performing any contraction we have

$$|T^{(1)}\rangle \otimes \ldots \otimes |T^{(n)}\rangle = (U_1 \otimes \ldots \otimes U_m) |G_1\rangle \otimes \ldots \otimes |G_n\rangle . \tag{2.41}$$

For each G_k, we will replace every U_j that acts on $|G_k\rangle$ with a gadget like the ones shown in Fig. 2.11. If $U_j \in \{H, S\}$ we can use the H and S gadgets directly. Otherwise, U_j can be expressed as a finite product of H and S, so we can implement U_j by linking together $O(1)$ gadgets (see Fig. 2.11c). If $U_j = I$ then we do not add any gadgets. Let us call the resulting graph state $|G'_k\rangle$. Because we add $O(1)$ gadgets for each vertex of G_k, and each gadget has $O(1)$ vertices, G'_k has $O(\text{rank}(T^{(k)}))$ vertices.

Recall that if measurement outcomes on the H or S gate gadgets are given by r_0, r_1, r_2, r_3, then the gadget implements $X^{r_1+r_3} Z^{r_0+r_2} H$ or $X^{r_1+r_3} Z^{r_2+r_4} S$. For each local unitary U_j, let $|\phi^{(j)}\rangle$ be the stabilizer state with the following property: When the qubits in the gadget for U_j are projected onto $|\phi^{(j)}\rangle$, the gate U_j is applied with no error. For example, if $U_j = S$, then Fig. 2.11b shows that we may take $|\phi^{(j)}\rangle = |+\rangle \otimes |+\rangle \otimes \frac{|0\rangle+i|1\rangle}{\sqrt{2}} \otimes |+\rangle$, and if $U_j = H$ we have $|\phi^{(j)}\rangle = |+\rangle \otimes \frac{|0\rangle+i|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle+i|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle+i|1\rangle}{\sqrt{2}}$. Since all gadgets are created by linking together the H and S gadgets, $|\phi^{(j)}\rangle$ always exists. Then

$$|T^{(1)}\rangle \otimes \ldots \otimes |T^{(n)}\rangle \propto \left(\prod_j \langle \phi^{(j)}| \right) |G'_1\rangle \otimes \ldots \otimes |G'_n\rangle . \tag{2.42}$$

(The $\propto$ sign hides a normalization factor, which is some power of $\sqrt{2}$.)

All we've done so far is rewrite the tensor network pre-contraction. We will now contract all shared indices. For every shared index i in $\mathcal{T}$ there are two graphs G_a, G_b that correspond to the two tensors $T^{(a)}, T^{(b)}$ sharing i. Both G_a and G_b have a vertex corresponding to the index i. Each of these two vertices have gadgets associated with them arising from the process above in which we replaced each U_j with a gadget. Let $\ell(a), \ell(b)$ be the vertices of G'_a, G'_b corresponding to the output qubits of these two gadgets. For each shared index i, we can find $\ell(a), \ell(b)$ as above. Let $\mathcal{L}$ be the set of all such pairs $\{\ell(a), \ell(b)\}$.

To contract the index i we must apply $(\langle 00| + \langle 11|)_{\ell(a),\ell(b)} \propto (\langle ++| \, \text{CZ}(I \otimes H))_{\ell(a),\ell(b)}$. Let us ignore the local operators $I \otimes H$ because they could have been incorporated into the U_j earlier. Then

$$|\mathcal{T}\rangle \propto \left(\prod_j \langle \phi^{(j)}| \right) \left(\prod_{\ell \in \mathcal{L}} \langle ++|_\ell \, \text{CZ}_\ell \right) |G'_1\rangle \otimes \ldots \otimes |G'_n\rangle . \tag{2.43}$$

Here we applied $\prod_{\ell \in \mathcal{L}} \langle ++|_\ell \, \text{CZ}_\ell$ to Eq. 2.42 and used the fact that it acts on different qubits than $\prod_j \langle \phi^{(j)}|$, so they commute.

Because each $|\phi^{(j)}\rangle$ is a tensor product of single-qubit stabilizer states, Eq. 2.43 is nothing more than the projection of a graph state $|\mathcal{G}\rangle$ onto a product of single-qubit stabilizer states. Choosing a uniformly random element of $\mathcal{T}$, which is equivalent to sampling z from $\Pr[z] = |\langle z|\mathcal{T}\rangle|^2$, is therefore also equivalent to solving an instance of the graph state simulation problem with postselection. We now describe this instance.

To form the graph $\mathcal{G}$, form $\cup_k G'_k$, and then add the edge $\ell(a)\ell(b)$ for each shared index, where $\ell(a), \ell(b)$ are as defined above. The $\mathcal{S}$ consists of all vertices corresponding to qubits in Eq. 2.43 that are not projected. The set $\mathcal{P}$ consists of all other vertices. For $v \in \mathcal{S}$, we have $P_v = Z$. For $v \in \mathcal{P}$, the value of P_v is determined by $\langle +|$ or $\langle \phi^{(j)}|$. Finally, we will take $m = 0^{|\mathcal{P}|}$, which ensures that all gadgets are implemented with no error.

To prove the theorem we show that the graph state simulation on $\mathcal{G}$ can be solved in time $\widetilde{O}(n^{\omega/2} r^\omega)$. We will construct an $O(r)$-coarse-graining $f : \mathcal{G} \mapsto G(\mathcal{T})$. Since $G(\mathcal{T})$ is assumed to be planar we can then use Theorem 13 to get a nice tree decomposition T for $G(\mathcal{T})$ with $||T||_\omega^\omega = \widetilde{O}(n^{\omega/2})$, then Lemma 27 to get a tree decomposition T' for $\mathcal{G}$ with $||T'|| = \widetilde{O}(n^{\omega/2} r^\omega)$, and finally Theorem 14 to solve the graph state simulation problem on $\mathcal{G}$.

