Investigating Quantum Computing via Algebraic and Logical Tools

by

Chaowen Guan

September 6th, 2019

A dissertation submitted to the
faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering

Abstract

"Quantum supremacy" or "quantum advantage" means demonstrating a quantum computer's ability to compute a task that no classical device can emulate in a comparable amount of time. This raises the question: how can one determine such an advantage? Much work on studying quantum supremacy has been done. In pursuing this question, this thesis takes a different angle from the great recent efforts at achieving quantum supremacy. In particular, it develops logical and algebraic tools for investigating *how well classical computers can emulate/simulate quantum computers.*

The first part studies a logical approach to classically emulate general quantum circuits. Specifically, we give a new explicit conversion from a quantum circuit C into a small set of Boolean formulas such that the acceptance amplitude of the circuit (on a given input x) can be computed from the numbers of satisfying assignments to the formulas. The fact of this has been known for two decades, but our compact constructions promote the use of heuristic #SAT solvers to perform emulation of quantum circuits. We implement a prototype of our simulator in which $\#SAT$ solver can be utilized to compute the acceptance probabilities.

The second part's main discovery is the *tight* connection between the strong simulation of quantum *stabilizer circuits* and two bedrock mathematical tasks: computing matrix rank and counting solutions to quadratic polynomials (both over the field $\mathbb{F}_2$). Precisely, it uses quadratic forms to obtain a strong simulation (i.e. computing the probability for any input and output) of stabilizer circuits. Our results improve the asymptotic running time from $O(n^3)$ to $O(n^\omega)$, where $\omega = 2.372...$ is the known exponent of matrix multiplication, as well as show a near-tight relationship to the task of computing matrix rank that was not known before. They also improve the $O(n^3)$-time algorithm for solution counting of quadratic forms over $\mathbb{F}_2$ to $O(n^\omega)$. Besides, we also find further connections to graph theory and matroid theory.

The third part builds on the second part to launch a direct attack on computing matrix rank over $\mathbb{F}_2$. Although rank reduces to matrix multiplication, they are not known to be equivalent. Getting any time better than $O(n^\omega)$ would be a major breakthrough. At a high-level, it combines quadratic forms and Fourier analysis to improve the time in some very special cases.

In the conclusion we close with a brief discussion of future research directions and then speculate about further applications of algebraic geometry in search of measures of the effort required to operate a quantum circuit that might explain the sustained difficult obstacles to maintaining quantum coherence that have been encountered.

Acknowledgements

I feel lucky to have had Kenneth W. Regan as my advisor who is always inspirational, supportive and generous. I have far too much to thank you for, Ken–I will always be indebted to you for all that you have taught me; for your outstanding advice and your faith in me, and so much more. Thanks for being an amazing advisor.

I thank my best friends: Si Chen, Zhan Qin, Jingyuan Fan, Zihao Shan, Michael Wehar, Yinghao Fu, Tong Guan, James Clay, Minghua Wang, Chen Xu and Xiangyu Guo for their supports through my entire graduate study. I thank the great committee members–Atri Rudra and Shi Li. My memories of this city are filled with the times I shared with these great people.

And above all I thank my family for their love, without which surely none of this would have been possible.

Contents

Chapter 1

Introduction

In the 1980s, Feynman [Fey82, Fey86] put forth the idea of quantum computation along with Deutsch [Deu85, Deu89], with Albert [Alb83] independently introducing quantum automata. The initial question that motivated Feynman to think up "quantum computers" was:

What kind of computer can we use to simulate (quantum) physics?

In principle, it is possible to use a classical computer to simulate the behavior of n-particle systems evolving according to quantum laws. However, it seems to require *exponentially* larger computing power. People have remarked that the particles seem to simulate themselves efficiently—so how does Nature compute? This brought out the following question:

Can we do it with a new kind of computer—a "quantum computer"?

Instead of simulating (quantum) physics, why don't we cluster the particles following their natural quantum-mechanical behavior to build a computer? Imaginably, this "quantum computer" would appear to be simulating a quantum system exponentially more efficiently than a classical computer would.

From the perspective of quantum physics, this idea implies that a multi-particle quantum system that computes like Nature (namely, quantum mechanics) will probably give exponential speedups for some natural computational problems. The most spectacular example that really brought the idea of quantum computers to wide attention is Shor's factoring algorithm [Sho94]. This was an algorithm implementable on a quantum computer that can factor any n-digit integer (with high probability) in roughly n^2 time. In contrast, the fastest known classical algorithm for factoring n-digit integer takes roughly $2^{n^{1/3}}$ time. As a consequence, a quantum computer capable of performing Shor's algorithm with moderate overhead would be able to break an enormous number of current real-world cryptographic applications in a matter of seconds.

This seemingly powerful nature of quantum computers has inspired researchers to explore the nature and limitation of quantum computing for three decades. The goal of demonstrating the ability of a quantum computer to perform important tasks that cannot be achieved classically is called "quantum supremacy" or "quantum advantage". Much work on studying "quantum supremacy" has been done. In this thesis, we take a different angle from these to show quantum supremacy. Particularly, this thesis develops logical and algebraic tools for investigating *how well classical computers can emulate/simulate quantum computers.*

This topic directly pertains to a second question:

how do you design and debug circuits/algorithms that you cannot even simulate efficiently with existing (classical) tools?

To debug circuits in the classical setting, programmers can add test conditions or read intermediate data to find where problems occur. In the quantum setting, those similar standard techniques would probably require measurements that destroy coherence. Furthermore, it seems likely that we may still not be able to build large-scale quantum computers in the short future, while physicists and chemists have long needed to simulate quantum systems. All these problems leads to a desire for methods to simulate quantum circuits on a classical computer.

1.1 Simulation of Quantum Circuits

Quantum circuits can efficiently perform n-bit computations that are commonly believed to require time exponential in n on classical computers. The salient example is Shor's algorithm [Sho94] for factoring n-bit integers using roughly $2n$ qubits. Cryptographic standards for hardness of factoring involve n on the order of $1,000$, while successful classical emulations of quantum circuits considered difficult have recently been claimed for n approaching 50 [HSST16, BIS$^+$16, HS17]. We are still a long way from implementing Shor's algorithm in practical time. By creating emulations of a quantum computer on classical systems we are able to test, at least to some degree, the workings of such algorithms. There are already several packages for simulating general quantum systems on classical computers: the quantum programming language QCL by [Öme14], the QuIDD (Quantum Information Decision Diagrams) package by Viamontes et al. [VMH03, VRMH03], and the parallel quantum computer simulator by Obenland and Despain [OD98].

Simulators and emulators use several forms. Aaronson and Chen [AC17] summarized that the space of emulation strategies is bounded by two pure strategies they call "Schrödinger" and "Feynman". The Schrödinger approach is to encode (pure) quantum states by length-N vectors, where $N = 2^n$, under some hybrid of explicit and implicit representation. A direct simulation implements the $N \times N$ matrices defining (compositions of) gate operations under sparse representations of these matrices. Thus the exponential penalty is paid up front but additional operations only involve $N \times N$ matrix computations with no further blowup. The Feynman way is to calculate an amplitude as a sum of terms, where it first pre-process a given circuit C in time scaling as $n^{O(1)}$. Only the final summation involves exponential blowup but the number of items being summed can have order worse than N or N^2.

1.2 Technical Tools

This thesis employs elementary mathematical tools that have been used to analyze quantum circuits before but extends the analysis in new ways. Here is a preview.

- **Boolean formulas**. A Boolean formula is a string of symbols consisting of variables, 0/1 (0 means false and 1 means true) and Boolean logics. Basic Boolean logics include $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation). In this thesis, we also use other operations such as $\oplus$ (exclusive or) and $\equiv$ (equivalence). Examples of using Boolean logic to help synthesize quantum circuits include [LCJ13, Lin14, SRWDM17]. What distinguishes the work is using Boolean formulas to effect an efficient reduction from quantum simulation to #SAT (in **Chapter** 5).

- **Polynomials**. The first express conversion to (sets of) polynomials was by Dawson et al. [DHH+04] and programmed by Gerdt and Severyanov [GS06]. It applied only to the universal set $\{\mathsf{H}, \mathsf{CNOT}, \mathsf{Tof}\}$ of gates with ± 1 entries, except for remarks in [DHH+04] about "mixed mode (mod-2/mod-8) arithmetic." Bacon, van Dam, and Russell [BvDR08a] tailored a construction to (singly and doubly) controlled phase-changing gates modulo various values. In **Chapter** 6, we use a special form, named *quadratic form*, of polynomials to further improve the strong simulation of stabilizer circuits and unveil its new connections with two fundamental problems: matrix rank and counting solutions to quadratic polynomials, both over $\mathbb{F}_2$.

- **Quadratic Forms** [Sch09, O13]. A quadratic form is a polynomial with terms all of degree two. For instance, $3x^2 + xy - 2y^2$ is a quadratic form in variables x and y. The coefficients usually belong to an integer ring or a field. More discussion on this is in Section 6.2. Quadratic forms over $\mathbb{Z}_4$ are the main objects we are working with in **Chapter** 6. To our knowledge, our level of applying the theory of quadratic forms is new.

- **Algebraic Geometry** [CLO13, Har13]. Classically, this subject studies zeros of multivariate polynomials, which are called algebraic varieties. By relating the degree of varieties to computational complexity, tools like **Bézout's Theorem** have succeeded in many applications in computer science. One of the most important results is Strassen's nonlinear lower bound on arithmetic circuits [Str73]. These techniques seem to have advantages in analyzing polynomial-represented problems. It also finds applications in the field of computational complexity. Mulmuley and Sohoni [MS01, MS07, MS08] introduced the geometric complexity theory which is an alternating approach to the P vs. NP problem. In quantum computing, Bacon, van Dam, and Russell [BvDR08a] found that the amplitude of certain special class of quantum circuits is highly related to the concept of singularity. Our discussion in **Chapter** 8 askes if we can find any trail of nonlinearity using algebraic geometry.

1.3 Logical Emulation of General Quantum Circuits

This starts from the Feynman simulation approach. In **Chapter** 5 we give a new explicit conversion from a quantum circuit C into a small set of Boolean formulas such that the acceptance amplitude of the circuit (on a given input x) can be computed from the numbers of satisfying assignments to the formulas. The fact of this has been known for two decades, but our compact constructions promote the use of heuristic #SAT solvers to perform emulation of quantum circuits. More precisely, we propose to design small sets of Boolean formulas $\phi_k^C(z_1, \dots, z_r)$ (in conjunctive normal form) such that C can be simulated with exact knowledge of the number of assignments in $\{0,1\}^r$ that satisfy the ϕ_k^C. The acceptance amplitude involves formulas ϕ_0, ϕ_1 with some number r of variables—h of them "free"—and gives values of the form

$$\frac{\#sat(\phi_0) - \#sat(\phi_1)}{R},$$

where $R = 2^{h/2}$ rather than be of order 2^h or 2^r. The total numbers n_0 and n_1 of satisfying assignments to ϕ_0 and ϕ_n will each have order 2^h, but their difference is *a priori* constrained to be at most R. It will therefore not suffice to compute n_0 to within a factor of $(1+\epsilon)$, say, nor likewise n_1. This is why exact #SAT solving is sought. The main result can be summarized as follows:

Theorem (Less formal description of Theorem 5.3.)**.** *Given C as a circuit of a common type of size s with m qubits. Then we can efficiently build a Boolean formula ϕ_C of size $O(s)$ and find a fractional constant R such that for any input $a \in \{0,1\}^m$ and output $b \in \{0,1\}^m$ to C:*

- *ϕ_C is a Boolean formula in variables $\vec{w}, \vec{x}, \vec{y}, \vec{z}$.*

- *The amplitude of C on input a and output b is equal to R times the sum over the number of 0/1-assignments to $\vec{y}$ in ϕ_C with $\vec{x} = a$, $\vec{z} = b$ and $\vec{w}$ be 0 through $K - 1$.*
- *For every assignment (a, c) to $\vec{x}, \vec{y}$ there is exactly one pair (L, b) such that $\phi_C[\vec{w} = L, \vec{x} = a, \vec{y} = c, \vec{z} = b]$ holds and it can be found in $O(s)$ time.*

Moreover, the last bullet above implies a "clever" brute-force #SAT solvers. Preliminary computational trials show that some freely available #SAT solvers give considerable advantage over brute-force and that this advantage scales non-linearly. Strategies for the #SAT solvers might be tuned for the special nature of the clauses arising from the "parity of AND" equations in our main technical theorem. Thus this can become a potential sub-field of intelligent software simulation of complex systems.

1.4 Algebraic Simulation of Stabilizer Circuits

A salient subclass of quantum circuits that have a deterministic polynomial-time simulation are *stabilizer circuits.* They can be generated by the following three gate matrices–Hadamard gate, phase gate and controlled-Z gate, respectively:

$$\mathsf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Their extensions to act on $\mathbf{C}^N$ by tensor product with the identity. It is worth pointing out that stabilizer circuits can also be generated with CZ gate replaced by CNOT gate, that is Hadamard gate, phase gate and controlled-NOT gate.

Stabilizer circuits play an important role in fault-tolerant circuits because they can be used to perform the encoding and decoding steps for a quantum error-correcting code. Besides that, those gates–H, S and CNOT–has many other applications. They are powerful enough to encompass most of the "paradoxes" of quantum mechanics, including the Greenberger-Horne-Zeilinger (GHZ) experiment [GHZ89], dense quantum coding [BW92], and quantum teleportation [BBC$^+$93]. The original polynomial-time algorithm by Gottesman and Knill [Got98] involved Gaussian elimination and so ran for all intents and purposes in order-of n^3 time. Aaronson and Gottesman [AG04] improved this to $O(n^2)$ time with a tableau method and also showed that every stabilizer circuit has an equivalent one with $O(n^2/\log n)$ gates. Both of these two work were built on the idea of *stabilizer groups* Section 3.2.

Chapter 6 uses quadratic forms instead of stabilizer groups to obtain a strong simulation (i.e. computing the probability $p = |\langle 0^n| \, C \, |0^n\rangle|^2$ [JvdN14]) of stabilizer circuits. Our results improve the asymptotic running time as well as show a near-tight relationship to the task of computing matrix rank that seems not to be noticed in these papers. They also improve the $O(n^3)$-time algorithm for solution counting of quadratic forms over $\mathbb{F}_2$, as given by Ehrenfeucht and Karpinski [EK90], to $O(n^\omega)$. Informally, our results are summarized as follows:

Theorem (Less formal description of Theorem 6.14)**.** *(a) Strong simulation of n-qubit stabilizer circuits on standard-basis inputs is in time $O(n^\omega)$ where $2 \leq \omega < 2.3729$.*

(b) Computing $n \times n$ matrix rank is linear-time equivalent to computing the probability p on the promise that p is positive.

In view of the normal form of [AG04] and in practice, the restriction on h and s in (b) is highly reasonable. Part (b) can be further rephrase as (Section 6.9): if membership in a certain class of undirected n-vertex graphs can be decided in $O(n^2)$, then the following problems all have the same time complexity:

- The strong simulation of quantum stabilizer circuits;
- The computation of rank of matrices over $\mathbb{F}_2$;
- The counting of solutions to classical quadratic forms modulo 4.

Note that the "promise" condition of (b) is ignorable in the direction from the rank r to p, but not from p to r. The sense of the latter direction is that if rank for dense matrices comes to have a lesser time $t(n)$ with $n^2 \leq t(n) < n^\omega$ than matrix multiplication, then computing p correctly in cases where $p > 0$ will have exactly the same time $t(n)$, whereas computing p in all cases might remain in n^ω time. We do not have a reduction from matrix multiplication itself (over $\mathbb{F}_2$) to strong simulation, hence our results do not imply an asymptotic equivalence between those. To be sure, we note as a practical caveat that among the known sub-cubic algorithms for matrix multiplication, only Strassen's original one [Str69], which runs in time $O(n^{2.81})$, is generally considered competitive for problem sizes in the range of thousands of qubits that are addressed concretely in the above-cited papers.

We may ask (1) *how far these techniques can apply to general quantum circuits*, and (2) *can the tools achieve better general complexity results?*

1.5 Outline of This Thesis

In **Chapter** 2 we present background knowledge on quantum computing and some corresponding complexity classes.

In **Chapter** 3 we survey past applications of algebraic methods in quantum computing.

In **Chapter** 4 we mainly discuss the connections between strong simulation of stabilizer circuits and two fundamental mathematical problems: computer matrix rank and counting solutions to quadratic polynomials (both over $\mathbb{F}_2$). More details of how to prove these connections are presented in **Chapter** 6.

In **Chapter** 5 we give a new explicit conversion from a quantum circuit C into a small set of Boolean formulas such that the acceptance amplitude of the circuit (on a given input x) can be computed from the numbers of satisfying assignments to the formulas. The fact of this has been known for two decades, but our compact constructions promote the use of heuristic #SAT solvers to perform emulation of quantum circuits. An alternate form facilitates uniform sampling of quantum outputs. The Boolean formulas assign one independent variable to each Hadamard gate; all other variables are forced by any assignment to them. Preliminary computational trials show that some freely available #SAT solvers give considerable advantage over brute-force and that this advantage scales non-linearly. Strategies for the #SAT solvers might be tuned for the special nature of the clauses arising from the "parity of AND" equations in our main technical theorem. Thus this can become a potential sub-field of intelligent software simulation of complex systems.

In **Chapter** 6 we show that a form of strong simulation for n-qubit quantum stabilizer circuits C of size s is computable in $O(s + n^\omega)$ time, where ω is the exponent of matrix multiplication. Solution counting for quadratic forms over $\mathbb{F}_2$ is also placed into $O(n^\omega)$ time. This improves previous $O(n^3)$ bounds. Our methods in fact show an $O(n^2)$-time reduction from matrix rank over $\mathbb{F}_2$ to computing an amplitude of C (hence also to solution counting) and a converse reduction that is $O(s + n^2)$ except for matrix multiplications used to decide whether $p > 0$. The current best-known worst-case time for matrix rank is $O(n^\omega)$ over $\mathbb{F}_2$, indeed over any field, while ω is currently upper-bounded by $2.3728\ldots$ Our methods draw on properties of classical quadratic forms over $\mathbb{Z}_4$. We study possible distributions of Feynman paths in the circuits and prove that the differences in $+1$ vs. -1 counts and $+i$ vs. $-i$ counts are always 0 or a power of 2. Further properties of quantum graph states and connections to graph and matroid theory are discussed.

In **Chapter** 7 we launch a possible attack on computing matrix rank over $\mathbb{F}_2$. Built on insights from the previous chapter, it combines quadratic forms and Fourier analysis to improve the time in some very special cases.

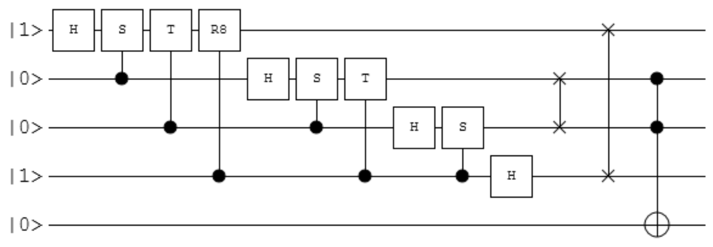
In **Chapter** 8 we close with a brief discussion of future research directions and speculate about more possible applications of algebraic methods in quantum computing.