To define f, first let $[n] = V(G(\mathcal{T}))$. Then for each $k \in [n]$, map all vertices of G'_k to k. If $u'v' \in \mathcal{G}$, then either u' and v' are both in the same G'_k, or $u' \in G'_j$ and $v' \in G'_k$ for some $j \neq k$ with $T^{(j)}$ and $T^{(k)}$ sharing an index. If $u', v' \in G'_k$, then $f(u') = f(v')$. Otherwise, because $T^{(j)}$ and $T^{(k)}$ share an index, we must have $f(u')f(v') \in E(G(\mathcal{T}))$. Therefore f is a coarse-graining. Because each $T^{(k)}$ has rank $O(r)$ and each gadget has constant size, $|V(G'_k)| = O(r)$ for all k, and so f is in fact an $O(r)$-coarse-graining. □

Proof of Theorems 31b and 31c. We will replace the tensor network with the graph state $|\mathcal{G}\rangle$ and form the tree decomposition T' for $\mathcal{G}$ in the same way as we did in the previous proof. The parameters $\mathcal{P}, \mathcal{S}, \{P_v\}_v$, etc. which were given as input to the graph state simulation problem in the previous proof are defined in the same way.

The solution to the strong graph state simulation problem on $|G\rangle$ with these parameters is equal to $|\mathcal{T}_{z_1,\ldots,z_N}|^2$. To prove Theorem 31b we solve this instance of the strong graph state simulation problem by calling Theorem 23, which gives the runtime bound of $\widetilde{O}(n^{\omega/2} r^\omega)$. To prove Theorem 31c we call Theorem 24 to solve the phase-sensitive graph state simulation problem in time $\widetilde{O}(n^{3/2} r^3)$, which gives $\mathcal{T}_{z_1,\ldots,z_N}$. □

It will also be useful in Chapter 3 to use the following minor extension of Theorem 31c, which we will not prove. The only difference is that we use T directly instead of constructing it using Theorem 13.

Theorem 32. *Let $\mathcal{T}$ be a Clifford tensor network made up of n tensors $T^{(1)}, \ldots, T^{(n)}$, each of which has rank at most r. Set $N = rank(\mathcal{T})$, let $z \in \{0,1\}^N$, let $G(\mathcal{T})$ be the underlying graph of $\mathcal{T}$, and let T be a nice tree decomposition of $G(\mathcal{T})$. Then a classical algorithm can compute $\mathcal{T}_{z_1,\ldots,z_N}$ in time $\widetilde{O}(||T||_3^3 r^3)$.*

2.8 Discussion

At the beginning of this chapter we stated that [8] showed that quantum computers have a depth advantage over their classical counterparts when solving the graph state simulation problem on grid graphs. Theorem 9 tells us that if $\omega = 2$, then this advantage is not present when considering gate complexity, as our algorithm runs in time $\widetilde{O}(n^{\omega/2})$. However, it may still be possible to prove a gate complexity advantage in the case where G is nonplanar but sparse. If G is nonplanar, our algorithm requires that a tree decomposition T is given as

input, and runs in time $O(n|T|^{\omega-1})$. Since G is sparse, the naïve quantum algorithm for solving the graph state simulation problem on G takes time $O(n)$. On the other hand, if G does not have treewidth $O(1)$, then the classical algorithm presented in this chapter will not run in time $O(n)$.

Another open problem is whether local complementation can allow us to solve the graph state simulation problem in time $\widetilde{O}(n^{\omega/2})$ on nonplanar graphs. The complete graph K_n has treewidth $n-1$ and is nonplanar for $n \geq 5$. Even when given a tree decomposition, running our algorithm directly would therefore yield a runtime of $O(n \cdot n^{\omega-1}) = O(n^{\omega})$, which is no better than the naïve algorithm. However, K_n is equivalent through local complementation to the star graph, which is planar. In Section 1.4 we saw that for any vertex a and graph G, we have $|G\rangle = U\,|L_a(G)\rangle$, where U is a product of local Clifford operators. Because local Clifford operators map Pauli bases to Pauli bases, instead of solving the graph state simulation problem on K_n in time $O(n^{\omega})$, we can solve the graph state simulation problem on the star graph in time $O(n)$, albeit with different $\{P_v\}_v$ bases. More generally, understanding how local complementation can affect planarity might allow us to extend the $\widetilde{O}(n^{\omega/2})$ runtime of our algorithm to a broader range of graphs.

Finally, it remains unknown whether our algorithm is truly more powerful than the active set approach. Although we were unable to show that the active set approach can always be used to achieve a runtime of $\widetilde{O}(n^{\omega/2})$, it may still be possible. A *path decomposition* is a tree decomposition in which the tree is a path graph. If T is a path decomposition for a graph G, then it has no merge nodes, which means that $\mathcal{C}$ has no merge ancillas. If $\mathcal{C}$ has no merge ancillas, then our algorithm is an instance of the active set approach, because there is a one-to-one correspondence between vertices and qubits. If it could be shown that for all planar graphs there exists a path decomposition T with $||T||_{\omega}^{\omega} = \widetilde{O}(n^{\omega/2})$, then it would be true that the active set approach can always be used to give a solution to the graph state simulation problem in time $\widetilde{O}(n^{\omega/2})$. A fact that may be useful is that for every n-vertex graph G and tree decomposition T of G, there exists a path decomposition T' of G with $|T'| = O(|T| \log n)$ [22]. This fact implies that in the nonplanar case our algorithm has no advantage over the active set approach.

Chapter 3

Stabilizer rank methods

In this chapter we will extend the circuit simulation results of Chapter 2 to universal circuits. We will write $T = \mathrm{diag}(1, e^{i\pi/4})$ and for $t > 0$ we will write χ_t to denote the stabilizer rank of the state $(T|+\rangle)^{\otimes t}$, a quantity which we will define shortly. We say that circuits with input $|0^n\rangle$, gates from the set $\{H, S, \mathrm{CZ}\} \cup \{T\}$, and computational basis measurements are *Clifford+T* circuits. Before defining the stabilizer rank, we state the results of this chapter.

Theorem 33. *Let C be an n-qubit Clifford+T circuit with t T gates and depth d. Suppose the two-qubit gates of C act along the edges of a planar graph G. Given $z \in \{0,1\}^n$, a classical algorithm can find $|\langle z|C|0^n\rangle|^2$ in time $O(\chi_t n^{3/2} t^6 d^3) \leq O(2^{0.3963t} n^{3/2} t^6 d^3)$.*

Theorem 34. *Let C, G, t, and d be as in Theorem 33. A classical algorithm can sample $z \in \{0,1\}^n$ from the distribution $\Pr[z] = |\langle z|C|0^n\rangle|^2$ in time $O(\chi_t^2 n^{5/2} t^6 d^3) \leq O(2^{0.7926t} n^{5/2} t^6 d^3)$.*

Although Clifford circuits are not universal, Clifford+T circuits are [26]. The gadget in Fig. 3.1 shows that a T gate can be replaced with a Clifford gate and postselected measurement when given access to the non-stabilizer ancilla state $|T\rangle := T|+\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$ [E.g. 32]. To simulate a universal quantum circuit it is therefore sufficient to simulate circuits with Clifford gates, postselected computational basis measurements, and non-stabilizer input states of the form $|0^n\rangle \otimes |T\rangle^{\otimes t}$, for $t \geq 0$.

Stabilizer states span $\mathbb{C}^{2^n}$, so every quantum state $|\psi\rangle$ can be expressed in the form

$$|\psi\rangle = \sum_{j=1}^{\chi} \alpha_j |\phi_j\rangle, \tag{3.1}$$

where each $|\phi_j\rangle$ is a stabilizer state, each α_j is in $\mathbb{C}$, and $1 \leq \chi \leq 2^n$. We call such a decomposition a *stabilizer decomposition*. The *stabilizer rank* of $|\psi\rangle$, written $\chi(|\psi\rangle)$ is the

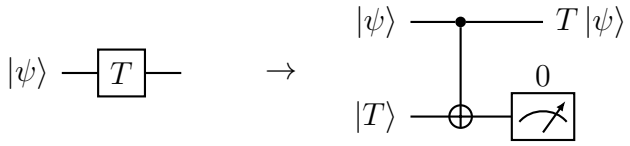


Figure 3.1: A Clifford operation and a postselected measurement can be used to apply T to $|\psi\rangle$, given the non-stabilizer ancilla state $|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)\sqrt{2}$.

minimum χ over all stabilizer decompositions [9]. In particular, it is known that $\chi(|T\rangle^{\otimes t}) \leq 2^{0.3963t}$ [27]. Since $\chi(|\phi\rangle \otimes |\psi\rangle) = \chi(|\psi\rangle)$ for any stabilizer state $|\phi\rangle$, we also have $\chi(|0^n\rangle \otimes |T\rangle^{\otimes t}) = \chi(|T\rangle^{\otimes t})$ for any $n, t > 0$. From now on, we will write $|T^t\rangle := |T\rangle^{\otimes t}$ and $\chi_t := \chi(|T^t\rangle)$.

Informally, stabilizer rank methods [9, 7, 6] simulate a circuit C that has Clifford gates but a non-stabilizer input state by forming a stabilizer decomposition for the input state, computing $C|\phi_j\rangle$ for each j, and then recombining the results to either compute an amplitude or sample from the output distribution of C. In this chapter we will show how stabilizer rank methods can be used in conjunction with planarity to obtain extensions of the results of Section 2.7. We will closely follow the techniques of [9, 7, 6].

3.1 Planar Clifford+T circuits

In this section we prove Theorems 33 and 34. For $d = O(\log n)$ previous work [9, 6] allows computation of output probabilities to be done in time $O(\chi_t n^3 t^3)$ and sampling to be done in time $O(\chi_t n^6 t^6)$. Theorem 33 offers improved scaling in the polynomial terms and identical scaling in the exponential term. Theorem 34 offers improved scaling in the polynomial terms but poorer scaling in the exponential term.

The proofs of Theorems 33 and 34 both require us to replace all T gates in the circuit C with the T gate gadget. Let us write this $(n+t)$-qubit circuit as C_{gadg}. An example of this circuit is shown in Fig. 3.2, centre. Using the calculation $(I \otimes \langle 0|)\text{CNOT}(|\psi\rangle \otimes |T\rangle) = T|\psi\rangle/\sqrt{2}$, we then have

$$C|0^n\rangle = 2^{t/2}(I^{\otimes n} \otimes \langle 0^t|)C_{\text{gadg}}(|0^n\rangle \otimes |T^t\rangle\,. \tag{3.2}$$

Consider a stabilizer decomposition $|T^t\rangle = \sum_{j=1}^{\chi_t} \alpha_j |\phi_j\rangle$. For each $|\phi_j\rangle$, let $D^{(j)}$ be a circuit of depth $O(t)$ such that $|\phi_j\rangle = D^{(j)}|0^t\rangle$. To see that such a $D^{(j)}$ exists, recall from Section 1.4 that we can write $|\phi^{(j)}\rangle = (V_1 \otimes \ldots \otimes V_t)|F_j\rangle$ for some local Clifford operators $V_1, \ldots, V_t$ and graph F_j. The state $|F_j\rangle$ can be prepared by preparing the state $|+^t\rangle$ and layering CZ gates according to an edge colouring of F_j [19]. The chromatic index of F_j is at most $t+1$, so $D^{(j)}$ has depth $O(t)$. Using this decomposition we can rewrite Eq. 3.2 as

$$C|0^n\rangle = 2^{t/2}\sum_{j=1}^{\chi_t} \alpha_j (I^{\otimes n} \otimes \langle 0^t|)C_{\text{gadg}}(I \otimes D^{(j)})|0^{n+t}\rangle\,. \tag{3.3}$$

The two-qubit gates of $C_{\text{gadg}}(I \otimes D^{(j)})$ act along the edges of some graph G_j, which we now describe. Let us write the vertices of F_j as $\{u_1, \ldots, u_t\}$ and the vertices of G as $\{v_1, \ldots, v_n\}$. Note that there is a one-to-one correspondence between $V(F_j)$ and the T gates in C, as each qubit in $D^{(j)}$ corresponds to the ancilla qubit in some T gate gadget. To form G_j, we begin with $G \cup F_j$, and then add an edge $u_i v_k$ whenever the T gate corresponding to u_i acts on wire k of C.