Chapter 2

Quantum Computing and Complexity Classes

2.1 Quantum Circuits

To study quantum computing, most of the time we don't need to understand quantum physics. A quantum circuit is a compact representation of a quantum system. It consists of some number m of *qubits* represented by *lines* and some number s of *gates* acting on qubit lines. Here is an example created using the popular visual quantum circuit applet by Davy Wybiral [WH]:



```
 0.25000000+0.00000000i |00000>    6.2500%
-0.23096988-0.09567086i |00010>    6.2500%
 0.17677670+0.17677670i |00100>    6.2500%
-0.09567086-0.23096988i |00110>    6.2500%
 0.00000000+0.25000000i |01000>    6.2500%
 0.09567086-0.23096988i |01010>    6.2500%
-0.17677670+0.17677670i |01101>    6.2500%
 0.23096988-0.09567086i |01111>    6.2500%
-0.25000000+0.00000000i |10000>    6.2500%
```

Figure 2.1: A five-qubit quantum circuit that computes a Fourier transform on the first four qubits.

The circuit C operates on $m = 5$ qubits. The input is the binary string $x = 10010$. The first $n = 4$ qubits see most of the action and hold the nominal input $x_0 = 1001$ of length $n = 4$, while the fifth qubit is an *ancilla* initialized to 0 whose purpose here is to hold the nominal output bit. The circuit has thirteen *gates*. Six of them have a single *control* represented by a black dot; they activate if and only if the control bit receives a 1 signal. The last gate has two controls and a *target* represented by the parity symbol $\oplus$. It is called a *Toffoli gate*. This gate will set the output bit if and only if *both* controls receive a 1 signal. The two gates before it merely *swap* the qubits 2 and 3 and 1 and 4, respectively. They have no effect on the output bit (i.e., the fifth bit) and are included here only to say that the first twelve gates combine to compute the *quantum Fourier transform* QFT_4. This is nothing more than the ordinary discrete Fourier transform F_{16} on $2^4 = 16$ coordinates.

The actual output $C(x)$ of the circuit is a *quantum state* $\mathcal{Z}$ that belongs to the complex vector space $\mathbb{C}^{32}$. Nine of its entries in the *standard basis* are shown in Figure 2.1; seven more were cropped from the screenshot. Sixteen of the components are absent, meaning $\mathcal{Z}$ has 0 in the corresponding coordinates. Despite the diversity of the nine complex entries $\mathcal{Z}_L$ shown, each has magnitude $|\mathcal{Z}_L|^2 = 0.0625$. In general, $|\mathcal{Z}_L|^2$ represents the probability that a *measurement*—of all qubits—will yield the binary string $z \in \{0,1\}^5$ corresponding to the coordinate L under the standard ordered enumeration of $\{0,1\}^5$. Here we are interested in those z whose final entry z_5 is a 1. Two of them are shown; two others (11101 and 11111) are possible and also have probability $\frac{1}{16}$ each, making a total of $\frac{1}{4}$ probability for getting $z_5 = 1$. Owing to the "cylindrical" nature of the set B of strings ending in 1, one can also say that a measurement of just the fifth qubit yields 1 with probability $\frac{1}{4}$.

Below three important ingredients of quantum computing are summarized.

2.1.1 Quantum State

Each quantum state is a *superposition.* A quantum bit (qubit) is allowed to be in a superposition of the state 0 and 1. As is customary, we use the Dirac's *bra-ket* notation and a qubit with the label q can described by linear combination

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where the normalization restriction $|\alpha|^2 + |\beta|^2 = 1$ applies with the amplitude $\alpha, \beta \in \mathbb{C}$. This representation formulates the state space of a single qubit as the unit vectors in the two-dimensional Hilbert space $\mathcal{H}_2$.

Hence, for k qubits, there will be 2^k basis states and the corresponding superposition becomes a linear combination of all 2^k possible strings of k bits:

$$|q_1\rangle \otimes \cdots \otimes |q_k\rangle = \sum_{i \in \{0,1\}^k} \alpha_i\,|i\rangle\,,$$

where again as required, those amplitudes α_i obey: $\sum_i |\alpha_i|^2 = 1$. A *pure state* is any linear combination of basis states.

A state is said to be *entangled* if it can be decomposed as a tensor product of all single-qubit states. A famous entangled state is the Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ which has no valid decomposition as a tensor product of two single qubits. There are also *mixed states*, which can be represented as classical distributions on pure states.

2.1.2 Unitary Transformation on Quantum Bits

Quantum mechanics only allows transformations of states that are linear and respect the normalization restriction. For this purpose, the operations on an N-dimensional Hilbert space are the $N \times N$ complex-valued norm-preserving matrices. Such matrices are called *unitary* and the group of them is denoted by $U_N(\mathbb{C})$ (unitary group of degree N). Also this meets the requirement that the inverse of U is the conjugate transpose U^* of the matrix.

The effect of a unitary transformation U on a state $|x\rangle = \sum_i \alpha_i |i\rangle$ is exactly described by the corresponding rotation of the vector $|x\rangle$ in the appropriate Hilbert space. For this reason, U stands both for the quantum mechanical transformation as well as for the unitary rotation, which can be represented via matrix multiplication: $U|x\rangle = U \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}$. Hence, the associativity of matrix multiplication gives that the effect of two consecutive transformation U and W is the same as the single transformation $W \cdot U$, and the non-commutativity of matrix multiplication implies that the order of a sequence of unitary transformations matters.

Another important fact is that the tensor product of two unitary matrices is again unitary. Then a single-qubit gate matrix M on qubit i can be represented with the operator $\mathsf{U} = \mathsf{I} \otimes \cdots \otimes \mathsf{I} \otimes \mathsf{M} \otimes \mathsf{I} \otimes \cdots \otimes \mathsf{I}$ with M in position i. This also allows us to focus only on small-size quantum gates.

Some common single-qubit gate matrices are the following:

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{R_8} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}.$$

Adding *controls* is one way to extend effects to other qubits. The controlled form CG of a gate G has the block-matrix representation $\begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{G} \end{bmatrix}$. The gate $\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, which is also called the NOT gate for the negation it effects on the classical bit corresponding to the qubit line, yields the controlled forms CX (aka. CNOT) and CCX (aka. Tof for the Toffoli gate) in the first and third example below:

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathsf{CS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}, \quad \mathsf{Tof} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Following [DHH+04] a quantum gate is *balanced* if all nonzero entries in its unitary matrix representation have the same magnitude. This means that all common gates like Hadamard, CNOT, Tof, and rotation gates are balanced.

A classical gate set is said to be *universal* if, by combining enough gates from the set, one can express any Boolean function on any number of bits. A set of quantum gates being universal is defined analogously.

Definition 2.1. *A set of quantum gates is called universal if any quantum computation can be expressed by combining gates from the set.*

There are two series of universal:

- approximating the amplitudes;
- approximating the probabilities.

One demand that for an quantum state $|\phi\rangle$ produced by a general circuit of size s, there is a circuit of gates from the set that produces a state $|\phi'\rangle$ where vector distance from $|\phi\rangle$ can be made as small as desired. The other series merely requires that $|\phi'\rangle$ gives the same (measured) output results as $|\phi\rangle$. The set $\{\mathsf{H}, \mathsf{Tof}\}$ is universal only in the latter, *weaker* series, because it gives only real-valued entries. Nevertheless, it can emulate the real and imaginary parts of $|\phi\rangle$ separately. The set $\{\mathsf{H}, \mathsf{CNOT}, \mathsf{T}\}$ is universal in the stronger series.

It is worth mentioning that the controlled-S (CS) gate is important because it and H form a *universal set* of gates. So do H and Tof, or H, CNOT and T, in similar sense to $\{\mathsf{NAND}\}$ and $\{\mathsf{AND}, \mathsf{NOT}\}$ are universal sets of Boolean gates.

2.1.3 Measurements

Mathematically, a single "measurement" is transforming a quantum state to one possible outcome, which can be described as follows:

$$|z\rangle = \sum_i \alpha_i |i\rangle \xrightarrow[\text{outcome } m_i]{} |i\rangle .$$

The possible outcome "i" of z correspond to a set of orthogonal vectors $\{|m_i\rangle\}$ of the measuring device. "Measuring z" (meaning "interacting with z") will cause the state to *collapse* according to the outcome "m_i." From a probabilistic point of view, when measuring the state $|z\rangle$, the outcome "i" will be observed with probability $|\alpha_i|^2$. This probability can be computed via inner product.

Let $|\phi\rangle = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}$ and $|\psi\rangle = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}$ with $N = 2^n$ and n is the number of qubits. The complex conjugation in Dirac's bra-ket notation is denoted as

$$\langle\phi| = |\phi\rangle^\dagger = \sum_k a_i^* \langle i| = \begin{bmatrix} a_0^* & a_1^* & \cdots & a_{N-1}^* \end{bmatrix}.$$

Define the inner product over $\mathbb{C}^N$ as

$$\langle\phi|\psi\rangle = \sum_k a_k^* b_k.$$

With this, we have the above probability $|\alpha_i|^2 = |\langle m_i|z\rangle|^2$. This also facilitates the calculation of measuring $|z\rangle$ in any general orthonormal basis $\{|v_i\rangle\}$. (Note that $|v_i\rangle$ and $|v_j\rangle$ are orthogonal if and only if $\langle v_i|v_j\rangle = 0$.) Now if a measurement over $|z\rangle$ is done with respect to the basis $\{|v_i\rangle\}$, the probability of obtaining outcome "v_i" will be $|\langle v_i|z\rangle|^2$. Normally, we have a quantum circuit C and its input state $|x\rangle$, and then the output state $|z\rangle = C|x\rangle$ with C be a product of a sequence of matrices representing quantum gates. With this notation, the probability of seeing outcome "v_i" becomes $|\langle v_i|C|x\rangle|^2$.

Consider the basis $\{|+\rangle, |-\rangle\}$ with

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

Now measuring a state $|z\rangle$ with respect to this basis means that the outcomes can only be $|+\rangle$ or $|-\rangle$, and the probabilities of seeing them are $|\langle +|z\rangle|^2$ and $|\langle -|z\rangle|^2$ respectively.

In Figure 2.1, nine of the possible outputs (in the standard basis) are shown along with their corresponding probabilities and each probability is derived by squaring and summing the real and imaginary parts.

With the notions of measurement and output probability, we can define what it means in computational complexity, that a quantum algorithm solves a language. Since we focus on the quantum circuit model, here we can identify a quantum algorithm with a family of quantum circuits where each quantum circuit is an instance of the algorithm of certain input size. A family (C_n) of quantum circuits is said to solve a language L if for all input size n and all input x of size n

$$\Pr[\text{measuring the first qubit of } C_n(x) \text{ equals } L(x)] > \frac{2}{3}.$$

This is equivalent to saying that the family (C_n) can effectively identify L.

Another needed definition is *cylinder*. This definition of cylinder will be useful in simulating sampling over outputs of quantum circuits, and we will only be concerned with cylindrical measurements in the standard basis, i.e., measuring some subset of the qubits.

Definition 2.2. *Let a quantum circuit C of n-qubit output. Take some set $I = \{i_1, \cdots, i_\ell\} \subseteq \{1, \cdots, n\}$ for some ℓ and a string $w = w_1 \cdots w_\ell \in \{0,1\}^\ell$. Then we call $B \subseteq \{0,1\}^n$ is a cylinder with respect to I and w if*

$$B = \{\, b \in \{\, 0, 1\,\}^n : b_{i_1} = w_1 \wedge b_{i_2} = w_2 \wedge \cdots \wedge b_{i_\ell} = w_\ell \,\}\,.$$

Straightforwardly, a cylinder B represents outputs that are consistent with partial measurement already made, and thus is important in sampling.

There are two kinds of classical simulations: *weak simulation* and *strong simulation* [JvdN14].

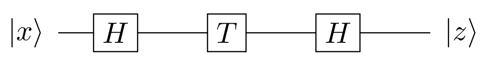
Definition 2.3. *A classical algorithm is said to be weakly simulating a quantum circuit if is it able to simulate sampling from the output distribution of the quantum circuit.*

Definition 2.4. *A classical algorithm is said to be strongly simulating a quantum circuit if is it able to calculate the probabilities of the output measurement outcomes with high accuracy.*

Our work in **Chapter** 5 and 6 concentrate on strong simulation.

2.1.4 More Examples

Single-qubit Circuit Example.



The input is a single qubit input x as a vector in $\in \mathbb{C}^2$. So is the output z. The first gate and the last gate are Hadamard gates, and the middle T-gate creates a $\frac{\pi}{4}$ phase shift by mapping $|1\rangle$ to $e^{i\pi/4}|1\rangle$ and leaving $|0\rangle$ unchanged. To see what the circuit really does, we can use its matrix representation, which is the following:

$$|z\rangle = \mathsf{HTH}\,|x\rangle = \frac{1}{2}\begin{bmatrix} 1 + e^{i\pi/4} & 1 - e^{i\pi/4} \\ 1 - e^{i\pi/4} & 1 + e^{i\pi/4} \end{bmatrix} |x\rangle\,.$$

Note that the matrix representation reads from right to left, for instance, the rightmost matrix H corresponds to the first Hadamard gate in the circuit. Let C be HTH, and then $\mathsf{HTH}\left|x\right\rangle = C\left|x\right\rangle$.

Write $\left|x\right\rangle = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$ with $x_0, x_1 \in \mathbb{C}$ and then

$$\left|z\right\rangle = \begin{bmatrix} \frac{x_0(1+e^{i\pi/4})+x_1(1-e^{i\pi/4})}{2} \\ \frac{x_0(1-e^{i\pi/4})+x_1(1+e^{i\pi/4})}{2} \end{bmatrix}.$$

Since $\left|z\right\rangle$ again belongs to $\mathbb{C}^2$, if we do a measurement, the squared norm of the first entry equals the probability of seeing outcome bit 0, and the second one for outcome 1. In particular, $|\left\langle 0|C|0\right\rangle|^2 \approx 0.85$ and $|\left\langle 1|C|0\right\rangle|^2 \approx 0.15$. The significance is that this disparity cannot be explained by saying that each H gate is like a classical random coin. This is made formal in the *CHSH game*.

CHSH Game [CHSH69]. There are two players, Alice and Bob, and a referee. The referee chooses two bits r, s independently and uniformly at random and then sends bit r to Alice and s to Bob. Alice and Bob must each answer a single bit: a for Alice, b for Bob. They win this game if $a \oplus b$ equals $r \wedge s$, i.e., the winning conditions are: if either r or s is 0, a and b should be equal; if $r = s = 1$, a and b should be different.

The *classical strategy* to maximize winning probability is simply that Alice and Bob always send the referee $a = b = 0$, regardless of what r and s are. In this case, Alice and Bob win 75% of the time, losing only if r and s are both 1. This can be proved by easily enumerating cases that this equation cant possibly hold for all 4 values of r and s. At best it can hold for 3 of the 4 values, which is exactly what this trivial strategy gets.

The *quantum strategy* requires Alice and Bob to pre-share an entangled Bell state, $\frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle$. Then it involves Alice and Bob measuring their respective qubits in different bases, depending on whether their received bits r and s are 0 or 1, and then outputting bits a and b respectively based on the outcomes of those measurements. If $r = 0$, Alice measures in basis $\{\left|0\right\rangle, \left|1\right\rangle\}$; if $r = 1$, she measures in basis $\{\left|+\right\rangle, \left|-\right\rangle\}$. On the other hand, Bob measures in basis $\{\left|u_0\right\rangle, \left|u_1\right\rangle\}$ if $s = 0$ and measures in basis $\{\left|v_0\right\rangle, \left|v_1\right\rangle\}$ where

$$\begin{aligned}
&\left|+\right\rangle = \frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle, \left|-\right\rangle = \frac{1}{\sqrt{2}}\left|0\right\rangle - \frac{1}{\sqrt{2}}\left|1\right\rangle, \\
&\left|u_0\right\rangle = \cos\frac{\pi}{8}\left|0\right\rangle + \sin\frac{\pi}{8}\left|1\right\rangle, \left|u_1\right\rangle = -\sin\frac{\pi}{8}\left|0\right\rangle + \cos\frac{\pi}{8}\left|1\right\rangle, \\
&\left|v_0\right\rangle = \cos(-\frac{\pi}{8})\left|0\right\rangle + \sin(-\frac{\pi}{8})\left|1\right\rangle, \left|v_1\right\rangle = -\sin(-\frac{\pi}{8})\left|0\right\rangle + \cos(-\frac{\pi}{8})\left|1\right\rangle.
\end{aligned}$$

At the end, Alice sets a to 0 if she sees $\left|0\right\rangle$ or $\left|+\right\rangle$ after measuring the pre-shared Bell state, and 1 if she sees $\left|1\right\rangle$ or $\left|-\right\rangle$; while Bob sets b to 0 if he sees $\left|u_0\right\rangle$ or $\left|v_0\right\rangle$, and 1 otherwise.

This strategy can make Alice and Bob win with probability $\cos^2\frac{\pi}{8} \approx 85\%$ for all possible values of r and s. To see this, let's consider the case where Alice gets $r = 0$ and measure. She will set $a = 0$, and she and Bob will win if and only if Bob outputs $b = 0$. So what are the odds that $b = 0$? Given that Alice measured her side of the Bell state already, Bob's side qubit collapsed to the $|0\rangle$ state as well. Now suppose $s = 0$. Then Bob measures the $|0\rangle$ state with the basis $\{|u_0\rangle, |u_1\rangle\}$, and outputs 0 if he sees $|u_0\rangle$, whose probability is $|\langle u_0|0\rangle|^2 = \cos^2\frac{\pi}{8}$. The same calculation can be applied to other cases. Hence, Alice and Bob win the game with probability about 85% in all four cases. It turns out that $cos^2\frac{\pi}{8}$ is the maximum probability with which Alice and Bob can win the CHSH game using a quantum strategy, a result known as *Tsirelson's bound* [Cir80].

Nevertheless, in **Chapter** 5 we will give an exact logical translation of this single-qubit quantum circuit that *does* represent each H gate by a classical binary variable. The quantum analysis will come from cosines in Theorem 5.3.

2.2 Background on Complexity Classes

We assume familiarity with the classes P, NP and $PSPACE$. Below we briefly review the *probabilistic* and *counting* classes most used in analyzing quantum computation.

BPP (**Bounded-Error Probabilistic Polynomial-Time**): the class of problems solvable by a *probabilistic* classical polynomial-time algorithm, which given any instance, must output the correct answer for that instance with probability at least 2/3. Thus P $\subseteq$ BPP $\subseteq$ PSPACE. It is widely conjectured that BPP = P [IW96], but not even known that BPP $\subseteq$ NP.

PP (**Probabilistic Polynomial-Time**): the class of problems solvable by a probabilistic classical polynomial-time algorithm, which given any instance, need only output the correct answer for that instance with probability greater than 1/2. The following problem is PP-complete: given a Boolean formula ϕ, decide whether at least half of the possible truth assignments satisfy ϕ. We have NP $\subseteq$ PP $\subseteq$ PSPACE and also BPP $\subseteq$ PP.

$\mathsf{P}^{\#\mathsf{P}}$ (**pronounced "P to the sharp-P"**): the class of problems solvable by a P machine that can access a "counting oracle." Given a Boolean formula ϕ, this oracle returns the number of truth assignments that satisfy ϕ. We have PP $\subseteq$ $\mathrm{P}^{\#\mathrm{P}}$ $\subseteq$ PSPACE.