Next, we construct a nice tree decomposition $\mathcal{T}$ that is a valid nice tree decomposition for all G_j. (This is possible because each G_j has the same vertex set.) Using Theorem 13 we find a nice tree decomposition $T(G)$ for G with $||T(G)||_3^3 = \widetilde{O}(n^{3/2})$. We form $\mathcal{T}$ by adding $\{u_1, \ldots, u_t\}$ to every bag of $T(G)$ except the root bag. To see that this is a valid

nice tree decomposition, first notice that because $T(G)$ is rooted and has root bag equal to $\varnothing$, so does $\mathcal{T}$. Since $T(G)$ is a nice tree decomposition, every vertex v_i appears in some bag, the endpoints of every edge of the form v_iv_k appear together in some bag, and the bags containing any given v_i form a connected subtree. Because we formed $\mathcal{T}$ by adding all u_i to the non-root bags, clearly every u_i appears in some bag, and the endpoints of any edge of the form u_iu_k appear together in some bag. Since each v_i appears in some bag, we also have that every edge of the form v_iu_k has v_i and u_k appearing together in some bag. Finally, we must show that the bags containing any given u_i form a connected subtree. This condition is true if the root node (i.e. the only bag not containing u_i) has only one child. This must be true because the root node of a nice tree decomposition is always a forget node, and forget nodes have just one child. Since we have added t vertices to every bag, $||\mathcal{T}||_3^3 = O(||T(G)||_3^3 t^3) = \widetilde{O}(n^{3/2}t^3)$. We now prove Theorems 33 and 34.

Proof of Theorem 33. We will compute $|\langle z|C|0^n\rangle|^2$ as

$$|\langle z|C|0^n\rangle|^2 = 2^t \left| \sum_{j=1}^{\chi_t} \alpha_j (\langle z| \otimes \langle 0^t|) C_{\text{gadg}} (I \otimes D^{(j)}) |0^{n+t}\rangle \right|^2. \tag{3.4}$$

For each j, the circuit $C_{\text{gadg}}(I \otimes D^{(j)})$ has depth $O(d+t) \leq O(dt)$. Therefore by Theorem 28, we can compute a single term $(\langle z| \otimes \langle 0^t|) C_{\text{gadg}}(I \otimes D^{(j)}) |0^{n+t}\rangle$ in time $O(||\mathcal{T}||_3^3 d^3 t^3) = \widetilde{O}(n^{3/2}t^6d^3)$. Repeating this for each j gives a total runtime of $\widetilde{O}(\chi_t n^{3/2}t^6d^3)$, as desired. □

Proof of Theorem 34. Let $|\psi\rangle = C|0^n\rangle$. Suppose first that we have an algorithm which takes as input a stabilizer decomposition of a (possibly non-normalized) state and outputs that state's norm. Such an algorithm could be used to sample from the distribution $\Pr[z] = |\langle z|\psi\rangle|^2$ as follows. First, we compute

$$\Pr[z_1 = 0] = \frac{||(|0\rangle_1 \langle 0|_1 \otimes I)|\psi\rangle||^2}{||\,|\psi\rangle\,||^2}, \tag{3.5}$$

and sample $z_1 \in \{0,1\}$ from this distribution. Next, we sample the second qubit by computing the conditional probability

$$\Pr[z_2 = 0|z_1] = \frac{||(|z_1\rangle_1 \langle z_1|_1 \otimes |0\rangle_2 \langle 0|_2 \otimes I)|\psi\rangle||^2}{||(|0\rangle_1 \langle 0|_1 \otimes I)|\psi\rangle||^2}, \tag{3.6}$$

and then selecting z_2 according to this distribution. Continuing in this way, we sample all bits of z one at a time, computing $\Pr[z_k = 0|z_1, \ldots, z_{k-1}]$ as

$$\Pr[z_k = 0|z_1, \ldots, z_{k-1}] = \frac{||(\bigotimes_{i=1}^{k-1} |z_i\rangle_i \langle z_i|_i \otimes |0\rangle_k \langle 0|_k \otimes I)|\psi\rangle||^2}{||(\bigotimes_{i=1}^{k-1} |z_i\rangle_i \langle z_i|_i \otimes I)|\psi\rangle||^2}. \tag{3.7}$$

Because this process uses a total of $O(n)$ calls to the algorithm for computing norms, if we can describe an $\widetilde{O}(\chi_t^2 n^{3/2}t^6d^3)$-time algorithm for computing an expression of the form

$$||(\bigotimes_{i=1}^{k} |z_i\rangle \langle z_i| \otimes I)|\psi\rangle||^2, \tag{3.8}$$