BQP (**Bounded-Error Quantum Polynomial-Time**): the class of problems solvable by a quantum polynomial-time algorithm, which given any instance, must output the correct answer for that instance with probability at least 2/3. We have BPP $\subseteq$ BQP $\subseteq$ PP [BV97, ADH97].

Chapter 3

Algebraic Methods

This chapter reviews literature on analyzing quantum circuits by polynomials in more detail than was said in the Introduction (Section 1.2).

3.1 Polynomial Methods for Query Lower bounds

In the *quantum query model*, the goal usually is to compute some function $f : \{0,1\}^n \to \{0,1\}$ on a given input $x = (x_1, \cdots, x_n) \in \{0,1\}^n$. The distinguishing feature of this model is the way $\vec{x}$ is accessed: $\vec{x}$ is not given explicitly, but instead, the algorithm is being charged unit cost for every *query* it makes to $\vec{x}$. Informally, a query asks for and receives the i-th element x_i of the input. Let $\mathbf{O}_f$ be a *quantum query operation*. Note that a quantum algorithm can apply $\mathbf{O}_f$ to a superposition of basis states and gain access to several bits x_i at the same time.

A T-query quantum algorithm starts in a fixed state, say the all-zero state $|0 \cdot 0\rangle$, and then interleaves unitary transformations $\mathbf{U}_0, \mathbf{U}_1, \cdots, \mathbf{U}_T$ with queries. The final state of the algorithm can be written as the following sequence of matrix-vector products:

$$\mathbf{U}_T \mathbf{O}_f \mathbf{U}_{T-1} \mathbf{O}_f \cdots \mathbf{O}_f \mathbf{U}_1 \mathbf{O}_f \mathbf{U}_0 |0 \cdots 0\rangle .$$

This state depends on the input $\vec{x}$ only via the T queries. The output of the algorithm is obtained by a measurement of the final state. For instance, if the output is Boolean, the output could be the first bit of the final state measured in the computational basis.

Now the (bounded-error) quantum query complexity of some function f is defined as the minimum number of queries needed for an algorithm that outputs the correct value $f(x)$ for every x in the domain of f, say with error probability at most $1/3$, while treating all the intermediate unitary transformations as costless. Note that in many cases, the overall computation time (say measured by the total number of elementary quantum gates) of quantum query algorithms is not much greater than the query complexity, which justifies that it suffices to just count the queries. Many fundamental quantum algorithms have this form, including Deutsch-Jozsa [DJ92], Simon [Sim97] and Grover [Gro96].

In 1998, Beals et al. [BBC$^+$01] observed that the bounded-error quantum query complexity of a function f is lower bounded by (one half times) the *approximate degree* of f. This work showed an alternate approach to understand quantum algorithms, which is called *the polynomial method*, and its advantage led to a number of new lower bounds on quantum query complexity [AS04, BKT18].

Write as follows the final state of a T-query algorithm with given input $\vec{x} \in \{0,1\}^n$ acting on an m-qubit space

$$\sum_{z \in \{0,1\}^m} \alpha_z(x) \left|z\right\rangle .$$

The main observation by Beals et al. is that: the acceptance probability (i.e. $|\alpha_z(x)|^2$ for some output $\vec{z}$) of a T-query quantum algorithm is a polynomial in $x_1, \cdots, x_n$ of degree at most $2T$. In particular, $\alpha_z(x)$ is a multilinear polynomial in x of degree at most T and hence $|\alpha_z(x)|^2$ has degree at most $2T$.

Consider a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and our algorithm acting on an m-qubit state. Our final result is obtained by measuring the first qubit of the final state, whose probability of outputting 1 is given by (using the representation above)

$$P(x) = \sum_{z \in \{1\} \times \{0,1\}^{m-1}} |\alpha_z(x)|^2.$$

Now if the quantum algorithm computes f with error ϵ (i.e. $|P(x) - f(x)| \leq \epsilon$ for all $x \in \{0,1\}^n$), then we say $P(x)$ is an approximating polynomial for f and the *ϵ-approximate degree* of f is the degree of P. Hence the (bounded-error) quantum query complexity of this function f will be the smallest degree of any approximating polynomial for f. This gives an idea of how to lower bound the query complexity of computing f: if one can show that every approximating polynomial for f has degree at least $2T$, then every quantum algorithm computing f with error ϵ requires at least T queries.

The main difference in our work is that we analyze concrete circuits, not circuits with oracle gates. As with Boolean complexity, the lower bounds for query complexity do not carry over. The next section surveys some past work on concrete circuits.

3.2 Simulating Circuits via Polynomials

The first express conversion to (sets of) polynomials was by Dawson et al. [DHH$^+$04] and programmed by Gerdt and Severyanov [GS06]. It applied only to the universal set $\{\,\mathsf{H}, \mathsf{CNOT}, \mathsf{Tof}\,\}$

of gates with ± 1 entries, except for remarks in [DHH$^+$04] about "mixed mode (mod-2/mod-8) arithmetic."

In their model for a quantum circuit C on m qubits with h Hadamard gates, each bit z_i of output $\vec{z}$ is represented by a polynomial $z_i(\vec{x})$ in "path" variables variables $x_1, \cdots, x_h$. Then each assignment to $x_1, \cdots, x_h$ gives a path from input $(a_1, \cdots, a_m)$ to some output $\vec{z}(\vec{x}) = (z_1(\vec{x}), \cdots, z_m(\vec{x}))$. Dawson et al. also defined a *phase* polynomial $\phi(x)$ in variables $x_1, \cdots, x_h$. Then the amplitude $\langle \vec{b} | \; C \; | \vec{a} \rangle$ is given by the following sum over paths from input $\vec{a}$ to output $\vec{b}$:

$$\langle \vec{b} | \; C \; | \vec{a} \rangle = \frac{1}{\sqrt{2^h}} \sum_{\vec{x} : \vec{z}(\vec{x})} (-1)^{\phi(\vec{x})}.$$

Drawing on the analysis of Dawson et al. [DHH$^+$04], Bacon, van Dam, and Russell [BvDR08a] tailored a construction to (singly and doubly) controlled phase-changing gates modulo various values of K, like those for $K = 4, 8, 16$ in the last section's example, plus the Fourier transform F_K. More concretely, they introduced and analyzed algebraic quantum circuits that are defined over all finite rings $\mathbb{Z}_m$ and finite fields $\mathbb{F}_p$. This class of quantum circuits uses algebraic operations of addition and multiplication, as well as the quantum Fourier transform. They showed that every algebraic quantum circuit has a unique multivariate polynomial f associated with it that captures the "action" of the circuit, where in general f will be a cubic polynomial but for linear circuits (i.e. without multiplication) f is only quadratic.

With such a unique action polynomial f, the acceptance amplitudes of an algebraic quantum circuit again is expressed as an exponential sum over "paths" between the input and output of the circuit, which is Feynman style as described in **Chapter** 1. Let C be an algebraic quantum circuit over the ring $\mathbb{Z}_m$ with w wires, and h Fourier transforms, and define $n := k - w$, they derived the acceptance amplitude as

$$\langle \vec{0} | \; C \; | \vec{0} \rangle = \frac{1}{\sqrt{m^h}} \sum_{\vec{x} \in \mathbb{Z}_m^n} \exp(2\pi i f(\vec{x})/m).$$

For such an algebraic quantum circuit over field $\mathbb{F}_{p^r}$, the acceptance amplitude is

$$\langle \vec{0} | \; C \; | \vec{0} \rangle = \frac{1}{\sqrt{p^{rh}}} \sum_{\vec{x} \in \mathbb{F}_{p^r}^n} \exp(2\pi i \mathsf{Tr}(f(\vec{x}))/p),$$

where it uses the standard trace operation $\mathsf{Tr} : \mathbb{F}_{p^r} \to \mathbb{F}_p$ with $\mathsf{Tr} : x \to x + x^p + \cdots + x^{p^{r-1}}$. They prove several properties of algebraic quantum circuits.

Dawson et al.'s work [DHH$^+$04] also motivated the work by Regan and Chakrabarti [RC09]. They extended the polynomial constructions by Dawson et al. to (i) work for any set of quantum gates obeying a certain "balance" condition and (ii) produce a single polynomial over any sufficiently structured field or ring. Hence their work also implies a new proof of the Gottesman-Knill theorem [Got98].

The Gottesman-Knill theorem [Got98] was first proved using the idea of stabilizer groups, which is formally stated as the following theorem [Got98, AG04]:

Theorem. *Given an n-qubit state $|\psi\rangle$, the following are equivalent:*

- *$|\psi\rangle$ can be obtained from $|0\rangle^{\otimes n}$ by H, S and CNOT only.*
- *$|\psi\rangle$ can be obtained from $|0\rangle^{\otimes n}$ by H, S, CNOT and measurement gates only.*
- *$|\psi\rangle$ is stabilized by exactly 2^n Pauli operators.*
- *$|\psi\rangle$ is uniquely determined by $S(|\psi\rangle) = \mathrm{Stab}(|\psi\rangle) \cap \mathcal{P}_n$, the group of Pauli operators that stabilize $|\psi\rangle$.*

This theorem implies that any state $|\psi\rangle$ that can be obtained from $|0\rangle^{\otimes n}$ via a stabilizer circuit can be described uniquely by $S(|\psi\rangle)$, in particular, the n generators of $S(|\psi\rangle)$. Building on this, the Gottesman-Knill theorem gives a constructive method for simulating an n-qubit stabilizer circuit on a classical computer within polynomial time in n. Simulating a single measurement gate takes $O(n^3)$ time, while other unitary gates take $O(n)$ time for each gate. Hence, a strong simulation using their method takes $O(n^4)$ since it needs to measure all n qubits. Aaronson and Gottesman [AG04] improved the simulation time for a single measurement gate to $O(n^2)$ at the expense of space and thus requires $O(n^3)$ time for strong simulation.

In this thesis, we study the strong simulation (as defined in Definition 2.4) of stabilizer circuits with algebraic method in **Chapter** 6, completely different from the above idea of stabilizer groups.

Chapter 4

Matrix Rank, Solution Counting and Strong Simulation

In this chapter, we mainly discuss the *tight* connection (in [GR19]) strong simulation of stabilizer circuits and two mathematical problems: *computing matrix rank* and *counting solutions to quadratic polynomials* (both over the field $\mathbb{F}_2$). Because these two fundamental problems long preceded quantum computing, this chapter serves the purpose of explaining these connections more explicitly than that later in **Chapter** 6. More details of how to prove these connections are presented in **Chapter** 6.

4.1 Matrix Rank

Given an $m \times n$ matrix A over a field F, the rank of A, denoted by $\mathsf{rk}(A)$, is the maximum number of linearly independent columns of A. The problem of computing matrix rank is a basic computational problem in numerical linear algebra that is used as a subroutine for other problems [VZGG13, TBI97]. It has a number of applications in graph theory [MS04, Har09, Che97, CLL11, San07, Lov07, Lov06] and combinatorial optimization [Har09, CLL11].

The up to date work on this problem was done by Cheung et al. [CKL13]. They presented a randomized algorithm to compute the rank in $\tilde{O}(|A|+r^{\omega})$, where $|A|$ denotes the number of nonzero entries in A and $\omega < 2.38$ is the matrix multiplication exponent. They also presented a randomized algorithm that updates the rank in $\tilde{O}(mn)$ field operations, supporting the operations of rank one updates and adding and deleting rows and columns.

4.2 Solution Counting to Quadratic Polynomials

Counting the number of solutions to a polynomial equation is another fundamental mathematical problem. In many cases it is NP-hard, indeed NP-complete. However, some important special cases belong to polynomial time. The simplest non-linear such case is that of quadratic polynomials over $\mathbb{Z}_2$. Ehrenfeucht and Karpinski [EK90] gave an $O(n^3)$-time algorithm. This runtime was best known until this thesis [Kar19].

The analogous problem over $\mathbb{Z}_4$ is at the crux of a remarkably fine-cut "dichotomy" [CLX14, CGW17] between #P-completeness and back in P. In each case we have a quadratic polynomial $p(x_1, \cdots, x_n)$ over $\mathbb{Z}_4$ of the following type:

I. The number of solutions in $\{0, 1, 2, 3\}^n$ is polynomial-time computable.

II. If all cross-terms $x_i x_j$ with $i \neq j$ have coefficient 2, then the number of solutions in $\{0, 1\}^2$ is polynomial-time computable.

III. However, if $p(x_1, \cdots, x_n)$ can have cross-terms with coefficient 1 or 3, then counting the number of binary solutions is #P-complete.

This is significant because stabilizer circuits give polynomials of type 2 whereas adding the controlled-S (CS) gate creates polynomials of type 3. Thus the power of the universal set $\{\mathsf{H}, \mathsf{CS}\}$ versus $\{\mathsf{H}, \mathsf{CNOT}, \mathsf{S}\}$ creates this divide. The work by Ehrenfeucht and Karpinski [EK90] applies to type (II) as well. That is, their algorithm can do binary solution counting on type (II) in $O(n^3)$ time. Our work improved this to $O(n^\omega)$ time. Ehrenfeucht and Karpinski also showed that exactly counting solutions to a degree-3 polynomial over $\mathbb{F}_2$ is #P-complete. Woods [Woo98] proved a more general results for quadratic polynomials: there is a deterministic $n^3 \cdot poly(\log q)$-time algorithm for counting solutions to a quadratic polynomial over $\mathbb{F}_q$ for every prime power q.

4.3 Near-Tight Connections to Stabilizer Circuits

The connections between counting solution of quadratic polynomials and quantum circuits were noted in [RC09, CGW17]. This distinctive point of **Chapter** 6 in this regard is that we give concrete running times for the polynomial cases that improve the previous results. Moreover, the connection between matrix rank and strong simulation of stabilizer circuits is new to our knowledge.

The connection between the above two problems and strong simulation of stabilizer circuits is implied by the work in **Chapter** 6. In other words, the connection can be rephrased as follows:

(a) Given a n-qubit stabilizer circuit C, if the amplitude of C on input 0^n and output 0^n is not zero, then the time for strong simulation of circuit C is of the same magnitude order as that for computing matrix rank and for solution counting to quadratic polynomials.

(b) Given arbitrary $n \times n$ matrix $\mathbf{A}$ over $\mathbb{F}_2$ and form $\mathbf{A}' = \begin{bmatrix} 0 & \mathbf{A}_0 \\ \mathbf{A}_0^\top & 0 \end{bmatrix}$, then the rank of $\mathbf{A}'$ is determined by the probability of seeing output 0^{2n} on C with input 0^{2n}, where C is the stabilizer circuit transformed from $\mathbf{A}'$.

Part (a) is implied by Theorem 6.13 and part (a) of Theorem 6.14. More precisely, Section 6.3 and 6.4 shows that the runtime for converting a given stabilizer circuit to a quadratic polynomial and its corresponding matrix is $O(n^2)$. Then by Section 6.3 and Theorem 6.13, the time for computing the amplitude is the dominated by the time for counting solutions on the quadratic polynomial or that for computing matrix rank.

Part (b) follows from Theorem 6.13 and part (b1) of Theorem 6.14. In particular, the conversion from a matrix to a stabilizer circuit can be done in $O(n^2)$ (as shown in Section 6.3), and Theorem 6.13 gives the fact that computing matrix rank becomes performing strong simulation of this obtained circuit, which would take at least $O(n^2)$. Hence, the running time for computing the rank of the given matrix in this way is asymptotically the same as strong simulation of stabilizer circuits.

Overall, via these connections, our improvement on the strong simulation of stabilizer circuits leads to improvement on the problem of solution counting to quadratic polynomial over $\mathbb{F}_2$. These connections also suggests that any improvement on one will imply improvement for the others.

Chapter 5

Logical Emulation of Quantum Circuits

The results in this chapter are in the paper [RCG18].

5.1 Overview

In this chapter we give a new logical conversion from a quantum circuit C into a small set of Boolean formulas $\phi_k^C(z_1, \dots, z_r)$ (in conjunctive normal form) such that C can be simulated with exact knowledge of the number of assignments in $\{0, 1\}^r$ that satisfy the ϕ_k^C. That there are such reductions has long been known, but there appears to be no work programming them concretely, nor developing *properties* of the reductions that might aid heuristic simulations.

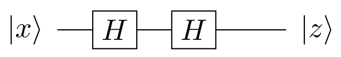
A reduction could be obtained by applying the generic Cook-Levin reduction to #SAT, starting from the polynomials p_C for instance. However, we seek the most efficient reductions, ones that are *natural*, *specific*, and (ideally) *tight*. We will use the technique that produces p_C as a proof guide. Our main general theorem in Section 5.3 requires only the knowledge that a quantum circuit with s gates computes a product of s large unitary matrices. Section 5.4 might need familiarity with quantum gates from sources such as [BDEJ95, BBC$^+$95, NC00] but we include the unitary matrices of the gates so it can be regarded as self-contained.

The salient property of our formulas is that they are all of the same form

$$p_{new} \equiv p_{old} \oplus a_1 \wedge \cdots \wedge a_m$$

This form is a controlled bit-flip (more details in Section 5.4). At qubit level, we already know that controlled-bitflip gates (plus Hadamard gates) are universal: the CNOT gate flips a qubit with one control, and the Toffoli gate flips with two. By going down to the level of Boolean logic, we show that controlled-bitflip equations can efficiently describe *all* behavior in quantum circuits. To begin, we show how they describe the behavior of Hadamard gates. Incidentally, note that the discussion of the HTH example (in Section 2.1.4) is relevant here.

First consider an example of two consecutive Hadamard gates:



Note that these two Hadamard gates will cancel and together behave as an identity. We can also see this behavior via our Boolean logic forms. Intuitively (from Section 5.2), the resulting Boolean formula will be like $(x \wedge y_1) \oplus (y_1 \wedge y_2)$ with each variable y_i associated with a Hadamard gate. This form can be converted into one as a conjunction of the above clauses, i.e.,

$$(p_1 \equiv p_0 \oplus (x \wedge y_1)) \bigwedge (p_2 \equiv p_1 \oplus (y_1 \wedge y_2)),$$

where p_i's are newly introduced variables and p_0 is fixed as 0. Denote the above equation as ϕ. Then the amplitude

$$\langle z|C|x\rangle = \frac{\#sat(\phi = 0) - \#sat(\phi = 1)}{(\sqrt{2})^2},$$

where $\#sat(\phi = 0)$ is the number of assignments to y_i's making $\phi = 0$, similar for $\#sat(\phi = 1)$. Say the input $|x\rangle = |0\rangle = |z\rangle$ which means $y_2 = 0$. We have $\langle 0|C|0\rangle = 1$ which means the input deterministically stays unchanged, and we can verify that it also holds for $|x\rangle = |1\rangle = |z\rangle$. However, if $|x\rangle = 0$ and $|z\rangle = 1$, $\#sat(\phi = 0) = 1 = \#sat(\phi = 1)$ and hence $\langle 1|C|0\rangle = 0$. This means that it is impossible to get output 1 if the input is 0. Similarly $\langle 0|C|1\rangle = 0$. Overall, we can see the identity property from our Boolean logic analysis. We will give more example later in Section 5.5, including how classical Boolean logic captures the circuit H-T-H in the CHSH game.