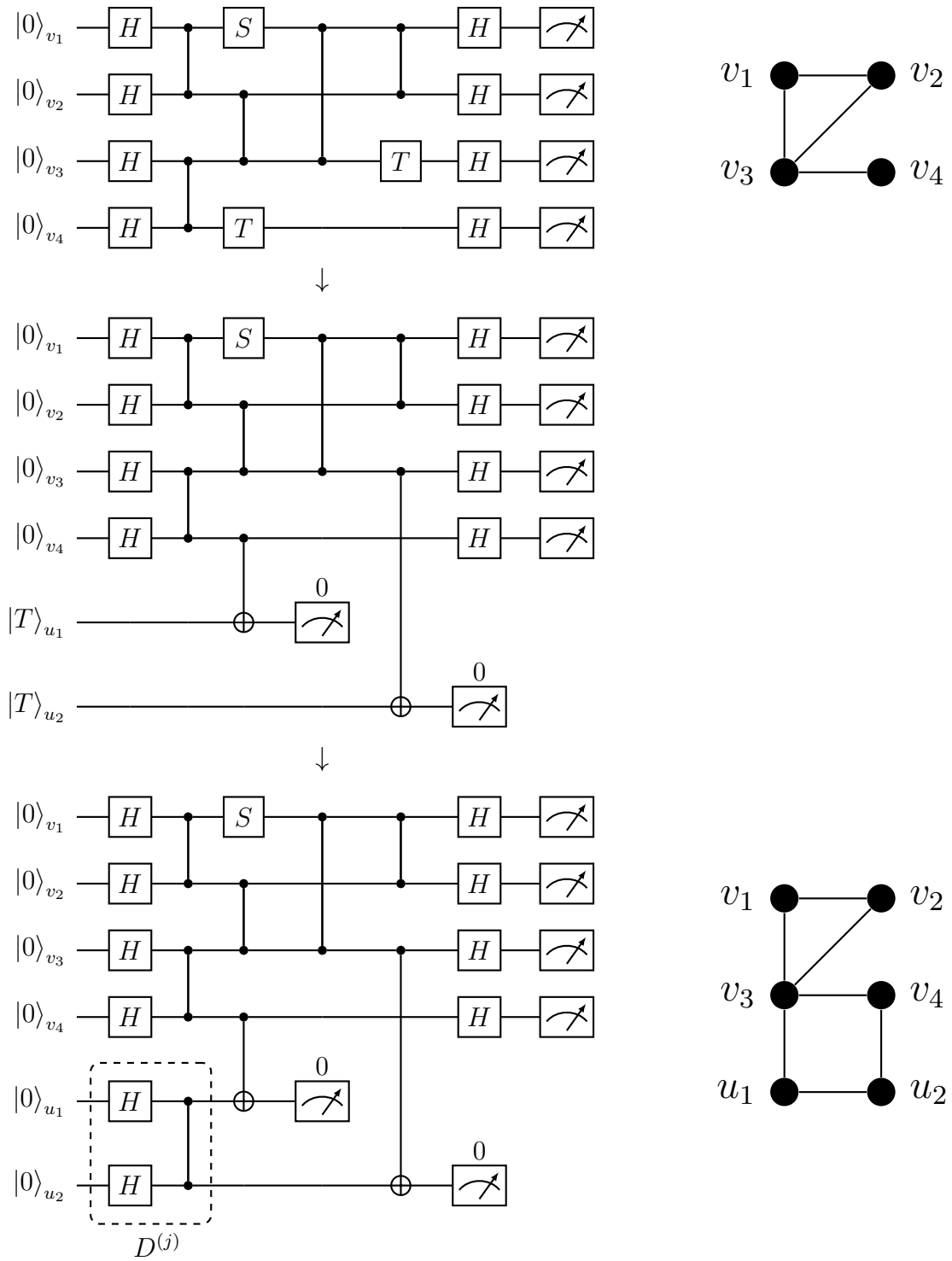


Figure 3.2: Example of the formation of $C_{\text{gadg}}(I \otimes D^{(j)})$ and G_j. *Top:* Circuit C and graph G. *Centre:* Circuit C_{gadg}. The CNOT gates can be written as $(I \otimes H)\text{CZ}(I \otimes H)$ but we omit this for simplicity. *Bottom:* Circuit $C_{\text{gadg}}(I \otimes D^{(j)})$ and the graph G_j. Here we have supposed that $|\phi^{(j)}\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$.

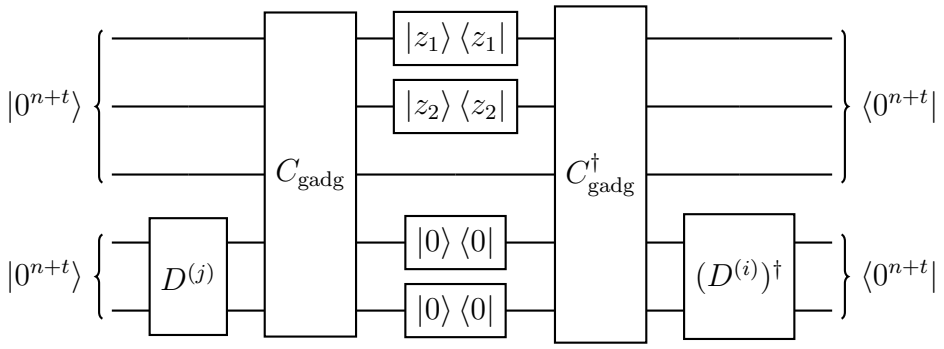


Figure 3.3: A small example of the circuit $\mathcal{C}$, defined by Eq. 3.11

for some $k \leq n$, then we'll be done. Let us write $\Pi = \bigotimes_{i=1}^{k} |z_i\rangle\langle z_i|$.

Using a stabilizer rank decomposition for $|T^t\rangle$, we have

$$||(\Pi \otimes I)|\psi\rangle||^2 = 2^t \left|\left|\sum_{j=1}^{\chi_t} \alpha_j (\Pi \otimes I \otimes |0^t\rangle\langle 0^t|) C_{\text{gadg}} (I \otimes D^{(j)}) |0^{n+t}\rangle\right|\right|^2 \tag{3.9}$$

This can be rewritten as

$$2^t \sum_{i,j=1}^{\chi_t} \bar{\alpha}_i \alpha_j \langle 0^{n+t}| (I \otimes D^{(i)})^\dagger C^\dagger_{\text{gadg}} (\Pi \otimes I \otimes |0^t\rangle\langle 0^t|) C_{\text{gadg}} (I \otimes D^{(j)}) |0^{n+t}\rangle . \tag{3.10}$$

There are χ_t^2 terms, so it will suffice to show how to compute a single summand in time $\widetilde{O}(n^{3/2}t^6d^3)$.

The quantity

$$\langle 0^{n+t}| (I \otimes D^{(i)})^\dagger C^\dagger_{\text{gadg}} (\Pi \otimes I \otimes |0^t\rangle\langle 0^t|) C_{\text{gadg}} (I \otimes D^{(j)}) |0^{n+t}\rangle \tag{3.11}$$

can be viewed in two ways. One way to view it is as a probability amplitude from a nonunitary circuit $\mathcal{C}$, shown in Fig. 3.3. The two-qubit gates of this circuit act along the edges of a graph $G(\mathcal{C}) = (V(G_i), E(G_i) \cup E(G_j))$. We can also view it as a fully contracted Clifford tensor network $\mathcal{N}$, made up of Clifford tensors $\{N^{(\ell)}\}_\ell$ from the set

$$\{|0\rangle, |1\rangle, H, S, \text{CZ}, |0\rangle\langle 0|, |1\rangle\langle 1|\}. \tag{3.12}$$

We will write $N^{(\ell)}$ to refer to both a tensor in $\mathcal{N}$ and a gate in $\mathcal{C}$. If we can find a nice tree decomposition $T(\mathcal{N})$ of the underlying graph $G(\mathcal{N})$ of $\mathcal{N}$ with $||T(\mathcal{N})||_3^3 = \widetilde{O}(n^{3/2}t^6d^3)$, then we'll be done, because we can then use Theorem 32 to compute Eq. 3.11 by contracting $\mathcal{N}$.