For CNOT gate with source qubit line labeled with u_i and target line with u_k, it creates the clause $v \equiv u_k \oplus u_i$ being conjoined into the above formula (here v is a new label for the source line); similarly for Toffili gate, it creates $v \equiv u_k \oplus (u_i \wedge u_j)$. Note that Hadamard gate, CNOT gate and Toffili gate constitute a universal quantum gate set. Hence we can say that all quantum circuits will have Boolean formulas as conjunction of clauses of this form. More concrete construction is presented in Section 5.2. We also extend the construction to other gates in Section 5.3.

Two important technical points concern approximation and sampling. For the first point, the acceptance amplitude involves formulas ϕ_0, ϕ_1 with some number r of variables—h of them "free"—and gives values of the form

$$\frac{\#sat(\phi_0) - \#sat(\phi_1)}{R},$$

where $R = 2^{h/2}$ rather than be of order 2^h or 2^r. The total numbers n_0 and n_1 of satisfying assignments to ϕ_0 and ϕ_n will each have order 2^h, but their difference is *a priori* constrained to be at most R. It will therefore not suffice to compute n_0 to within a factor of $(1+\epsilon)$, say, nor likewise n_1. This is why exact #SAT solving is sought.

On the other hand, many prominent quantum algorithms—Shor's among them—use quantum steps only to generate samples z from a distribution $\mathcal{D}$ on $\{0,1\}^n$. Here approximations z' to z are often tolerated. Hence we are most narrowly interested in the *sampling* problem for satisfying assignments which is related to *uniform generation.* Exact counting generally implies uniform sampling, but when and whether the latter affords more slack is a difficult problem in general. The second point is that to imitate the classical reduction from uniform sampling to *#sat* we need results that give the acceptance probability, rather than the amplitude, as a difference

$$\frac{\#sat(\psi_0) - \#sat(\psi_1)}{R^2},$$

where ψ_0 and ψ_1 are copies of ϕ_0 and ϕ_1. Whether further savings can be realized by further use of approximation will lead to further questions both about sampling and the workings of individual heuristic #SAT solvers [SBB$^+$04, SBK05a, SBK05b, Thu06]—and perhaps more general settings of *model counters* involving algebra.

5.2 Binary Case

We will first consider the common universal quantum gate set of Hadamard (H), controlled-NOT (CNOT) and Toffoli (Tof) gates. Then in the next section we will outline how to extend the constructions and proofs for other gates.

When C is composed of only H, CNOT, and Tof, the nonzero entries are ± 1 ignoring factors of $\sqrt{2}$ and so the resulting values $e^{i\theta}$ are likewise ± 1. Paths giving $+1$ are *positive paths* and those giving -1 are *negative paths.* For any basis value $a \in \{0,1\}^m$ given as input to the circuit and basis value $b \in \{0,1\}^m$ as a targeted output, we can isloate the paths that begin in row a of M_1 and end with column b of M_s. The number of those paths that are positive is denoted by $p^+(a,b)$, the number of negative paths by $p^-(a,b)$. If $h \leq s$ is the number of Hadamard gates then the amplitude of obtaining b as output by $C(a)$ is given by the product

$$\langle a|\ C\ |b\rangle = \frac{p_C^+(a,b) - p_C^-(a,b)}{2^{h/2}}$$

(ignoring complex conjugation since the values are all real). Our goal is hence to build two boolean formulas ϕ_0, ϕ_1 such that

$$\#p^-(a,b) = \#\text{ assignments satisfying } \phi_0$$
$$\#p^+(a,b) = \#\text{ assignments satisfying } \phi_1.$$

Before presenting the theorem for this case, it is worth pointing out some intuitions behind that:

- CNOT and Toffoli gates don't have effect on the sign of paths. The generated clauses for these two gates should always output 1 as long as the variable assignment is valid, i.e., preserving the input/output consistency.
- Only when both input qubit and output qubit to a Hadamard gate are 1's does the path's sign change. The binary basis values of these bits are represented by the variable pairs u_j, v_j.
- Accordingly, a path is positive if and only if it changes sign in Hadamard gates an even number of times (including zero). Such a path corresponds to an assignment that satisfies an even number of the terms $(u_j \wedge v_j)$.
- Whereas, the path is negative if and only if it corresponds to an assignment that satisfies an odd number of the terms $(u_j \wedge v_j)$, and so satisfies μ with $w = 1$.
- Assuming all other clauses (associated with CNOT and Toffoli gates) constantly have value 1, the formula ϕ_1 thus outputs 1 with assignments from the set of negative paths (denoted by $\mathcal{S}_-$), while ϕ_0 outputs 1 with assignments from the set of positive paths (denoted by $\mathcal{S}_+$).
- From above, the acceptance probability can be computed as
$$\frac{(|\mathcal{S}_+| - |\mathcal{S}_-|)^2}{2^h}.$$

Before presenting the main theorem of this section (Theorem 5.2), we first show the following lemma which is a more preliminary version of the theorem. The proof for this lemma gives a inductive construction procedure, while Theorem 5.2 mainly serves the purpose of being extended to other cases better.

Lemma 5.1. *We can convert C into a formula ψ_C of the form $\mu \wedge \eta$, where*

$$\mu = (\bar{w} \oplus (u_1 \wedge v_1) \oplus (u_2 \wedge v_2) \oplus \cdots \oplus (u_h \wedge v_h)).$$

Here some u_i variables may coincide with a v_j variable, but the $\{u_i\}$ and $\{v_j\}$ are individually distinct.

Proof. Let u_i, u_j, u_k stand for the current labels on the qubits i, j and/or k involved in a gate.

I. Label the inputs with variables $x_1, \cdots, x_n$. If there are ancilla qubits, then continue labeling them $x_{n+1}, \cdots, x_m$, although if they will always be initialized to 0, then one can label them 0 straightaway.

II. Label the outputs with variables $z_1, \cdots, z_n$, again using more if there are more qubits.

III. Let h be the number of Hadamard gates in the circuit, and allocate variables $y_1, \cdots, y_h$.

IV. Initially, set a formula $\psi = 1$.

V. For the next Hadamard gate $\mathbf{H}_j$ on some qubit line i, allocate a new variable y_j and then

   - when counting positive paths, calculate $\psi = \psi \overline{\oplus} (\neg(u_i \wedge y_j))$, and make y_j the new label on line i. Here $\overline{\oplus}$ means the complement to exclusive-or operation, that is, exclusive-or-then-negation.
   - when counting negative paths, calculate $\psi = \psi \oplus (u_i \wedge y_j)$, and make y_j the new label on line i.

VI. For a **CNOT** gate, leave the control label u_i unchanged, but u_j to $u_i \oplus u_j$. There is no change to ψ.

VII. For a Toffoli gate with controls on line i, j, leave u_i and u_j alone, but change the target u_k to $(u_i \wedge u_j) \oplus u_k$. There is no change to ψ.

VIII. When done with all the gates, for each i, create the measurement constraint $e(u_i, z_i)$, where u_i is the last label on line i and

$$e(u_i, z_i) = (\overline{u_i} \vee z_i) \wedge (u_i \vee \overline{z_i}).$$

Note that $e(0,0) = e(1,1) = 1$, while $e(0,1) = e(1,0) = 0$, so these enforce equality of the final labels.

IX. The final boolean formula ϕ is defined by taking the conjunction between ψ and all the measurement constraints. That is,

$$\phi = \psi \bigwedge_{i=1}^{n} e(u_i, z_i).$$

Note that one boolean formula can be constructed only for counting either positive paths or negative paths, not both. Hence, to keep track of both positive paths and negative paths at the same time, we need two boolean formulas. Take a closer look, we can see that the two ψ terms in the formula for positive paths is actually complement to that in the formula for negative paths. In particular,

$$\overline{\psi_1 \oplus \psi_2} = \psi_1 \overline{\oplus} \psi_2 = \overline{\psi_1}\overline{\oplus}\overline{\psi_2}$$

Therefore, if we maintain a boolean formula for positive paths, and we can derive the boolean formual for negative paths from it (which would be $\overline{\psi} \bigwedge e(u_i, z_i)$), and vice verse.

$\square$

For any cylinder B (as defined in Definition 2.2) of target output values, we can define $p^+(a, B) = \sum_{b \in B} p^+(a, b)$ and define $p^-(a, B)$ similarly. We will find it convenient to maintain these sets of paths when B is a cylinder. Singleton sets b have this form with $w = b$ as do sets $B_i = \{b : b_i = 1\}$ which represent measuring the single qubit i to test for a 1 value. Cylinders are important because $b_1, \dots, b_m$ as well as $a_1, \dots, a_m$ will stand for variables in our Boolean formulas whose values may selectively be substituted by 0-1 bit values.

Theorem 5.2. *Given any m-qubit circuit C of h Hadamard gates and $s-h$ Toffoli and CNOT gates, input $a \in \{0,1\}^m$, and cylinder $B \subseteq \{0,1\}^m$, we can construct a Boolean formula ϕ_C of size $O(s+m)$ in conjunctive normal form with variables $y_1, \dots, y_h, v_1, \dots, v_{s-h}, w, \dots, w_h$ together with $x_1, \dots, x_m$ and $z_1, \dots, z_m$ such that*

$$\begin{aligned} p_C^+(a, B) &= \#sat(\phi[\vec{x} = a, \vec{z} \in B, w_h = 0]) \\ p_C^-(a, B) &= \#sat(\phi[\vec{x} = a, \vec{z} \in B, w_h = 1]). \end{aligned}$$

Moreover, no two satisfying assignments agree on $y_1, \dots, y_h$.

Proof. We start with variables x_i, letting "u_i" initially refer to x_i on each line, and start with the equation $w_0 = 0$, i.e., $\bar{w}_0$. We let ℓ run from 1 to h this time, not 1 to s.

I. For each Hadamard gate on line i, increment ℓ, allocate fresh variables w_ℓ and y_ℓ, and conjoin the equation
$$(w_\ell = w_{\ell-1} \oplus (u_i \wedge y_\ell)).$$
To set up the next stage we note that "u_i" now refers to y_ℓ.

II. For each Toffoli gate with sources i, j and target k, increment o, allocate a fresh variable v_o and conjoin the equation
$$(v_o = u_k \oplus (u_i \wedge u_j)).$$
Now "u_k" refers to v_o.

III. For a CNOT gate with source i and target k, we conjoin $(v_o = u_k \oplus u_i)$ instead. Note this is the same as fixing $u_j = 1$ in the Toffoli case.

After placing the last gate, we conjoin the output-equating clauses $(u_i = z_i)$ for each qubit line i. Note again that in the case $h = 0$ we have $w_h = w_0$, and so $\phi_C[w_h = 1]$ is unsatisfiable—in keeping with there being no negative paths.

To finish the proof, we can use either of the following conversions to CNF. We can convert to 4CNF without introducing any more variables by applying to each equation the conversion

$$\begin{aligned} & (q = p \oplus (u \wedge y)) \\ \equiv\ & (\bar{u} \rightarrow p = q) \wedge (\bar{y} \rightarrow p = q) \wedge ((u \wedge y) \rightarrow p \neq q) \\ \equiv\ & (u \vee p \vee \bar{q}) \wedge (u \vee \bar{p} \vee q) \wedge (y \vee p \vee \bar{q}) \wedge (y \vee \bar{p} \vee q) \wedge (\bar{u} \vee \bar{y} \vee p \vee q) \\ & \wedge (\bar{u} \vee \bar{y} \vee \bar{p} \vee \bar{q}). \end{aligned} \tag{5.1}$$

To obtain 3CNF we need to introduce a new variable t and equation $t = u \wedge y$. Doing so does not increase the number of satisfying assignments. The clauses thus obtained are:

$$\begin{aligned} & (q = p \oplus t) \wedge (t = u \wedge y) \\ \equiv\ & (\bar{t} \rightarrow p = q) \wedge (t \rightarrow p \neq q) \wedge (\bar{u} \rightarrow \bar{t}) \wedge (\bar{y} \rightarrow \bar{t} \wedge ((u \wedge y) \rightarrow t) \\ \equiv\ & (t \vee p \vee \bar{q}) \wedge (t \vee \bar{p} \vee q) \wedge (\bar{t} \vee p \vee q) \wedge (\bar{t} \vee \bar{p} \vee \bar{q}) \wedge (u \vee \bar{t}) \\ & \wedge (y \vee \bar{t}) \wedge (\bar{u} \vee \bar{y} \vee t). \end{aligned} \tag{5.2}$$

The end equations $u_i = z_i$ become clause pairs $(u_i \vee \bar{z}_i) \wedge (\bar{u}_i \vee z_i)$. Overall, if C has m qubits, h Hadamard gates, and $s - h$ Toffoli plus CNOT gates, then the 4CNF formula has:

- h Hadamard variables $y_1, \ldots, y_h$;
- $h + 1$ indicator variables $w_0, \ldots, w_h$;
- $s - h$ line variables $v_1, \ldots, v_{s-h}$;
- m input variables $x_1, \ldots, x_m$—which, however, are substituted for when presenting any input $a \in \{0, 1\}^m$; and
- up to m output variables $z_1, \ldots, z_m$ per the discussion of cylinders above.

These variables form $6h + 6t = 6s$ 3-clauses and 4-clauses, plus the 1-clause $\bar{w}_0$ plus up to $2m$ output clauses, making $O(s + m)$ clauses in all. The 3CNF formula adds s more variables and has $7s$ clauses besides $(\bar{w}_0)$ and the output ones. As observed above, the formulas meet the requirements of Theorem 5.2 for any B. □

The Hadamard and Toffoli gates form a *universal* gate set for approximating quantum probabilities [Shi03] by themselves. It is striking that each adds one 4-ary *cBF*, though different roles for the variables. The Hadamard and CNOT gates do not form a universal set, but adding either T or CS creates a set that is capable of approximating quantum amplitudes (with complex values) not just probabilities.

5.3 More General Theorem

The result of the previous section suffices to describe a family of universal quantum computations. Thus it is good enough in theory. However, besides this universal family, there are many other commonly used gates, such as T, controlled-S, controlled-Z, and so on. Hence, for general use, in this section we want to show that we can efficiently represent a wide variety of quantum gates and circuits. The logic will represent the complex amplitude of every component of the state vector. Every nonzero component is the final "location" of some Feynman path(s). Some zero components may be final locations of paths that cancel.

First, we give a translation for completely general unitary gates (of any arity). Then we show that for a wide range of quantum gates, the general construction naturally gives small sets of controlled-bitflip equations for each gate.

Let C be a quantum circuit on m qubits with s gates and put $M = 2^m$. Consider the formal product of the s-many unitary $M \times M$ matrices U_ℓ, one for each gate in C. It expands to a sum of s-fold products of matrix entries. Every nonzero product of entries in this sum can be called a *Feynman path* through the object described by C. The value of the product is a complex number $re^{i\theta}$ with *phase* θ. Our general theorem will describe θ and r in binary notation via Boolean coding, i.e., Boolean formulas.

One thing to note is that (as already seen in Section 2.1.4) it is customary to write the input a to C on the left and list the gates/gate-matrices left-to-right as $U_1, \dots, U_s$, but the matrix computations are

$$C(a) = U_s U_{s-1} \cdots U_2 U_1 a; \quad \langle b| \; C \; |a\rangle = \langle b, C(a)\rangle.$$

If a path begins in row i of a, then it enters U_1 through column i and exits through some row j, whereupon it enters U_2 through column j. For better intuition we might wish to see it using either the transposed computation or the conjugate transpose,

$$C(a) = a^T U_1^T \cdots U_{s-1}^T U_s^T \text{ or } C(a)^* = a^* U_1^* \cdots U_{s-1}^* U_s^*,$$

and talk about the path entering row i of the first matrix and exiting via column j, etc., so as to align with how we read the circuit. At any stage, the path is in (row i, column j) of some matrix and has a current phase θ. We call this i the *location* of the path. The core of the proof is to write logical formulas that describe allowed changes in locations and phases of paths.

We make the following two mild assumptions about U_ℓ and θ.

- All nonzero entries of U_ℓ have the same magnitude, which means that U_ℓ is balanced as defined by Section 2.1.2.

- Phase θ is an integer multiple of $2\pi/K$, say $c2\pi/K$, where $K = 2^k$ for some k and some constant c.

This property of being balanced is preserved under tensor products, so it suffices to verify it for the $2^r \times 2^r$ matrix defining an r-ary gate locally. Denote $\omega = e^{2\pi i/K}$ a primitive K-root of unity. The second assumption makes $e^{i\theta}$ as a primitive K-root of unity as well. We also call either $2\pi/K$ or $1/K$ the *min-phase* of the circuit, and then the phases identified with $0, \ldots, K-1$ modulo K. These two assumptions won't hurt the generality of our theorem because: (1) all common gates are balanced, and (2) with min-phase, any phase θ can be represented in binary notation.

If $x = x_1, \ldots, x_n$ are variables in a Boolean formula ϕ and $a \in \{0,1\}$, then $\phi[x = a]$ stands for the formula obtained by substituting a_i for x_i for each i and simplifying operations involving constants. Now we can state and prove the main general theorem:

Theorem 5.3. *Let C be a circuit of m qubits and s balanced gates of minphase $1/K = 2^{-k}$ and maximum arity $r \leq m$ qubits. Then we can efficiently build a Boolean formula ϕ_C of size $O(m + sk2^{2r})$ in variables $\vec{w}, \vec{x}, \vec{y}, \vec{z}$ and find a constant R such that for all $a, b \in \{0,1\}^m$:*

$$\begin{aligned}\langle a|\ C\ |b\rangle =& \frac{1}{R}\sum_{L=0}^{K-1} \#sat(\phi_C[\vec{w} = L, \vec{x} = a, \vec{z} = b])\omega^L \\ =& \quad \frac{1}{R}\sum_{L,c} \phi_C(\vec{w} = L, \vec{x} = a, \vec{y} = c, \vec{z} = b])\omega^L.\end{aligned}$$

Moreover, for every assignment (a, c) to $\vec{x}, \vec{y}$ there is exactly one pair (L, b) such that $\phi_C[\vec{w} = L, \vec{x} = a, \vec{y} = c, \vec{z} = b]$ holds and it can be found in $O(s)$ time.

Proof. We track paths in stages $\ell = 1$ to s as they begin in a column $a = J_0 \in \{0,1\}^m$ of U_1 and terminate in row $b = I_s$ of U_s. We allocate $s+1$ suites $W_0, \ldots, W_s$ of variables $w_{0,\ell}, \ldots, w_{k-1,\ell}$ which collectively track the phase $L \in \{0, \ldots, K-1\}$ of a path by $L = \sum_{j=0}^{k-1} w_{j,\ell} 2^j$. At each stage ℓ we identify m location references $u_1, \ldots, u_m$ on the qubit lines whose values determine an entry column $J \in \{0,1\}^m$ to the matrix U_ℓ. Initially the u_i refer to the input variables $x_1, \ldots, x_m$, $R = 1$, and $w_{0,0} = \cdots = w_{k-1,0} = 0$. The u_i are not actual literals. For stage ℓ we allocate up to m fresh variables $y_1, \ldots, y_m$ whose values $I \in \{0,1\}^m$ stand for possible exit rows I_ℓ, which become either the entry column $J_{\ell+1}$ for the next stage or are equated with the output variables $z_1, \ldots, z_m$. [1]

If J is any column value in $\{0,1\}^m$, then u_J denotes the unique conjunction of signed literals $\pm u_i$ (over $i = 1$ to m) whose value is 1 on J and 0 for all $J' \neq J$. For instance, if

[1]Note that the "u_i" will be meta-symbols, and extended constructions will allow them to be negated variables. We will also later distinguish between allocated variables y_i whose values are forced not free, calling them v_i instead.