Because $\mathcal{T}$ is a valid tree decomposition for both G_j and G_i, it is also a valid tree decomposition for $G(\mathcal{C})$. An $O(d+t)$-coarse-graining $f : G(\mathcal{N}) \to G(\mathcal{C})$ is defined as follows. For each ℓ, if $N^{(\ell)}$ is not a CZ gate, set $f(\ell)$ to the wire on which $N^{(\ell)}$ acts in $\mathcal{C}$. If $N^{(\ell)}$ is a CZ gate acting on qubits a, b, set $f(\ell)$ to a or b arbitrarily.

To show that f is an $O(d+t)$-coarse-graining, consider an edge $\ell\ell' \in E(G(\mathcal{N}))$. If $N^{(\ell)}$ and $N^{(\ell')}$ are one-qubit gates acting on the same wire of $\mathcal{C}$, then $f(\ell) = f(\ell')$. If $N^{(\ell)}$ acts on

qubit a and $N^{(\ell')}$ acts on qubits a, b, then either $f(\ell) = f(\ell) = a$; or $f(\ell) = a$ and $f(\ell') = b$, in which case $f(\ell)f(\ell') = ab$ is an edge of $G(\mathcal{C})$ by definition of $G(\mathcal{C})$. The remaining two cases are similar. Therefore f is a coarse-graining. Because $\mathcal{C}$ has depth $O(d+t)$, f is an $O(d+t)$-coarse-graining. Since $d + t = O(dt)$, f is also and $O(dt)$-coarse-graining

Because f is an $O(dt)$-coarse-graining and $||\mathcal{T}|| = \widetilde{O}(n^{3/2}t^3)$, we have $||T(\mathcal{N})||_3^3 = \widetilde{O}(n^{3/2}t^6d^3)$, by Lemma 27. By Theorem 32, we can contract $\mathcal{N}$ in time $\widetilde{O}(n^{3/2}t^6d^3)$. □

We conclude this chapter with two open problems. The powers of 3 that appear in Theorems 33 and 34 arise from the need to use a phase-sensitive Clifford simulator when solving the phase-sensitive graph state simulation problem. If affine form could be improved so that k Hadamard gates could be applied in time $O(n^2k^{\omega-2})$, or even $O(n^\omega)$, then given a tree decomposition T, the phase-sensitive graph state simulation problem would be solvable in time $O(||T||_\omega^\omega)$ rather than $O(||T||_3^3)$. This would allow us to replace every 3 and 6 in these theorems with ω and 2ω. Another open problem is whether the algorithm described by Theorem 34 can be modified to have linear scaling in χ_t rather than quadratic. Methods for computing norms in time χ_t rather than χ_t^2 are known [7, 6], but these methods do not seem to be compatible with our approach, as they may require us to simulate circuits whose edges act along the edges of arbitrary graphs.

References

[1] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Phys. Rev. A* 70.5 (Jan. 2004).

[2] Josh Alman and Virginia Vassilevska Williams. "A Refined Laser Method and Faster Matrix Multiplication". In: *arXiv preprint arXiv:2010.05846* (2020).

[3] Simon Anders and Hans J. Briegel. "Fast simulation of stabilizer circuits using a graph-state representation". In: *Physical Review A* 73.2 (Feb. 2006).

[4] A. Ben-Israel and T.N.E. Greville. *Generalized Inverses: Theory and Applications*. CMS Books in Mathematics. Springer New York, 2006.

[5] Hans Bodlaender, Pål G. Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. "A $c^k n$ 5-Approximation Algorithm for Treewidth". In: *arXiv preprint arXiv:1304.6321* (2013).

[6] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. "Simulation of quantum circuits by low-rank stabilizer decompositions". In: *Quantum* 3 (Sept. 2019).

[7] Sergey Bravyi and David Gosset. "Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates". In: *Physical Review Letters* 116.25 (June 2016).

[8] Sergey Bravyi, David Gosset, and Robert König. "Quantum advantage with shallow circuits". In: *Science* 362.6412 (Oct. 2018).

[9] Sergey Bravyi, Graeme Smith, and John A. Smolin. "Trading Classical and Quantum Computational Resources". In: *Physical Review X* 6.2 (June 2016).

[10] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[11] Lars Eirik Danielsen. "On self-dual quantum codes, graphs, and Boolean functions". In: *arXiv preprint arXiv:quant-ph/0503236* (2005).

[12] Jeroen Dehaene and Bart De Moor. "Clifford group, stabilizer states, and linear and quadratic operations over GF(2)". In: *Phys. Rev. A* 68.4 (Oct. 2003).

[13] Jeffrey M Dudek, Leonardo Duenas-Osorio, and Moshe Y Vardi. "Efficient contraction of large tensor networks for weighted model counting through graph decompositions". In: *arXiv preprint arXiv:1908.04381* (2019).

[14] Alan George. "Nested dissection of a regular finite element mesh". In: *SIAM Journal on Numerical Analysis* 10.2 (1973).

[15] David Gosset, Daniel Grier, Alex Kerzner, and Luke Schaeffer. "Fast simulation of planar Clifford circuits". In: *arXiv preprint arXiv:2009.03218* (2020).

[16] Daniel Gottesman. "Stabilizer Codes and Quantum Error Correction". PhD thesis. Pasadena, USA: California Institute of Technology, May 1997.

[17] Johnnie Gray and Stefanos Kourtis. "Hyper-optimized tensor network contraction". In: *Quantum* 5 (Mar. 2021).

[18] Marc Hein, Jens Eisert, and Hans J Briegel. "Multiparty entanglement in graph states". In: *Physical Review A* 69.6 (2004).

[19] Peter Høyer, Mehdi Mhalla, and Simon Perdrix. "Resources Required for Preparing Graph States". In: *17th International Symposium on Algorithms and Computation (ISAAC 2006)*. Vol. 4288. Lecture Notes in Computer Science. Dec. 2006.