$J = 01101$ then $u_J = (\bar{u}_1 \wedge u_2 \wedge u_3 \wedge \bar{u}_4 \wedge u_5)$. We denote row conjuncts v_I similarly. Entering stage ℓ of the circuit, we consider all possible current phases $p_{\ell-1}$ coded by the variables $W_{\ell-1} = w_{0,\ell-1}, \dots, w_{k-1,\ell-1}$. For all pairs I, J we add clauses as follows:

- If $U_\ell[I, J] = 0$ then we add $\neg(u_J \wedge v_I)$, which becomes a clause of $2m$ disjoined literals.
- If $U_\ell[I, J] = re^{i\theta}$, then by balance, $r \neq 0$ is independent of I, J and $\theta = 2\pi i d/K$ for some $d < K$. Then R is multiplied by r and we add for $j = 0$ to $k - 1$ the clause
$$((u_J \wedge v_I) \rightarrow (w_{j,\ell} = w_{j,\ell-1} \oplus F_d(W_{\ell-1}))),$$
where F_d is the fixed function true on all $c < K$ such that $c + d$ causes a flip in bit j.

Note that F_d can be a function of the variables $w_{0,\ell-1}, \dots, w_{j,\ell-1}$ alone. We can alternately consider that over $j = 0$ to $k - 1$ alone we have added the single clauses

$$w_{j,\ell} = w_{j,\ell-1} \oplus F'(u_1, \dots, u_m, v_1, \dots, v_m, w_0, \dots, w_j),$$

where F' takes into account all the phases d that arise in the matrix entries $U_\ell[I, J]$ as specified by the value J for $u_1, \dots, u_m$ and I for $v_1, \dots, v_m$. Economizing F' will occupy much of the remainder of the paper, but for this proof we reason about F_d for all the u_J and v_I.

Finally we note that $v_1, \dots, v_m$ become "$u_1, \dots, u_m$" for the next stage if there is one, else we conjoin the clauses $\wedge_{i=1}^{m}(v_i = z_i)$ (or just substitute $z_1, \dots, z_m$ directly). The last act is to add the clauses $\wedge_j \bar{w}_{j,0}$ and declare $\vec{w}$ in the theorem statement to refer to the terminal $w_{j,s}$ phase variables. Then $\vec{y}$ in the theorem statement ranges over $w_{j,\ell}$ for $1 \leq \ell \leq s - 1$ and variables $v_{i,\ell}$ introduced as "v_i" in the corresponding stages ℓ. (We will pin it down further in specific instances later.) This finishes the construction of ϕ_C.

To see that it is correct, first consider any path P from a to b whose phase changes by L. First we substitute $\vec{x} = a$ and $\vec{z} = b$ and $W_s = L$. In the base case $s = 0$ with empty circuit, P can only be a path from a to $b = a$ with $L = 0$. Then we have $W_s = W_0$ and substituting L gives $\top$ if $b = a$ and $L = 0$, $\perp$ otherwise. For $s \geq 1$, to P there corresponds a unique assignment of row and column values

$$a = J_1, \quad I_1 = J_2, \quad \dots, \quad I_{s-1} = J_s, \quad I_s = b$$

to literals designated "u_i" and "v_i" at each stage ℓ. For all $(I, J) \neq (I_\ell, J_\ell)$, all clauses $(u_I \wedge v_J) \rightarrow \dots)$ are vacuously satisfied. This leaves the clause

$$((u_{J_\ell} \wedge v_{I_\ell}) \rightarrow (w_{j,\ell} = w_{j,\ell-1} \oplus F_d(W_{\ell-1}))),$$

where d is the phase of the nonzero entry $U_\ell[I, J]$. By induction, the values of $W_{\ell-1}$ in the assignment either have the phase c of the path entering that stage or the assignment is already determined to be unsatisfying. These determine the value $F_d(W_{\ell-1})$ and hence collectively over j these clauses determine that W_ℓ must have the correct value $c + d$ modulo K, else they are not satisfied. Since the values of the variables in W_ℓ are forced, we have a unique continuation of a satisfying assignment. In the last stage, the current phase value must become L. Hence we have mapped P to one satisfying assignment of $\phi_C[\vec{x} = a, \vec{z} = b, \vec{w} = L]$ (with W_0 already substituted to zeros).

Going the other way, suppose y is any satisfying assignment to ϕ_C (again with $W_0 = 0$). We argue that y maps uniquely to a path P_y. We get $a = J_1$ from the values assigned to $\vec{x}$, then the values $J_2, \ldots, J_s$ of the other column entries, and finally the exit row I_s which gives a b. The values of phases along the path are likewise determined by the assignment and must be correct. Hence the assignment yields a unique path. The path must be legal: at any stage the left-hand side of one clause of the form $(u_J \wedge v_I) \rightarrow \ldots$ holds so its consequent must be made true.

Thus the correspondence of counting paths and counting satisfying assignments is parsimonious for each phase value L, so the equation in Theorem 5.3 follows. Finally, we may observe that if U_ℓ is a tensor product of a $2^r \times 2^r$ matrix and identity matrices, then whenever I and I' vis-à-vis J and J' agree on the r qubit lines touched by the gate, their clauses can be identified, leaving at most 2^{2r} distinct clauses added at stage ℓ. The rest of the size estimation is straightforward. □

As already remarked, the main purpose of the work in this chapter is to find the most economical (and elegant) formulas for specific families of quantum gates. We also note that any initialization L_0 can be used for W_0 provided the corresponding target for W_s is shifted to be $L + L_0$. Here we finish this section by noting one further general feature of the emulation that already follows from this proof.

For any set B of target output values and phase L, we can define $p_C^L(a, B) = \sum_{b \in B} p_C^L(a, b)$, where $p_C^L(a, b)$ denotes the number of paths from a to b having phase L. We will find it convenient to maintain these sets of paths when B is a cylinder (as defined in Definition 2.2). Singleton sets $\{ b \}$ have this form with $I = \{ 1, \ldots, m \}$ and $c = b$, as do sets $B_i = \{ b : b_i = 1 \}$ which represent measuring the single qubit i to test for a 1 value. Cylinders are important because we can choose *not* to substitute all $z_1, \ldots, z_m$ variables by values $b_1, \ldots, b_m$.

Note must however be taken that a path to b and path to b' do not interfere—because they have different "locations." Hence in particular, taking weighted sums of $p_C^L(a, B)$ is not the same as measuring outcomes in B. One needs to sum them for all $b \in B$. We will fix this issue by proving a parallel theorem for the acceptance probability. We state it here just for circuits of gates whose phases are multiplies of i:

Theorem 5.4. *Let C be a circuit of m qubits and s balanced gates of min-phase $\pi/2$ and maximum arity $r \leq m$ qubits. Then we can efficiently build a Boolean formula ψ of size $O(m + s2^{2r})$ in variables $\vec{v}, \vec{w}, \vec{x}, \vec{y}, \vec{z}$ and find a constant R such that for all $a \in \{0,1\}^m$ and cylinders $B \subseteq \{0,1\}^m$:*

$$\sum_{b \in B} |\langle a| \; C \; |b\rangle|^2 = \frac{1}{R^2}(\#sat(\psi') - \#sat(\psi'')), \tag{5.3}$$

where ψ' and ψ'' are projections of ψ depending on B. Moreover, for every assignment (a, c) to $\vec{x}, \vec{y}$ there is exactly one completion to an assignment that satisfies ψ' or ψ'' (never both) and it can be found in $O(s)$ time.

It is best to prove this after (in Section 5.6) gaining a concrete understanding of the efficiency issues for the cases $K = 2, 4, 8$ and the motivation for sampling and uniform generation. The next section segregates the variables $\vec{y}$ of the general case into *Hadamard variables* $y_1, \ldots, y_h$ and other variables $v_1, \ldots, v_{s-h}$, whence "$\vec{v}$" in the above statement.

5.4 Circuit Simulation By "Controlled-Bitflip" Clauses

Now we show how the construction of Theorem 5.3 when applied to common quantum gates, yields Boolean equations e of the controlled-bitflip kind:

$$p' = p \oplus \wedge_{i=1}^{j} u_i. \tag{5.4}$$

When $j = 0$ this becomes $p' = \neg p$ (not $p' = p$) since an empty AND defaults to `true`. So then when $j = 1$ and $u_1 = 1$, it is a *bitflip.* In general, with the u_i's presence, 5.4 is essentially a *controlled bitflip.* All u_i's in the aggregate term $\wedge_{i=1}^{j} u_i$ are control bits, and p is the target. But in the implementation of our prototype simulator, we introduces new variables to restrict the length of $\wedge_{i=1}^{j} u_i$ to be 2. More details can be seen in the examples in Section 5.5.

The fact that CNOT and Toffoli gates alongside Hadamard gates have a universal set says that all quantum computers can be represented as using only controlled bitflips and quantum measurements. The essence of our Boolean equations is that all quantum gate behavior and the dense set of complex amplitudes can be naturally described via controlled bitflips *alone.* Moreover, the controls naturally alter "phase sign". The literals in e are all positive. Negative literals enter only when e is converted into (e.g.) conjunctive normal form. We call a clause of the form (5.4) a *controlled-Bit-Flip equation*, *cBF* for short, of *arity j*.

Lemma 5.5. *A cBF of arity j is equivalent to a conjunction of two $(j + 2)$-clauses and $2j$ 3-clauses.*

Proof. The two j-clauses are $(\bar{u}_1 \vee \cdots \bar{u}_j \vee p \vee p')$ and $(\bar{u}_1 \vee \cdots \bar{u}_j \vee \bar{p} \vee \bar{p}')$. They make p and p' have opposite sign in case all u_i are true. The 3-clauses are $(u_i \vee p \vee \bar{p}') \wedge (u_i \vee \bar{p} \vee p')$ for $i = 1$ to j. They make p and p' equal in case some u_i is false. $\square$

We show next that all of the most commonly used quantum gates—including those mentioned in the recent frontier references [HSST16, BIS$^+$16, HS17]—translate efficiently controlled-bitflip equations. The entries of their defining matrices are all powers of $\omega = e^{i\pi/4}$ which we identify with the *phases* $0, \ldots, 7$ (modulo 8). First we describe the Boolean variables in full:

- Variables $x_1, \ldots, x_m$ are input variables assigned $a_1, \ldots, a_m$. The *line designators* $u_1, \ldots, u_m$ initially denote $x_1, \ldots, x_m$, respectively.
- The output variables $z_1, \ldots, z_m$ are respectively set equal to the variables designated by $u_1, \ldots, u_m$ upon finishing the circuit, and may be substituted by any subset of target output values $b_1, \ldots, b_m$ to fix measurements.
- The top-phase variables $p_0, \ldots, p_\ell, \ldots$ distinguish the top of the circle (phases 0,1,2,3 for $p_\ell = 0$) from the bottom of the circle (phases 4,5,6,7 for $p_\ell = 1$).
- Quarter-phase variables $q_0, \ldots, q_\ell, \ldots$ distinguish the quadrants 0,1 and 4,5 ($q_\ell = 0$) from 2,3 and 6,7 ($q_\ell = 1$).
- The variables $r_0, \ldots, r_\ell, \ldots$ tell whether the current phase of a path is even ($r_\ell = 0$) or odd ($r_\ell = 1$).
- *Free variables* $y_1, \ldots, y_h, \ldots$ each represent a bit of nondeterminism. Initially $h = 0$, $R = 1$.
- *Bound variables* $v_1, \ldots, v_s, \ldots$ are placed on qubit lines i and become the new u_i when placed.

The triple (p_ℓ, q_ℓ, r_ℓ) combines to specify the current phase of a path, while the vector of values of the variables currently pointed to by $u_1, \ldots, u_m$ represents the path's current *location.* All paths on input $a = (a_1, \ldots, a_m)$ begin at location a with phase 0 represented by the initializations $p_0 = 0$, $q_0 = 0$, and $r_0 = 0$. To illustrate the encoding scheme for phases, consider counter-clockwise rotations by π, $\pi/2$, and $\pi/4$ in the unit circle when the current phase is denoted by $(p_{\ell-1}, q_{\ell-1}, r_{\ell-1})$:

- By π: $p_\ell = \neg p_{\ell-1}$; no change to $q_{\ell-1}, r_{\ell-1}$.
- By $\pi/2$: $q_\ell = \neg q_{\ell-1}$; $p_\ell = p_{\ell-1} \oplus q_{\ell-1}$; no change to $r_{\ell-1}$.

- By $\pi/4$: $r_\ell = \neg r_{\ell-1}$; $q_\ell = q_{\ell-1} \oplus r_{\ell-1}$; $p_\ell = p_{\ell-1} \oplus (q_{\ell-1} \wedge r_{\ell-1})$.

Note how the update to p_ℓ under rotation by $\pi/4$ represents a two-place "carry" when incrementing the phase in binary notation with $r_{\ell-1}$ holding the least significant bit. A rotation by $\pi/8$ would involve new phase variables s_ℓ distinguishing parity modulo 16 with r_ℓ re-defined as false for phases $0, 1, 4, 5, 8, 9, 12, 13$ and true for $2, 3, 6, 7, 10, 11, 14, 15$. None of these changes the location, and in consequence applying these rotations does not change the amplitude of any output b on the qubit lines. Thus the *cBF* format captures the effect of carrying.

Gates with diagonal matrices do not change the location or allocate a new line variable y_r or v_r. Gates whose non-zero entries are all 1 (possibly divided by a normalizing constant) do not change any phase variable, while those with non-zero off-diagonal entries can change the location as well as phase. Here are the details for individual gates:

- Hadamard gate $\mathsf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ on line i: Allocate new top-phase variable p_ℓ, allocate new free variable y_h on line i, add equation $p_\ell = p_{\ell-1} \oplus (u_i \wedge y_h)$, set $u_i := y_h$, and multiply R by $\sqrt{2}$.

- Pauli gate X (aka. NOT) $= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ on qubit line i: Allocate new line variable v_s on line i, add equation $v_s = \neg u_i$, update $u_i := v_s$, no change to other variables or R. (Alternately we could just flip the sign on u_i, but we prefer to keep it referring to a variable—in our C++ code it is an array pointer.)

- CNOT (aka. CX or just C) $= \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{X} \end{bmatrix}$ with *control* on line i and *target* on line j: Allocate new v_s on line j, add equation $v_s = u_j \oplus u_i$, update $u_j := v_s$, no other changes.

- Toffoli gate Tof (aka. CCX) $= \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{CX} \end{bmatrix}$ with controls on i, j and target on k: Allocate new v_s on line k, add equation $v_s = u_k \oplus (u_i \wedge u_j)$, update $u_k := v_s$, no other changes.

- Pauli gate $\mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ on qubit line i: Allocate new top-phase variable p_ℓ, add equation $p_\ell = p_{\ell-1} \oplus u_i$, no other change.

- Pauli gate $\mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ on qubit line i: Since $\mathsf{Y} = i\mathsf{XZ}$, we can compose the actions for Z and X, with the final scalar multiplication by i being optional.

- Phase gate $\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$: Allocate new p_ℓ, q_ℓ with equations $q_\ell = q_{\ell-1} \oplus u_i$ and $p_\ell = p_{\ell-1} \oplus (u_i \wedge q_{\ell-1})$. No other change.

- $\mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & \omega \end{bmatrix}$, where $\omega = e^{i\pi/4} = \frac{1+i}{\sqrt{2}}$: Allocate all new p_ℓ, q_ℓ with equations:

$$\begin{aligned} r_\ell &= r_{\ell-1} \oplus u_i \\ q_\ell &= q_{\ell-1} \oplus (r_{\ell-1} \wedge u_i) \\ p_\ell &= p_{\ell-1} \oplus (q_{\ell-1} \wedge r_{\ell-1} \wedge u_i). \end{aligned}$$

- V (aka. $\sqrt{\mathsf{NOT}}$ or $\mathsf{X}^{1/2}$) $= \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \omega & \omega^7 \\ \omega^7 & \omega \end{bmatrix}$: Directly transcribing this per the last section does not yield a single *cBF*, but we can use $\mathsf{V} = \mathsf{HSH}$ which does it with four *cBF*'s.

- $\mathsf{Y}^{1/2} = \frac{1+i}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$: Use $\mathsf{Y} = \mathsf{HZ} \cdot \omega$, where again the scalar multiplication by ω (i.e., counter-clockwise phase shift by $\pi/4$) can be ignored.

- $\mathsf{CS} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{S} \end{bmatrix}$ with source i and target j: Allocate p_ℓ, q_ℓ and add equations $q_\ell = q_{\ell-1} \oplus (u_i \wedge u_j)$ and $p_\ell = p_{\ell-1} \oplus (u_i \wedge u_j \wedge q_{\ell-1})$.

- $\mathsf{CV} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{V} \end{bmatrix}$ with source i and target j: This is equavalent to placing H on line j, then CS with source i and target j, and finally H on line j again.

- $\mathsf{CZ} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{Z} \end{bmatrix}$ with source i and target j: As with CS (and with an equivalent formula for CV), it just adds u_i as a conjunct to the equation for Z on line j.

5.5 Examples and Execution on Our Simulator

Ultimately all the *free* variables are assigned when placing Hadamard gates. The sequence of *cBF*'s, as gates are placed in left-to-right order (with matrices composed in right-to-left order), obeys the following invariant:

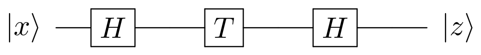
Lemma 5.6. *For any truth assignment $c = (c_1, c_2, \ldots, c_h)$ to the Hadamard variables $y_1, \ldots, y_h$, input $a = a_1 \cdots a_m$ to the variables x_i, and initialization of p_0, q_0, r_0, the sequence of cBF's can be evaluated in order with all right-hand side values defined in the initialization or in previous steps.* □

This enables an "intelligent backtrack" brute-force solution counting routine that, when incrementing $c \in \{0, 1\}^h$ to the next c' in standard order, need only roll back to the first cBF containing y_g, where c and c' agree in the first $g - 1$ bits. Roughly speaking, this saves a factor of h when carrying out the brute-force iteration through c to tabulate the results of each path. This idea is implemented in a prototype of our simulator.

From Section 5.4, T gate uses one 5-ary cBF, which becomes two 5CNF clauses under Lemma 5.5 and a conversion similar to 5.1. In the prototype of our simulator, we restrict all controlled-bitflip equations to have only two control bits and this enables all clauses to be eventually in 3CNF or 4CNF forms (as discussed in 5.2 and 5.1). Hence before presenting our experimental results, we show some examples of the construction of Boolean formulas and discuss how those controlled-bitfilp equations transform eventually in our simulator.

5.5.1 Single-qubit Example

Section 2.1.4 shows a basic but typical example of single-qubit circuit H-T-H.



With the techniques developed in Section 5.3, this single-qubit circuit will be converted into the following set of Boolean formulas, with the initializations $p_0 = 0, q_0 = 0$, and $r_0 = 0$:

phase $\frac{\pi}{4}$:	phase $\frac{\pi}{2}$:	phase π:
$r_1 \equiv r_0 \oplus y_0$	$q_1 \equiv q_0 \oplus (r_0 \wedge y_0)$	$p_1 \equiv p_0 \oplus (y_0 \wedge x)$
		$p_2 \equiv p_1 \oplus (q_0 \wedge r_0 \wedge y_0)$
		$p_3 \equiv p_2 \oplus (y_1 \wedge y_0)$.

In the implementation of our simulation system, the equation $p_2 \equiv p_1 \oplus (q_0 \wedge r_0 \wedge y_0)$ becomes

$$p_2 \equiv p_1 \oplus (q_0 \wedge t) \wedge (t \equiv r_0 \wedge y_0)$$

which is similar to 5.2.

Let r_i's be variables associated with phase $\frac{\pi}{4}$, q_i's for $\frac{\pi}{2}$, and p_i's for π. Below Figure 5.1 shows how the set of Boolean formulas looks like in our simulator.

```
p1 = p0 XOR (y0 && x1)
r1 = r0 XOR (y0)
c0 = r0 XOR (r1)
q1 = q0 XOR (r0 && c0)
c1 = q0 XOR (q1)
p2 = p1 XOR (q0 && c1)
p3 = p2 XOR (y1 && y0)
```

Figure 5.1: Boolean formulas for single-qubit circuit H-T-H.

The main difference from above lies in $p_2 \equiv p_1 \oplus (q_0 \wedge r_0 \wedge y_0)$ which becomes $p_2 \equiv p_1 \oplus (q_0 \wedge c_1)$ in the simulator, with $c_1 \equiv r_0 \wedge y_0$. Hence all the controlled-bitflip equations can convert into 3CNF or 4CNF forms.

We know that $|\langle 0|C|0\rangle|^2 \approx 0.85$ and $|\langle 1|C|0\rangle|^2 \approx 0.15$. This disparity cannot be explained by saying that each H acts like a classical random coin. This phenomenon is made more formal in the CHSH game (referred to Section 2.1.4). In particularly, Bell's Theorem behind the CHSH game is often interpreted as saying that one *cannot* use classical binary random variables to analyze quantum probability. However, this work shows that one *can* use such variables *internally*–one classical Boolean variable per H gate–provided that the *external analysis* is correct. Here the external factors are the cosine terms in Theorem 5.3.

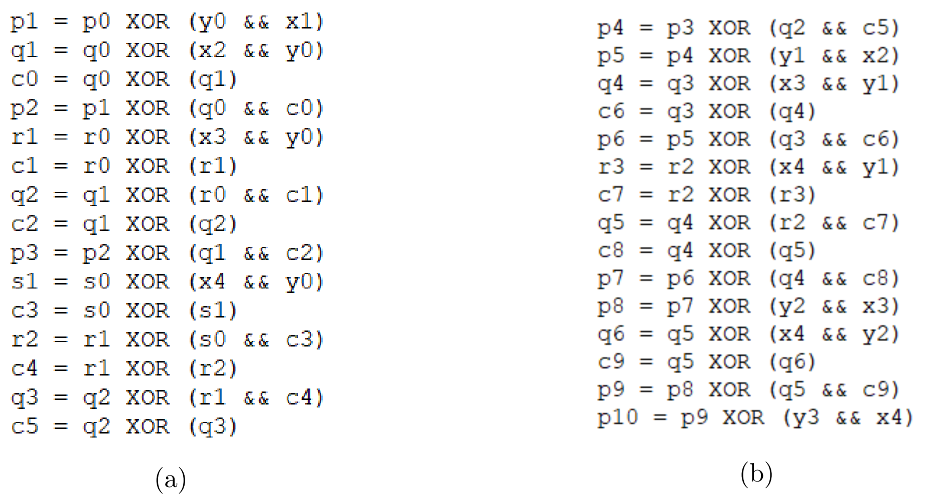
5.5.2 Quantum Fourier Transformation (QFT)

Adding variables standing for phases $1/16, 1/32, \ldots$ can code an even wider range of gates used in some well-known exact recursions for the *quantum Fourier transform* on n qubits, whose matrix is given by

$$\mathbf{QFT}_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

where $N = 2^n$ and $\omega = e^{-2\pi i/N}$. (It is worth noting that with $K = 8$ this is the same ω as used in defining T and V above.) However, these recursions involve $\Theta(n^2)$ gates each needing $\Theta(n)$-many *cBF*'s of arity $\Theta(n)$, giving formulas of size $\Theta(n^4)$ overall. An asymptotically better way notes that if the incoming phase is H then the new phase is $H' = H + I \cdot J$ modulo N. This string relation has Boolean circuits of size $O(n \log n \log\log n)$ via the Schönhage-Strassen integer multiplication algorithm [SS71] and its conversion to circuits. (Ironically, this uses F_{2n}.) Concrete quantum circuits for the relation were presented by Markov and Saeedi [MS12], but we can simply convert each Boolean circuit gate into a *cBF*.

The construction of the circuit for QFT over 4 qubits is the one in Figure 2.1. For the sake of illustration, we also show its set of Boolean formulas generated by our prototype in Figure 5.2.



```
p1 = p0 XOR (y0 && x1)
q1 = q0 XOR (x2 && y0)
c0 = q0 XOR (q1)
p2 = p1 XOR (q0 && c0)
r1 = r0 XOR (x3 && y0)
c1 = r0 XOR (r1)
q2 = q1 XOR (r0 && c1)
c2 = q1 XOR (q2)
p3 = p2 XOR (q1 && c2)
s1 = s0 XOR (x4 && y0)
c3 = s0 XOR (s1)
r2 = r1 XOR (s0 && c3)
c4 = r1 XOR (r2)
q3 = q2 XOR (r1 && c4)
c5 = q2 XOR (q3)
```

(a)

```
p4 = p3 XOR (q2 && c5)
p5 = p4 XOR (y1 && x2)
q4 = q3 XOR (x3 && y1)
c6 = q3 XOR (q4)
p6 = p5 XOR (q3 && c6)
r3 = r2 XOR (x4 && y1)
c7 = r2 XOR (r3)
q5 = q4 XOR (r2 && c7)
c8 = q4 XOR (q5)
p7 = p6 XOR (q4 && c8)
p8 = p7 XOR (y2 && x3)
q6 = q5 XOR (x4 && y2)
c9 = q5 XOR (q6)
p9 = p8 XOR (q5 && c9)
p10 = p9 XOR (y3 && x4)
```

(b)

Figure 5.2: Boolean formulas QFT over 4 qubits

The construction follows the convention rules in Section 5.4 together with one extra step (as analogous to the single-qubit example above) which introduces new c_i's variables to shorten the number of control bits. This makes all controlled-bitflip forms have only two direct control bits at the expense of extra controlled-bitflip equations, and it results in the set of controlled-bitflip equations as shown collectively in Figure 5.2. From the ideas in Section 5.4, variables p_i's are associated with phase π, q_i's with phase $\frac{\pi}{2}$, r_i's with phase $\frac{\pi}{4}$, and s_i's with phase $\frac{\pi}{16}$. They are enough because from the matrix representation, this QFT will be $\mathbf{QFT}_N$ with $N = 2^4$, where $\omega = e^{-2\pi i/16}$, and the finest phase for the amplitude $\langle a|\ C\ |b\rangle$ can only go as small as $\frac{\pi}{16}$.

Generally, let $b(c)$ denote the final location of a path c and $f(c)$ the final phase. The amplitude of the *outcome* $C(a) = b \in \{0, 1\}^m$ is given by

$$\langle a|\ C\ |b\rangle = \sum_{c:b(c)=b} \omega^{f(c)} = \sum_{J=0}^{N-1} |\{\, c : b(c) = b \wedge f(c) = J \,\}|\omega^J \tag{5.5}$$

with $\omega = e^{2\pi i/N}$. What the right-hand side of (5.5) finally means is that we can get the amplitude by counting the number of satisfying assignments to each of the formulas ϕ_J obtained from the basic $\phi = \phi_C$ by substituting the binary representation of each J for the phase variables.

5.5.3 Demo of Shor's Algorithm

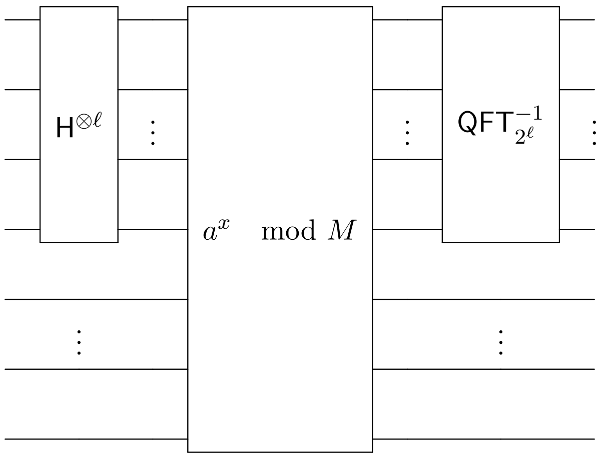


Figure 5.3: The quantum subroutine in Shor's algorithm.

Figure 5.3 shows the main quantum component in Shor's algorithm [Sho94] for factoring a given n-bit integer M starts by choosing $Q = 2^\ell$ where $\ell = 2n+1$, so that $M^2 < Q < 2M^2$, and a random $a < M$ which we may presume is relatively prime to M (else it succeeds at once). On the left hand side, there are $2n$ Hadamard gates being applied to $2n$ qubits separately, while the right hand side is a QFT over $2n$ qubits. In the middle, it places a deterministic circuit C_0 that maps any binary-encoded number $x < Q$ to $f_a(x) = a^x \pmod{M}$. More precisely, C_0 maps $x \cdot 0^\ell$ to the concatenation $x \cdot y$ where $y = f_a(x)$ as an ℓ-bit number. The combination with the Hadamard gates creates the *functional superposition*

$$\Phi = \frac{1}{\sqrt{Q}} \sum_x |x f_a(x)\rangle \, .$$

The quantum part then applies QFT_ℓ to the first ℓ lines and measures them to get an output $b < Q$. The rest is a classical attempt to use b to find a *period* r such that $f_a(x) = f_a(x+r)$ for all x, and then use r to find a factor of M to tumble out. Failure means re-starting the outer loop with another a', but the analysis shows that with high probability it needs only $O(\log n)$ restarts. The sampling routine in Section 5.6 can emulate the inner loop. It will be interesting to see if this can attack moderately large numbers M. The brute-force method is OK for $M < 100$. For comparison, `libquantum` [BW03, WML$^+$10, WB11] complied on the same hardware can work up to about $M = 5000$. More is discussed in Section 5.7.

Since the conversion of Hadamard gate and QFT have been discussed in preview sections (Section 5.4 and Section 5.5.2), we restrict our attention to the modular exponentiation gate ($a^x \mod N$) in this section. Instead of simulating this classical $a^x \mod N$ directly, we emulate this gate with Montgomery multiplication [Mon85] which has been used to implement Shor's algorithm before [PG12, RNSL17]. Montgomery multiplication computes modular exponentiation using only addition, multiplication, and operations modulo a power of 2, and hence it is often the most efficient choice if the modulus is not being too close to a power of 2.

5.6 Probability Form and Sampling

We give a revision of the main theorem tailored for the probability on a cylinder (Definition 2.2), rather than amplitude. The advantage is that it can be used for *sampling*.

To discuss sampling, first we review the classical reduction from counting to uniform generation. It starts with one call to $\#sat$ to compute the cardinality of the solution set S.

- Using one call to $\#sat$, compute $|S_0| = |\{ x \in S : x_1 = 0 \}|$ and $|S_1| = |\{ x \in S : x_1 = 1 \}| = |S| - |S_0|$.
- Set $x_1 = 0$ with probability $|S_0|/|S|$ and $x_1 = 1$ otherwise.
- Substitute the value of x_1 into ϕ and recurse on x_2 and so on.

The *quantum sampling* task is to generate outputs b—belonging to $\{0,1\}^m$ or some (cylindrical) subset B thereof—according to the probability distribution $|\langle a| C |b\rangle|^2$. A naive attempt to emulate the above process beginning with $B = (\{1\}, 0)$ would substitute $z_1 = 0$ but leave the variables $z_2, \dots, z_m$ open in the first formula ϕ_0. Applying the counting in (5.3) to the phase-shifted formulas derived from ϕ_0 would fail, however, because it would attempt to cancel counts of solutions with different final locations $b_2, \dots, b_m$ whose waves do not interfere. The fix is to maintain the probabilities directly rather than the amplitudes in a way that preserves the cylindrical structure. We state the binary case first:

Theorem 5.7. *Let C be a circuit of m qubits, h Hadamard gates, and $s - h$ Toffoli and CNOT gates. Then we can efficiently build a Boolean formula ψ_C of size $O(m+s)$ in variables $\vec{v}, \vec{v'}, \vec{w}, \vec{w'}, \vec{x}, \vec{y}, \vec{y'}, \vec{z}$ and find a constant R such that for all $a \in \{0,1\}^m$ and cylinders $B \subseteq \{0,1\}^m$, $\Pr[C(a) \in B] = \sum_{b \in B} |\langle a| C |b\rangle|^2 =$*

$$\frac{1}{2^h}(\#sat(\psi_C[\vec{x} = a, \vec{z} = b] \wedge w_h = w'_h) - \#sat(\psi_C[\vec{x} = a, \vec{z} = b] \wedge w_h \neq w'_h). \tag{5.6}$$

Moreover, for every assignment (a, c, c') to $\vec{x}, \vec{y}, \vec{y'}$ there is exactly one completion to an assignment that satisfies ψ_C and it can be found in $O(s)$ time.

The proof is subsumed by that of Theorem 5.4. Note that for $+i, -i$ phases, we get exactly the same equation since i and $-i$ cancel. Now we can make the uniform generation procedure work by using $\psi_0 = \psi_C[z_1 = 0]$ in place of ϕ_0. We need two calls to the $\#sat$ oracle to evaluate $\#sat(\psi_0 \wedge w_h = w'_h)$ and $\#sat(\psi_0 \wedge w_h \neq w'_h)$. Their difference over 2^h gives the correct probability because by summing over assignments to $z_2, \dots, z_m$ we are summing (5.6) over $b \in B_0$. We do not need to make separate calls for the case $z_1 = 1$. Thus the efficiency in the number of oracle calls is the same as in the classical iterative reduction. The resulting binary string b is generated with the same probability distribution $\mathcal{D}_C$ as measuring all registers of $C(a)$ would give, and the time needed is $O(mhT_{2s+m})$ where T_n is the time for calls to the $\#sat$ oracle on an n-variable formula.

A second elegant point, after what we have noted about the equations for Hadamard *vis-à-vis* Toffoli gates, is that ψ_C is virtually the same as $\phi_{C'}$ for the circuit C' that computes the standard "compute-uncompute trick" [BBC+95]. For $K = 4$ we again get a difference of two calls to $\#sat$:

Proof of Theorem 5.4. Here by the constructions in Section 5.4 the formula ϕ_C has variables p_h, q_h denoting the final phase, with $p_h = 0$ for 1 and i versus $p_h = 1$ for -1 and $-i$, and $q_h = 0$ for $1, -1$ versus $q_h = 1$ for $i, -i$. Again we make a copy ϕ'_C with final phase variables $p'_h < q'_h$. For a final state $\alpha = a + bi - c - di$ we have

$$|\alpha|^2 = (a^2 + b^2 + c^2 + d^2) - (2ac + 2bd).$$

The positive term is expressed by conjoining $(p'_h = p_h) \wedge (q'_h = q_h)$. The negative term is expressed by the combinations $(p_h q_h, p'_h q'_h) = (00, 10)$, $(10, 00)$, $(01, 11)$, or $(11, 01)$. The conjunction allowing exactly these combinations is $(p'_h \neq p_h) \wedge (q_h = q'_h)$. □

5.7 A Few Experimental Results

The tasks of counting the number of solutions to a given polynomial equation and the number of satisfying assignments to a Boolean formula belong to #P as defined above. The general cases of these tasks are both NP-hard. Although NP-hardness has generally been regarded as strong evidence of *asymptotic* intractability, recently there have been broad advances on solving *concrete* cases of these tasks. Most of this success has come from so-called SAT-solvers asked to find just one satisfying assignment, but #SAT solvers charged with counting the number of satisfying assignments *exactly* have gained traction.

We show the experimental results from the performance of solving the generated boolean formulas using our brute-force (BF) method and current versions of the *Cachet* [SBB+04, SBK05a, SBK05b] and *sharpSAT* [Thu06] solvers. For consistency, we use m to denote the

Table 5.1: CNOT staircase (microsecond (us))

m	BF	sharpSAT	Cachet
4	34.817	7409.250	35869.500
8	618.083	8020.500	36244.500
12	12238.524	8292.250	35244.875
16	122063.122	8099.500	34120.375
20	2101033.678	9594.000	39993.500
24	62935719.848	9024.125	42994.000

number of qubit lines in a quantum circuit. All the circuits we are testing on are first having a bank of Hadamard-gates applied to the input vector, and subsequently differ as follows:

- CNOT staircase: apply CNOT gates with control n and target $n+1$ followed by a T-gate on line n for $n = 1, \cdots, m-1$. Concretely, a sequence of applied gates would be $\mathsf{CNOT}(1,2), \mathsf{T}(2), \mathsf{CNOT}(2,3), \mathsf{T}(3), \ldots, \mathsf{T}(m), \mathsf{CNOT}(m,1)$.
- CNOT staircase with appended CZ and CV gates that alternate: After the bank of H gates and the above CNOT staircase, apply CV and CZ alternately to lines $n, n+1$ and $n+2$. Concretely, a sequence of applied gates would be:

$$\begin{aligned}&\mathsf{CNOT}(1,2), \mathsf{T}(2), \mathsf{CNOT}(2,3), \mathsf{T}(3), \ldots, \mathsf{T}(m), \mathsf{CNOT}(m,1)\\ &\mathsf{CV}(1,2), \mathsf{CZ}(2,3), \mathsf{CV}(3,4), \mathsf{CZ}(4,5), \cdots, \mathsf{CZ}(n-2,n-1), \mathsf{CV}(n-1,n).\end{aligned}$$

- Initial segments of circuits proposed in [BIS$^+$16] to be hard for classical simulations.

The following results were run from C++ code on a Dell PowerEdge R720 departmental machine. Our code represents each element of a *cBF* by a pointer to an unsigned integer standing for a Boolean value. The brute-force times are single-threaded; the others use compilations of the official current source code releases of *sharpSAT* and *Cachet*.

Table 5.1 shows the results of solving the generated boolean formulas for circuits consisting of a "staircase" of m Hadamard and CNOT gates producing entanglements. While the brute-force (BF) running time grows exponentially in m as expected, the running times of *sharpSAT* and *Cachet* change little. This suggests that the solvers are able to figuratively flatten the staircase so that the transformed solutions are easy to count. The next experiment tries to frustrate this by sprinkling controlled gates of non-binary phases amid the lines. The CV gates add extra nondeterminism in the standard basis.

Table 5.2: CNOT staircase with CZ and CV (microsecond (us)

m	BF	sharpSAT	Cachet
4	409.706	8,373	42,744
8	91,534.342	8,486	49,493
12	13.858 s	10,400	64,300
16	14587.857 s	18,500	52,000
20	$\gg$ 5 hours	145,700 s	106,000
24	$\gg$ 5 hours	1,196,100	362,700

In Table 5.2, the relations between them are similar to that in 5.1 until the line for $m = 16$. The BF running time balloons up even more owing to the extra nondeterministic variables. The *sharpSAT* solver seems to have special difficulty with the circuits of 20 and 24 qubits.

We also tried initial sets of layers from the circuits treated in [HS17] based on indications from [BIS$^+$16] of their being hard to simulate classically. Those circuits had too much non-determinism for BF but gave results within a few hours for *sharpSAT* and *Cachet* until the circuits reached 6 or 7 layers of 24 to 36 qubits—well short of the 40-layer simulations on massively parallel hardware announced by [HS17].