[20] Oscar H Ibarra, Shlomo Moran, and Roger Hui. "A generalization of the fast LUP matrix decomposition algorithm and applications". In: *Journal of Algorithms* 3.1 (1982).

[21] Richard Jozsa. "An introduction to measurement based quantum computation". In: *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment* 199 (2006).

[22] Ephraim Korach and Nir Solel. "Tree-width, path-width, and cutwidth". In: *Discrete Applied Mathematics* 43.1 (1993).

[23] Stefanos Kourtis, Claudio Chamon, Eduardo R. Mucciolo, and Andrei E. Ruckenstein. "Fast counting with tensor networks". In: *SciPost Phys.* 7 (5 2019).

[24] Ryan L. Mann. "Simulating Quantum Computations with Tutte Polynomials". In: *arXiv preprint arXiv:2101.00211* (2021).

[25] Igor L. Markov and Yaoyun Shi. "Simulating Quantum Computation by Contracting Tensor Networks". In: *SIAM Journal on Computing* 38.3 (Jan. 2008).

[26] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. 2nd. Cambridge University Press, 2010.

[27] Hammam Qassim. "Classical Simulations of Quantum Systems Using Stabilizer Decompositions". PhD thesis. Waterloo, Canada: University of Waterloo, 2020.

[28] Robert Raussendorf and Hans J. Briegel. "A One-Way Quantum Computer". In: *Phys. Rev. Lett.* 86.22 (May 2001).

[29] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. "Measurement-based quantum computation on cluster states". In: *Phys. Rev. A* 68.2 (Aug. 2003).

[30] Neil Robertson and P.D Seymour. "Graph minors. III. Planar tree-width". In: *Journal of Combinatorial Theory, Series B* 36.1 (1984).

[31] Dan Shepherd. "Binary Matroids and Quantum Probability Distributions". In: *arXiv preprint arXiv:1005.1744* (2010).

[32] Peter W Shor. "Fault-tolerant quantum computation". In: *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE. 1996.

[33] Volker Strassen. "Gaussian elimination is not optimal". In: *Numerische Mathematik* 13.4 (1969).

[34] Maarten Van den Nest. "Classical Simulation of Quantum Computation, the Gottesman-Knill Theorem, and Slightly Beyond". In: *Quantum Info. Comput.* 10.3 (Mar. 2010).

[35] Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. "Local unitary versus local Clifford equivalence of stabilizer states". In: *Physical Review A* 71.6 (2005).

[36] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. "Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (June 2019).

Appendix A

Fast linear algebra

The naïve algorithm for $n \times n$ matrix multiplication takes time $O(n^3)$, but it is known that this approach is not optimal [33]. We define ω to be the infimum of the set of ω' such that there exists an $O(n^{\omega'})$-time algorithm for matrix multiplication. It is currently known that $2 \leq \omega < 2.37286$ [2]. Any matrix multiplication algorithm also yields a matrix-inversion algorithm with the same asymptotic runtime [10]. In this appendix we first extend a result of Ibarra, Moran, and Hui, who show that several linear algebra tasks on non-square matrices can also be performed in matrix-multiplication time [20]. Then we prove some facts needed in Section 1.3.

Theorem 35 (Ibarra, Moran, Hui [20])**.** *Let A be an $m \times n$ matrix. There exist classical algorithms for the following tasks, with all operations over the field $\{0,1\}$.*

a. In time $O(\min(m,n)^{\omega-1} \max(m,n))$, find a maximum set of linearly independent rows or columns of A.

b. In time $O(n^\omega)$, find the affine solution space $\{Ru + t : u \in \{0,1\}^m\}$ for the system of equations $Ax = b$, with R having full rank.

Theorem 35a is shown explicitly in [20] but Theorem 35b is not, so we provide a proof here.

Proof of Theorem 35b. A generalized inverse of an $m \times n$ matrix A is an $n \times m$ matrix A^g such that $A = AA^gA$. We begin by finding A^g. Another algorithm of [20] allows us to do this directly if $m \leq n$. Otherwise we can use the fact that taking a generalized inverse commutes with matrix transposition and use the same algorithm on A^T. Either way, the running time is $O(n^\omega)$.

A system $Ax = b$ has a solution if and only if $A^gb = b$, and if it does, the set of solutions is given by [4]

$$\{(I - A^gA)z + A^gb : z \in \{0,1\}^n\}. \tag{A.1}$$

Since $I - A^gA$ might not have full rank[1], we use the first part of the theorem to remove any linearly dependent columns of $I - A^gA$ in time $O(n^\omega)$. □

[1]For example, if A is invertible then $I - A^gA = I - A^{-1}A = 0$.

We now restate and prove some lemmas from Section 1.3.

Lemma 4. *Let $N \geq 1$, let $f : \{0,1\}^N \to \{0,1,2,3\}$ be a linear function, let $g : \{0,1\}^N \to \{0,1\}$ be quadratic, and for $y \in \{0,1\}^N$ write $\bar{y} = y_2 \dots y_N$. An $O(N^2)$-time algorithm can find a linear function $f' : \{0,1\}^{N-1} \to \{0,1,2,3\}$, affine function $f'' : \{0,1\}^{N-1} \to \{0,1\}$, quadratic function $g' : \{0,1\}^{N-1} \to \{0,1\}$, and scalar $\alpha \in \mathbb{C}$ such that*

$$\sum_{y_1=0}^{1} i^{f(y)}(-1)^{g(y)} = \alpha i^{f'(\bar{y})}(-1)^{g'(\bar{y})}\delta_{f''(\bar{y}),0}. \tag{A.2}$$

Proof. For $j,k \leq n$, let us write f_j and g_{jk} for the y_j and y_jy_k coefficients in f and g, respectively. Then we have

$$\sum_{y_1=0}^{1} i^{f(y)}(-1)^{g(y)} = i^{f(0\bar{y})}(-1)^{g(0\bar{y})} + i^{f(1\bar{y})}(-1)^{g(1\bar{y})} \tag{A.3}$$

$$= i^{f(0\bar{y})}(-1)^{g(0\bar{y})}(1 + i^{f_1}(-1)^{g_{11}+\sum_{j>1} g_{1j}y_j}). \tag{A.4}$$