The results show that *sharpSAT* and *Cachet* give better scalability on these circuits. They as yet do not, however, even "recognize" the identity $\mathsf{HH} = I$ in the sense of having similarly close running times when extra HH pairs are added to these circuits. Of course, our BF method has its time compounded by a factor of 4 for each pair since it blindly tries all combinations. This points to the goal is tuning the solvers for a repertoire of basic quantum simplifications, in the hope that this will boost the heuristics already employed.

The final preliminary experiment, just at press time, emulated the circuits for Shor's algorithm that are constructed by `libquantum` [BW03, WML$^+$10, WB11]. The `libquantum` package and its `shor` routine are distinguished among quantum simulation software by being part of the SPEC CPU2006 benchmark suite [SBH06]. We modified the v1.1.1 release code so that it prints out each quantum gate in the readable format of our emulator. The circuits are generated specially for each M and choice of random seed a. For $M = 2021$ and $a = 7$ the circuit built by the `shor` routine uses 22 principal qubits, 35 ancillas, and has 98,135 elementary gates. By far the largest block is for the modular exponentiation step which consists entirely of deterministic gates (only NOT, CNOT, and Toffoli). They are somewhat larger than the original circuits for Shor's algorithm detailed in [BCDP96]. They are far from optimal; indeed, Markov and Saeedi [MS12, MS13] showed 6-to-8-fold improvements by high-level means and other gate-level improvements have been made [Bea03, PG14, HRS17].

The SPEC CPU2006 benchmark consists of one run of `shor` on M and a, which does just one iteration of the quantum circuit—no restarts in case it doesn't succeed. It uses a numerical gate-by-gate simulation. For M approaching 10,000 our compile of `shor` overflows its hash table of over 500MB. It functioned correctly on $M = 2021 = 43 * 47$, which is just under 2^{11}. Our emulator's brute-force routine reaches its limit for numbers larger than $2^9 = 512$, which entails running through $2^{36} = 64$ billion assignments for each of 9 sampled bits. Optimizing the initial modular exponentiation stage would make very little difference in our brute-force routine because only one in every 2^{18} assignments backtracks beyond the final QFT step which has 18 Hadamard gates of its own. Runs with the #SAT solvers succeeded for $M = 15$ and $M = 21$ but bogged down for $M = 55$, with *sharpSAT* expanding to over 34GB of system resources. This evidently owes to the second copy of ϕ in the proof of Theorem 5.7 doubling the count of Hadamard gates again. The brute-force compilation needed only the single copy and stayed within 71MB, under 0.1% of system memory, per billion assignments tried.

5.8 Conclusions

We have defined a natural *emulator* in the sense of [HSST16]. Preliminary experimental work shows that it is competent even in brute-force simulation and enables distinctly high performance through #SAT solvers in several instances. It has a high memory footprint only in the accumulation of final results. The sampling procedure of Section 5.6 essentially eliminates that footprint but at double the cost in nondeterminism and a squaring of brute-force simulation time. Overall the architecture is markedly different from that of commonly employed systems.

Higher performance may come from software advances in #SAT solvers. These might be tailored to leverage the "controlled-bitflip" form of the equational clauses before conversion to conjunctive normal form. At the very least, our work has supplied a new class of natural instances by which to challenge these solvers.

We close with an analogy to elaborate the main issue with our architecture. Solvers that represent whole state vectors in some form and emulate circuit levels sequentially figuratively have the memory footprint of a giant. Once the giant gets going, however, it walks with a steady gait. Our model instead employs an army of fleet-footed mice and can send one 'mouse' (i.e., evaluate one Feynman path) at a time with zero footprint—except for housing the results of the mice at the end. The issue is that each intermediate nondeterministic gate doubles the size of the mouse army. The brute-force simulation does intelligent backtracking but does not carry out simplifications that might reduce the implicit army.

The formulas manipulated by SAT and #SAT solvers, insofar as they expand via resolution and other techniques, are between the mice and the giant. The further success of this approach may depend on how much implicit combination they can achieve. For some sampling steps of quantum algorithms, certain tradeoffs between accuracy of the counting and size can be tolerated. How this can possibly interact with the deep tradeoff of approximation and hardness in sampling, over which the argument over "quantum supremacy" is currently centered, remains to be seen.

Chapter 6

Stabilizer Circuits, Quadratic Forms, and Computing Matrix Rank

The results in this chapter are in our paper [GR19].

6.1 Overview

The main discovery of the work in this chapter is the *tight* connection between the strong simulation of stabilizer circuits and two bedrock mathematical tasks: *computing matrix rank* and *counting solutions to quadratic polynomials* (both over the field $\mathbb{F}_2$).

In particular, we show how strong simulation of a stabilizer circuit C can be reduced to the problem of computing matrix rank with the promise that $\langle 0^n| \, C \, |0^n\rangle$ is nonzero, and the same reduction can be reverted directly, which overall gives the almost tight connection between these two problems. They can be summarized as follows:

(a) Strong simulation of n-qubit stabilizer circuits of size s with h Hadamard gates (or other nondeterministic single-qubit gates) on standard-basis inputs is in time $O(s + n + h^\omega)$ where $2 \leq \omega < 2.3729$. This works for amplitude as well as probability.

(b) Computing $n \times n$ matrix rank is linear-time equivalent to computing the probability p (for circuits where $h = \Theta(n)$ and $s = O(n^2)$) on the promise that p is positive, and equivalent to computing p on the narrower promise that the graphs underlying the circuits are bipartite.

Moreover, the proofs for item (a) and (b) respectively, imply the following two algorithms.

Consider the following algorithm for computing the rank r of an $n \times n$ matrix $\mathbf{A}_0$ over the field $\mathbb{F}_2$:

I. Form the symmetric block matrix $\mathbf{A} = \begin{bmatrix} 0 & \mathbf{A}_0 \\ \mathbf{A}_0^\top & 0 \end{bmatrix}$.

II. Form the quantum *graph state* circuit $C_\mathbf{A}$ for the bipartite graph with adjacency matrix $\mathbf{A}$.

III. Calculate $p =$ the quantum probability that $C_\mathbf{A}(0^{2n}) = 0^{2n}$. The bipartite case assures $p > 0$.

IV. Output $r = \log_2(1/\sqrt{p})$.

All steps except 3 take $O(n^2)$ time. Hence, for dense matrices, this is a linear-time reduction from r to p. In the converse direction, the following algorithm is for computing the amplitude $\langle 0^n | \, C \, | 0^n \rangle$ for any n-qubit quantum *stabilizer circuit* C:

I. Convert C to a classical quadratic form q_C over $\mathbf{Z}_4$ that retains all quantum properties of C.

II. Take the matrix $\mathbf{A}$ of q_C over $\mathbf{Z}_4$ and associate a canonical $n \times n$ matrix $\mathbf{B}$ over $\mathbb{F}_2$ to it.

III. Compute the decomposition $\mathbf{B} = \mathbf{PLDL}^\top\mathbf{P}^\top$ over $\mathbb{F}_2$ where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is lower-triangular, and $\mathbf{D}$ is block-diagonal with blocks that are either 1×1 or 2×2.

IV. Take $\mathbf{L}^{-1}$ over $\mathbb{F}_2$ but compute $\mathbf{D}' = \mathbf{L}^{-1}\mathbf{P}^\top\mathbf{AP}(\mathbf{L}^{-1})^\top$ over $\mathbf{Z}_4$. (Note $\mathbf{P}^\top = \mathbf{P}^{-1}$.) If any diagonal 1×1 block of $\mathbf{D}$ has become 2 in $\mathbf{D}'$, output $\langle 0^n | \, C \, | 0^n \rangle = 0$. Else, $\langle 0^n | \, C \, | 0^n \rangle$ is nonzero and is obtained by a simple $O(n)$-time recursion.

Here step 1 from [RCG18] takes time linear in the number s of quantum gates in C, which for standard-basis inputs can be bounded above by $O(n^2/\log n)$ with $O(n)$ quantum Hadamard gates [AG04]. Step 3 is computable in $O(n^\omega)$ time by [DP18], where ω is the exponent of matrix multiplication and is at most $n^{2.372865}$ [Sto10, Wil12, Gal14]. This is also the best known time for computing $n \times n$ matrix rank over any field and for the particular inverses and products in step 4 as well (see [CKL13]). However, when $\langle 0^n | \, C \, | 0^n \rangle \neq 0$ we show that its absolute value is computable quickly from r alone after step 2.

The connections used in our proof run through the real-time conversion of quantum circuits C to "phase polynomials" q_C over $\mathbf{Z}_K$ for $K = 2^k$, $k \geq 1$ in [RC09, RCG18], which extended results by [DHH$^+$04] for $k = 1$, and the analysis of quadratic forms over $\mathbf{Z}_4$ by Schmidt [Sch09] drawing on [Alb38, Bro72]. In the case of graph-state circuits and *stabilizer circuits* more generally, q_C becomes a classical quadratic form over $\mathbf{Z}_4$, as treated also in [CGW18]. Our approach is related to ones involving Gauss sums [BvDR08a, CCLL10, CGW18, Bk18] but exploits the availability of normal forms. For bipartite $\mathbf{A}$ as above, it further devolves into a quadratic form q'_C over $\mathbb{F}_2$ that is *alternating* (as defined below) plus an ancillary vector v. A linear change in basis—which also sends v to a vector w but leaves the probability computation unaffected—gives over $\mathbf{Z}_4$ the normal form

$$q'_C = y_1y_2 + y_3y_4 + \cdots + y_{2g-1}y_{2g} + \sum_{j=1}^{n} 2y_jw_k. \tag{6.1}$$

Here the rank r must be even and $g = r/2$. This corresponds to block-diagonal matrices $\mathbf{D}$ with g-many 2×2 blocks as produced by [DP18], together with 1×1 blocks coming from w. The 1×1 blocks matter most for $j > r$. The matrix $\mathbf{D}'$ over $\mathbb{Z}_4$ may no longer be block-diagonal but its diagonal reveals w.

Hereafter, let $N_c(q)$ stand for the number of arguments $x \in \{0,1\}^n$ giving $q(x) = c \pmod 4$ for $c = 0,1,2,3$. Along the way to our main theorem, we prove that for any classical quadratic form q over $\mathbf{Z}_4$, the differences $|N_0(q) - N_2(q)|$ and $|N_1(q) - N_3(q)|$ are either zero or a power of 2. This resolves the effects of the "w" part of the normal form (Equation (6.1)) for the alternating case in particular.

6.2 Circuits and Quadratic Forms

Let R be a commutative ring with unity 1. A *quadratic form* in n variables over R is a homogeneous polynomial

$$q(x_1, \cdots, x_n) = \sum_{1 \le i \le j \le n} a_{i,j}x_ix_j$$

of degree two with coefficient $a_{i,j}$ in R. The study of particular quadratic forms dates back many centuries [BFLR00, Hah08]. The Babylonians in the 18th century B.C. had insight into the ways that the quadratic form $x_1^2 + x_2^2 - x_3^2$ over the integer $\mathbb{Z}$ represents 0. However, the reasonably systematic study of quadratic forms begins with Diophantus and his *Arithmetica* [Hea64] in 3rd century A.D. Since then, many famous mathematician figures appeared in the development of theories over quadratic forms, including Pierre de Fermat, Leonhard Euler, Luigi Lagrange and so on. Not until 19th century, the famous treatise *Disquisitiones Arithmeticae* [Gau66] of 1801 by Carl Friedrich Gauss brought that theory to essentially its modern state.

Quadratic forms can be categorized as *classical* and *non-classical*. A classical quadratic form f in variables $\vec{x} = (x_1, \cdots, x_n)$ is one whose coefficients of crossing terms are all even, while non-classical one has odd coefficients over cross product terms. Being a classical quadratic form means that f can be induced by a symmetric $n \times n$ integer matrix $\mathbf{A}$ as

$$f(\vec{x}) = \vec{x}^\top \mathbf{A}\vec{x},$$

and this is the form that this chapter concentrates on. More precisely, we will deal with classical quadratic forms over $\mathbb{Z}_4$ with $\vec{x} \in \{0,1\}^2$.

Classical quadratic forms are indifferent between 0 and 2 as arguments, likewise 1 versus 3, because $2^2 = 0$, $3^2 = 1$, and $2 \cdot 1 = 2 \cdot 3 = 2$ modulo 4, so counting solutions over $\mathbf{Z}_4^r$ and over $\{0,1\}^r$ is equivalent for them. An amazing "Dichotomy" phenomenon studied by Jin-yi Cai et al. [CLX14, CGW17] show that counting solutions to (all) quadratic forms in $\mathbb{Z}_4^n$ is in P. However, counting *binary* solutions to *non-classical* quadratic forms is NP-hard, indeed, $\#\mathsf{P}$-complete, while counting binary solutions to classical quadratic forms over $\mathbb{Z}_4$ is in P, and we improved the time from $O(n^3)$ to $O(n^\omega)$ (more in Section 6.6).

A high-level idea of how we associate stabilizer circuits with classical quadratic forms is discussed in Section 6.4. Adding the controlled-S gate CS to the family of stabilizer circuits makes it a universal quantum set. Those general quantum circuits will have non-classical quadratic forms over $\mathbb{Z}_4$, over which (as mentioned above) in general it is intractable to count binary solutions.

6.3 Quantum Stabilizer Circuits and Graph-State Circuits

Recall that the family of stabilizer circuits can be generated by the following three gate matrices:

$$\mathsf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

The original polynomial-time algorithm by Gottesman and Knill [Got98] involved Gaussian elimination and so ran for all intents and purposes in order-of n^3 time. Aaronson and Gottesman [AG04] improved this to $O(n^2)$ time with a tableau method and also showed that every stabilizer circuit has an equivalent one with $O(n^2/\log n)$ gates. Anders and Briegel [AB06] improved the running time concretely and for circuits of size $s = o(n^2)$ using a graph-state representation, as we will also do. Dehaene and De Moor [DM03] described quantum states produced by stabilizer circuits via linear and quadratic forms over $\mathbb{F}_2$ in ways simplified and extended by van den Nest [vdN09].

We seek even simpler and faster methods that lend themselves to further algorithmic properties, such as quick update when changes are made to C in the sense of "dynamic algorithms." We employ the theory of classical quadratic forms over $\mathbf{Z}_4$ as developed by

Schmidt [Sch09] and more recently by Cai, Guo, and Williams [CGW18]. The quadratic forms are built using the real-time algorithm of [RC09, RCG18] for computing what we call the *additive partition polynomials* q_C for quantum circuits C that meet a mild "balance" condition. Related works involving low-degree polynomials and counting complexity include [BvDR08a, BJS10, Mon17, KPS17].

The polynomial q_C has variables $x_1, \dots, x_n$ corresponding to binary input values, $z_1, \dots, z_n$ for the binary output values, and $y_1, \dots, y_h$ representing nondeterminism from Hadamard (and possibly other) gates. For any $a, b \in \{0,1\}^n$, letting q_{ab} denote q with those values substituted for the x_i and z_j variables, we have for some $R > 0$:

$$\langle b| \; C \; |a\rangle = \frac{1}{R} \sum_{y \in \{0,1\}^h} \omega^{q_{ab}(y)}, \tag{6.2}$$

where ω is a K-th root of unity such that all phases produced by the circuit are powers of ω. Stabilizer circuits give $K = 4$ so that the powers in this exponential sum belong to $\mathbf{Z}_4$. Generally $R = 2^{h/2}$ but its value is reduced if some nondeterministic y_j variables are forced to equal outputs.

The rules for calculating q are straightforward. Initially $q = 0$ and each qubit line i has its current *annotation* u_i defined by $u_i = x_i$. In general, let u_i stand for the current annotation of line i, and let $y_1, \dots, y_{\ell-1}$ be the nondeterministic variables allocated thus far.

- Hadamard gate on line i: Allocate a new variable y_ℓ, do $q \mathrel{+}= 2u_i y_\ell$, and reassign u_i to be y_ℓ.
- Phase gate S on line i: $q \mathrel{+}= u_i$, u_i unchanged.
- CZ gate on lines i and j: $q \mathrel{+}= 2u_i u_j$, no other change.
- At the end of each qubit line i, we can identify z_i with the variable last denoted by u_i.

Since we are concerned only with $0, 1$ as arguments, we can also do $q \mathrel{+}= u_i^2$ in the case of S, thus making all terms homogeneously quadratic. The *conjugate polynomial* q^* does $q^* \mathrel{+}= 3u_i^2$ instead, but does the same as q for H and CZ.

The annotation u_j becomes quadratic in the case of CNOT, but the degree does not rise any higher: In rules where u_i is multiplied, the multiplier contains a factor 2 which cancels the $2u_i u_j$ modulo 4. The last subtlety is what happens when an annotation that is not a single variable is to be equated with a variable z_j. If it has the form $u_i + u_j - 2u_i u_j$ then we add to q the term $2w(u_i + u_j - 2u_i u_j - z_j) = 2wu_i + 2wu_j + 2wz_j \pmod 4$ where w is a fresh variable. For binary values from the standard basis, if z_j does not equal the XOR of u_i and u_j then the added term reduces to $2w$. Because w appears nowhere else, assignments with

$w = 0$ and those with $w = 1$ will globally cancel in (6.2). Thus only cases with $z_j = u_i \oplus u_j$ contribute. This proves

Theorem 6.1 ([RCG18]). *When C is a stabilizer circuit, the polynomial q in (6.2) becomes a quadratic form over $\mathbf{Z}_4$ in which all terms involving two variables have coefficient* 2.

Another important ingredient in our work is the use of graph-state representation which was first introduced by Raussendorf and Briegel [RB01]. Since then, this tool made a major impact on quantum computing [RBB03, HEB04, AB06, HDE$^+$06] and quantum information [SW01, LYGG08]. It has been proved [VdNDDM04, GKR02, Sch01] that every stabilizer state is equivalent to a graph state. We follow the definition of graph states from [AB06]:

Definition 6.2. *Let G be a given graph (V, E) of $|V| = n$ vertices. The corresponding n-qubit graph state $|G\rangle$ is constructed as*

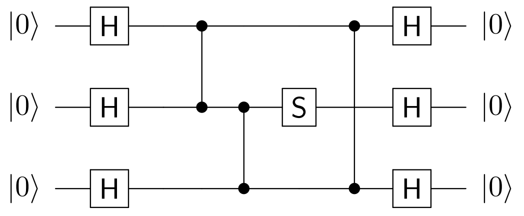
$$|G\rangle = \left(\prod_{(i,j)\in E} \mathsf{CZ}_{i,j}\right)\left(\prod_{i\in V} \mathsf{H}_i\right)|0\rangle^{\otimes n},$$

where $\mathsf{CZ}_{i,j}$ is a controlled-Z gate applied to qubit line i and j, and H_i is a Hadamard gate applied to line i.

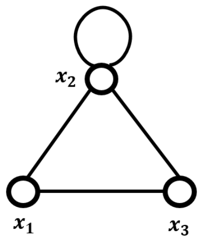
In summary, graph-state circuits consist of:

- An initial n-ary Walsh-Hadamard transform $\mathsf{H}^{\otimes n}$, effected by placing one Hadamard gate at the start of each qubit line.
- For every edge (i, j) in the given graph G, place a CZ gate between lines i and j. Order does not matter because these operations commute.
- If G has a self-loop at node i, place an S gate there.
- A final $\mathsf{H}^{\otimes n}$.