We can assume that $f_1 \in \{0,1\}$. If $f_1 = 2$ then we set $f_1 \leftarrow 0$ and $g_{11} \leftarrow g_{11} + 1$ in Eq. A.4. If $f_1 = 3$ we set $f_1 \leftarrow 1$ and $g_{11} \leftarrow g_{11} + 1$. If $f_1 = 1$ we can use the identity $1 + (-1)^k i = (1+i)(-1)^k i^{k^2}$ for integer k to rewrite this as

$$\sum_{y_1=0}^{1} i^{f(y)}(-1)^{g(y)} = (1+i)i^{f(0\bar{y})+\sum_{j>1}(g_{1j}+g_{11})y_j}(-1)^{g(0\bar{y})+\sum_{j>k>1} g_{1j}g_{1k}y_jy_k} \tag{A.5}$$

This is in the form of Eq. A.2 with $f'' = 0$ and $\alpha = 1 + i$. Computing the coefficients of the new function $g' = g(0\bar{y}) + \sum_{j>k>1} g_{1j}g_{1k}y_jy_k$ may be done in time $O(N^2)$. If $f_1 = 0$ then taking $f''(\bar{y}) = g_{11} + \sum_{j>1} g_{1j}y_j$ gives

$$\sum_{y_1=0}^{1} i^{f(y)}(-1)^{g(y)} = 2i^{f(0\bar{y})}(-1)^{g(0\bar{y})}\delta_{f''(\bar{y})}. \tag{A.6}$$

This is also in the form of Eq. A.2, with $\alpha = 2$. In this case, f', f'' and g' may be computed in time $O(N)$. □

Lemma 5. *Let $n > 1$, let $\mathbb{B} = \{Ru + t' : u \in \{0,1\}^m\} \subseteq \{0,1\}^n$ be an affine space with R in RCEF, and let $f'' : \{0,1\}^n \to \{0,1\}$ be an affine function. Then $\{y \in \mathbb{B} : f(y) = 0\}$ can be found in time $O(n^2)$.*

Proof. Let $f''(y) = F^T y + c$ for $F \in \{0,1\}^n$ and $c \in \{0,1\}$, and let us write $R^{(i)}$ for the ith column of R. We can find the desired subset by solving

$$F^T Ru + F^T t' + c = 0. \tag{A.7}$$

If no j with $F^T R^{(j)} \neq 0$ exists, then the desired subset is either empty (if $F^T t' + c = 1$) or all of $\mathbb{B}$ (if $F^T t' + c = 0$). Otherwise, choose such a j arbitrarily. Then we must have $u_j = F^T t' + c + \sum_{i \neq j} F^T R^{(i)} u_i$. We will use this representation to rewrite y in terms of all u_i with $i \neq j$. For each column $i \neq j$, set $R^{(i)} \leftarrow R^{(i)} + R^{(j)}(F^T R^{(i)})$ and set $t' \leftarrow t' + (F^T t' + c)R^T F$. Finally, remove $R^{(j)}$ and set $m \leftarrow m - 1$. This process leaves R in RCEF and takes time $O(n^2)$. □

Lemma 6. *Let A be an $n \times m$ matrix in RCEF and let $\Gamma \subseteq [n]$. There exists an invertible $m \times m$ matrix W such that AW is in RCEF and can be computed in time $O(|\Gamma|^{\omega-2}n^2)$ and the set $\{(AW)_j : j \in \Gamma \text{ and } (AW)_j \text{ is a pivot row}\}$ is maximal over all possible W.*

Proof. Let us assume for simpler notation that A takes the form $\left[\begin{smallmatrix} I \\ * \end{smallmatrix}\right]$ and that $\Gamma = \{m+1, \ldots, m+|\Gamma|\}$. We will write A as

$$A = \begin{bmatrix} I \\ S \\ T \end{bmatrix}, \tag{A.8}$$

where S is $|\Gamma| \times m$ and T is $(n-m-|\Gamma|) \times m$. Using Theorem 35a we find a maximum subset of linearly independent rows of S. Then we consider only these rows and use the theorem again to find a maximum subset of linearly independent columns of the submatrix of S consisting of those rows. This takes time $O(|\Gamma|^{\omega-1}n)$ and gives us an invertible $r \times r$ submatrix S_1 of S, where $r = \operatorname{rank}(S) \leq |\Gamma|$. Without loss of generality we assume that these rows and columns are the first r rows and columns of S. We can now write A as

$$A = \begin{bmatrix} I_r & 0 \\ 0 & I_{m-r} \\ S_1 & S_2 \\ S_3 & S_4 \\ T_1 & T_2 \end{bmatrix}. \tag{A.9}$$

We will take $W = W_1 W_2$, where

$$W_1 = \begin{bmatrix} S_1^{-1} & 0 \\ 0 & I \end{bmatrix} \qquad \text{and} \qquad W_2 = \begin{bmatrix} I & 0 \\ -S_2 & I \end{bmatrix}. \tag{A.10}$$

Computing S_1^{-1} takes time $O(|\Gamma|^\omega) \leq O(|\Gamma|^{\omega-2}n^2)$. To compute AW_1 in time $O(|\Gamma|^{\omega-1}n)$, notice that multiplying by W_1 does not affect columns $r+1, \ldots, m$ of A, so we are effectively performing an $(n \times r) \times (r \times r)$ matrix multiplication, which can be done in time $O(r^{\omega-1}n) \leq O(|\Gamma|^{\omega-1}n)$. To multiply by W_2, observe that we need to multiply the first r columns of AW_1 by $-S_2$ and add that to the last $n-r$ columns. This is an $(n \times r) \times (r \times (m-r))$ matrix product, which takes time $O(n^2 r^{\omega-2}) \leq O(n^2|\Gamma|^{\omega-2})$. Finally, we note that the set $\{(AW)_j : j \in \Gamma \text{ and } (AW)_j \text{ is a pivot row}\}$ is maximum because of the fact that S_1 was an invertible submatrix of S of maximum size. □