Below Figure 6.1 is an example of a graph-state circuit and its corresponding graph:



(a)



(b)

Figure 6.1: Example of graph-state circuit and its associated graph: x_i corresponds to the i-th qubit line, and each CZ gate is indicated by a vertical line joining two solid dots.

Note that the matrix representation of CZ is

$$\mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

and the notation of CZ gate in Figure 6.1 exhibits its symmetry: it does not matter which of the two qubit lines serve as the control and the other as the target.

Let us bear in mind that since (6.2) computes all amplitudes, the polynomial $q = q_C$ includes all information about the quantum behavior of the circuit C. Thus nothing is lost by manipulating (only) q_C. As an application, we deduce the known fact that graph-state circuits are entirely representative of stabilizer circuits with $O(s+n)$ overhead.

Proposition 6.3. *There is an $O(s+n)$-time procedure that given any n-qubit stabilizer circuit C with h Hadamard gates and $x, z \in \{0,1\}^n$ constructs a graph state circuit C_G on h qubits such that $\langle z| \, C \, |x\rangle = \langle 0^h| \, C_G \, |0^h\rangle$.*

Proof. Build q_C in real time as above and substitute for x and z. This leaves h variables y_k from the Hadamard gates plus any w_j variables that were employed. Now define the graph G to have an edge (i,j) for every term $2y_iy_j$ (or $2y_iw_j$) in q_C, and a self-loops at i for every term ay_i^2, $a = 1,2,3$. Note that the coefficients a of the self-loop terms may arise from the substitutions for particular binary values of x and z. The corresponding graph-state circuit has inputs x', z' of its own, but those are zeroed in forming $\langle 0^h| \, C_G \, |0^h\rangle$. The leftover terms in q_{C_G} are identical to those of q_C after the substitution. $\square$

If $h = \Theta(n)$ then the number of variables is linear in n. Our original aim was to use this correspondence to be competitive with the above-cited $O(n^2)$ algorithms—and ones that improve then the graph is sparse—in the concrete sense of better leading constants and simplified cases. For those algorithms previously not known to have time better than $O(n^3)$ or similar, our practical objective in what follows is not so much reducing the exponent to ω but rather to $O(n^2)$ time given knowledge of the rank r, for contexts where r might be foreknown or well approximated.

6.4 Classical Quadratic Forms Over $\mathbf{Z}_4$

From Section 6.2, a *classical quadratic form* f in variables $\vec{x} = (x_1, \ldots, x_n)$ can be written as

$$f(\vec{x}) = \vec{x}^\top \mathbf{A} \vec{x} \tag{6.3}$$

with $\mathbf{A}$ to be a symmetric $n \times n$ integer matrix. Since every coefficient of a cross term $x_i x_j$ in f is even, and hence over $\mathbb{Z}_4$ all nonzero cross terms have coefficient 2. Such a form over $\mathbf{Z}_4$ treats arguments 0 and 2 the same, likewise 1 and 3, so we may regard it as a function of $\{0,1\}^n$ into $\mathbf{Z}_4$. Then we want to regard (6.3) as composed of matrix-vector operations over $\mathbb{F}_2$ plus some extra calculation to get the answer in $\mathbf{Z}_4$ where $2, 3$ as well as $0, 1$ may be values.

First note that by the symmetry, every off-diagonal entry of $\mathbf{A}$ may without loss of generality be 0 or 1. Next, define a binary vector $\vec{v}$ by $v_j = 1$ if the j-th main-diagonal entry of $\mathbf{A}$ is 2 or 3, else $v_j = 0$. Finally define a binary matrix $\mathbf{B}$ from $\mathbf{A}$ by

$$\mathbf{B} = \mathbf{A} - 2diag(\vec{v}). \tag{6.4}$$

Then we have

$$f(\vec{x}) = \vec{x}^\top \mathbf{B} \vec{x} + 2\vec{x}^\top \cdot \vec{v} \tag{6.5}$$

with calculation in $\mathbf{Z}_4$. The $\vec{x}^\top \mathbf{B} \vec{x}$ calculation is now valid in $\mathbb{F}_2$, however. The quadratic form is *alternating* if the main diagonal of $\mathbf{B}$ is all zero, else it is *non-alternating*. When $\mathbf{B}$ comes from or is regarded as the adjacency matrix of a graph, alternating means the graph is simple and undirected (as will hold in our reductions from rank using a simple bipartite graph) and non-alternating means the graph is undirected but with one or more self-loops.

We note the general development of this decomposition and associated concepts by Schmidt [Sch09] in a way not wedded to the standard basis. Since we fix 4 as the modulus throughout this section, we follow [Sch09] in now using K to denote $\{0,1\}$ as a subset of $\mathbf{Z}_4$, defining an operation $\oplus$ on K by $a \oplus b := (a+b)^2$, and defining V as an n-dimensional vector space "over K" noting that $(K, \oplus, \cdot)$ is the same as the field $\mathbb{F}_2$. Then classical quadratic forms are equivalently defined as follows:

Definition 6.4 (see [Alb38, Sch09]). *A symmetric bilinear form on V is a mapping $B : V \times V \to K$ that satisfies*

I. symmetry: $B(\vec{x},\vec{y}) = B(\vec{y},\vec{x})$;

II. bilinearity: $B(\alpha\vec{x} \oplus \beta\vec{y},\vec{z}) = \alpha B(\vec{x},\vec{z}) \oplus \beta B(\vec{y},\vec{z})$ for $\alpha,\beta \in K$.

B is *alternating* if $B(\vec{x},\vec{x}) = 0$ for all $\vec{x} \in V$, else it is *non-alternating*. Let $\Lambda = \{\lambda_1, \cdots, \lambda_n\}$ be any basis for V over K. Then B is uniquely determined (relative to this basis) by the $n \times n$ matrix $\mathbf{B}$ with entries $b_{ij} = B(\lambda_i, \lambda_j)$. The *rank* of B is the rank of its matrix $\mathbf{B}$.

Definition 6.5 (see [Bro72, Sch09]). *A $\mathbb{Z}_4$-valued classical quadratic form is a mapping $f : V \to \mathbb{Z}_4$ that satisfies:*

I. $f(\alpha\vec{x}) = \alpha^2 f(\vec{x})$ for $\alpha \in K$;

II. $f(\vec{x} \oplus \vec{y}) = f(\vec{x}) + f(\vec{y}) + 2B(\vec{x},\vec{y})$, where $B : V \times V \to K$ is a symmetric bilinear form.

Then f is *alternating* if the associated bilinear form B is alternating, *non-alternating* otherwise, and its *rank* r is the rank of B.

Proposition 6.6 ([Sch09]). *There is a vector $\vec{v} \in K^n$ such that for all $\vec{x} \in K^n$ over the basis Λ,*

$$f(\vec{x}) = \vec{x}^\top \mathbf{B}\vec{x} + 2\vec{x}^\top \cdot \vec{v}.$$

The point of dropping down to $\mathbb{F}_2$ is to leverage the notions of matrix similarity over $\mathbb{F}_2$ and the following theorem about changes of basis in V. Over $\mathbb{F}_2$ the appropriate definition of $\mathbf{B}$ and $\mathbf{B}'$ being *similar* (from [Alb38]) is that there exists an invertible matrix $\mathbf{Q}$ such that $\mathbf{B}' = \mathbf{Q}^\top \mathbf{B}\mathbf{Q}$. This preserves the property that similar matrices have the same rank. The notions of *alternating* and *non-alternating* are the same as given for the binary matrix $\mathbf{B}$ above, depending on whether the main diagonal of $\mathbf{B}$ is all zero or not.

Theorem 6.7 ([Alb38]). *Let $\mathbf{A}$ be a K-valued $n \times n$ symmetric matrix of rank r.*

(a) If $\mathbf{A}$ is alternating, then $\mathbf{A}$ has even rank and is similar to a matrix that has zeros everywhere except on the subdiagonal and the superdiagonal, which are $1010 \cdots 10100 \cdots 0$ with $r/2$ ones.

(b) If $\mathbf{A}$ *is non-alternating, then* $\mathbf{A}$ *is similar to a diagonal matrix, whose main diagonal is of* r*-many ones.*

With the representation of $f(\vec{x}) = \vec{x}^\top \mathbf{B} + 2\vec{x}^\top \cdot \vec{v}$, the paper [Sch09] uses this to define normal forms with regard to $\mathbf{Z}_4$:

Corollary 6.8 ([Sch09])**.** *Given a quadratic form* f *of rank* r *as above over the basis* Λ*, we can find a basis* $M = (\mu_1, \ldots, \mu_n)$ *for* V *over* K*, mapping* $\vec{x} = (x_1, \ldots, x_n)$ *over* Λ *in* V *to* $\vec{y} = (y_1, \ldots, y_n)$ *such that:*

(a) If f *is alternating, then*

$$f(\vec{y}) = 2\sum_{j=1}^{r/2} y_{2j-1}y_{2j} + 2\sum_{i=1}^{n} w_i y_i,$$

for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$*.*

(b) If f *is non-alternating, then there is the equivalent linear form*

$$f(\vec{y}) = \sum_{j=1}^{r} y_j + 2\sum_{i=1}^{n} w_i y_i,$$

for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$*.*

Schmidt actually retains the symbols $\vec{x}$ and $\vec{v}$ in his statement but we have used $\vec{y}$ and $\vec{w}$ to indicate the change of basis. Our analysis in the next section will, however, treat $\vec{y}$ as the standard basis, so the generic symbols $x_1, \ldots, x_n$ will re-appear, and $w_1, \ldots, w_n$ will just be ordinary 0-1 values. This switch will be echoed in the next section in that once we substitute for the input qubit values x_i and output values z_j in the quadratic form q_C from Section 6.3, the actual variables of q_C left over will be named $y_1, \ldots, y_h$ where $h = O(n)$. But to emphasize that the counting lemmas preceding the main results hold apart from the quantum context, we will revert to the standard symbols $x_1, \ldots, x_n$ in their statements and proofs.

Now we reference [DP18] to note some facts about matrix decompositions related to the above normal forms. Note that the inverse of a non-singular lower triangular matrix is lower triangular.

Lemma 6.9. *(a) For every symmetric* $n \times n$ *matrix* $\mathbf{B}$ *over* $\mathbb{F}_2$ *there is a permutation matrix* $\mathbf{P}$ *such that the symmetric matrix* $\mathbf{B}' = \mathbf{P}^\top \mathbf{B} \mathbf{P}$ *has the decomposition* $\mathbf{B}' = \mathbf{L}\mathbf{D}\mathbf{L}^\top$*. Here* $\mathbf{L}$ *is an* $n \times n$ *lower triangular matrix with unit diagonal and* $\mathbf{D}$ *is diagonal if* $\mathbf{B}$ *is non-alternating, else* $\mathbf{D}$ *is block-diagonal as described in Theorem 6.7(a).*

(b) *The matrix $\mathbf{D}$ in (a) is permutation-equivalent to any matrix $\mathbf{D}'$ fulfilling the corresponding case of Theorem 6.7 when applied to $\mathbf{B}'$ or to $\mathbf{B}$.*

(c) *The matrix $\mathbf{D}$ in (a) is unique among LDU decompositions applied to $\mathbf{B}'$.*

(d) *When $\mathbf{D}' = \mathbf{L}^{-1}\mathbf{P}^\top\mathbf{A}\mathbf{P}(\mathbf{L}^{-1})^\top$ is computed over $\mathbf{Z}_4$ rather than $\mathbb{F}_2$, it may no longer be diagonal or block-diagonal, but it represents the same quadratic form with arguments in V and values $\mathbf{Z}_4$ in as in Corollary 6.8 over the new basis. In both the alternating and non-alternating cases, the main diagonal of $\mathbf{D}'$ equals the main diagonal of $\mathbf{D}$ plus $2w$ where w is the vector in Corollary 6.8.*

Proof. (a) This is known and noted in [DP18]. A key point from Gaussian elimination is that if we alternate elementary matrices $\mathbf{L}_i$ that do elimination in the ith column of the lower triangle and swaps $\mathbf{P}_{j,k}$ of rows j and k, then we can rewrite $\mathbf{P}_{j,k}\mathbf{L}_i$ where $j,k > i$ as $\mathbf{L}_i'\mathbf{P}_{j,k}$. The matrix $\mathbf{L}_i'$ is obtained by interchanging the entries in rows j and k of column i and those in positions j and k on the main diagonal. (The latter is unnecessary when all diagonal entries are 1) and is still lower-triangular. Since each $\mathbf{L}_i'$ is still lower triangular and we can repeat the switch for further row swaps, we obtain the lower-triangular matrix formally designated as $\mathbf{L}^{-1}$ as the product of the $\mathbf{L}_i'$ and the matrix designated as $\mathbf{P}^\top$ as the product of all swaps. Since $\mathbf{B}$ is symmetric, corresponding events on the right give $\mathbf{D} = \mathbf{L}^{-1}\mathbf{P}^\top\mathbf{B}\mathbf{P}(\mathbf{L}^{-1})^\top$ of the diagonal or block-diagonal forms stated in all of [Alb38, Sch09, DP18].

Part (b) follows simply because $\mathbf{D}$ and $\mathbf{D}'$ have the same rank and the same block-diagonal structure in the alternating case or diagonal structure in the non-alternating case).

The proof of (c) is the following. Suppose $\mathbf{B}' = \mathbf{LDU} = \mathbf{MEV}$ where $\mathbf{LM}$ are lower triangular and $\mathbf{U}, \mathbf{V}$ are upper triangular, not even caring that $\mathbf{U} = \mathbf{L}^\top$ and $\mathbf{V} = \mathbf{M}^\top$ but just that they are invertible.

First consider the non-alternating case where $\mathbf{D}$ and $\mathbf{E}$ are diagonal but not necessarily of full rank. They must have the same rank r. Then $\mathbf{M}^{-1}$ is also lower triangular, so that $\mathbf{C} = \mathbf{M}^{-1}\mathbf{LD}$ is lower triangular, and $\mathbf{U}^{-1}$ is upper triangular, so that $\mathbf{EVU}^{-1}$ is upper triangular. $\mathbf{C} = \mathbf{MLD} = \mathbf{EVU}^{-1}$, and the only way a lower-triangular matrix can equal an upper-triangular matrix is when both are diagonal. So $\mathbf{C}$ is diagonal, and we need only argue that $\mathbf{C} = \mathbf{D}$ ($= \mathbf{E}$). This follows because they have the same rank and for any i such that $\mathbf{D}[i,i] = 0$, also $\mathbf{C}[i,i] = 0$.

In the alternating case, $\mathbf{M}^{-1}\mathbf{L}$ is lower triangular but its product C with $\mathbf{D}$ can also have a non-zero diagonal above the main diagonal. The product $\mathbf{EVU}^{-1}$ is upper-triangular except for the diagonal below the main. Hence C must be tri-diagonal. Every off-diagonal nonzero element of C equals a diagonal element of $\mathbf{M}^{-1}\mathbf{L}$ multiplied by the corresponding off-diagonal entry of D and also equals a diagonal element of $\mathbf{VU}^{-1}$ multiplying the corresponding entry of $\mathbf{E}$. By invertibility over $\mathbb{F}_2$ the diagonal entries are all 1, so we have proved that D and E agree on all off-diagonal entries. The proof that they agree with each other (but not necessarily with $\mathbf{C}$) in their 1×1 blocks on the diagonal is similar to that for the alternating case.

The point in (d) is that when computed over $\mathbb{Z}_4$, $\mathbf{D'} = \mathbf{L}^{-1}\mathbf{P}^\top \mathbf{A}\mathbf{P}(\mathbf{L}^{-1})^\top$ represents the same quadratic form f originally given by $\mathbf{A}$ in (Equation (6.3)) but over the transformed basis that maps $\vec{x}$ to $\vec{y}$. Thus

$$f(\vec{y}) = \vec{y}^\top \mathbf{D'}\vec{y} = \vec{y}^\top \mathbf{D}\vec{y} + 2\sum_{i=1}^{n} y_i w_i. \tag{6.6}$$

In the non-alternating case, this means any symmetric pairs $d'_{j,k}, d'_{k,j}$ of off-diagonal elements of $\mathbf{D'}$ must sum to 0 modulo 4, and likewise off-diagonal elements in the alternating case apart from the block elements on the super-diagonal and sub-diagonal. The diagonal must satisfy $d'_{j,j} = d_{j,j} + 2w_j \pmod 4$ in either case. □

6.5 Algorithm and Examples

Recall the following algorithm described in Section 6.1 for computing the amplitude $\langle 0^n| \ C \ |0^n\rangle$ for any n-qubit quantum *stabilizer circuit* C:

I. Convert C to a classical quadratic form q_C over $\mathbf{Z}_4$ that retains all quantum properties of C.

II. Take the matrix $\mathbf{A}$ of q_C over $\mathbf{Z}_4$ and associate a canonical $n \times n$ matrix $\mathbf{B}$ over $\mathbb{F}_2$ to it.

III. Compute the decomposition $\mathbf{B} = \mathbf{PLDL}^\top\mathbf{P}^\top$ over $\mathbb{F}_2$ where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is lower-triangular, and $\mathbf{D}$ is block-diagonal with blocks that are either 1×1 or 2×2.

IV. Take $\mathbf{L}^{-1}$ over $\mathbb{F}_2$ but compute $\mathbf{D'} = \mathbf{L}^{-1}\mathbf{P}^\top\mathbf{A}\mathbf{P}(\mathbf{L}^{-1})^\top$ over $\mathbf{Z}_4$. (Note $\mathbf{P}^\top = \mathbf{P}^{-1}$.) If any diagonal 1×1 block of $\mathbf{D}$ has become 2 in $\mathbf{D'}$, output $\langle 0^n| \ C \ |0^n\rangle = 0$. Else, $\langle 0^n| \ C \ |0^n\rangle$ is nonzero and is obtained by a simple $O(n)$-time recursion.

Here step 1 from [RCG18] takes time linear in the number s of quantum gates in C, which for standard-basis inputs can be bounded above by $O(n^2/\log n)$ with $O(n)$ quantum Hadamard gates [AG04]. Step 3 is computable in $O(n^\omega)$ time by [DP18], where ω is the exponent of matrix multiplication and is at most $n^{2.372865}$ [Sto10, Wil12, Gal14]. This is also the best known time for computing $n \times n$ matrix rank over any field and for the particular inverses and products in step 4 as well (see [CKL13]). However, when $\langle 0^n| \ C \ |0^n\rangle \neq 0$ we show that, in Section 6.6, its absolute value is computable quickly from r alone after step 2.

Put more simply, the decomposition in [DP18] is the same as that obtained in [Sch09] following [Alb38, Bro72], so the normal forms for classical quadratic forms over $\mathbf{Z}_4$ in the latter papers inherit the $O(n^\omega)$ time computability from [DP18] working over $\mathbb{F}_2$.

Example 1. Consider the alternating form $q(x_1, x_2, x_3) = 2x_1x_2 + 2x_1x_3 + 2x_2x_3$. It gives

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

which is the adjacency matrix of the triangle graph. Gaussian elimination begins by swapping row 1 and row 2, then no more swaps are needed. So we have:

$$\mathbf{P} = \mathbf{P}_{1,2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{P}^\top, \quad \mathbf{B}' = \mathbf{P}^\top\mathbf{B}\mathbf{P} = \mathbf{B}, \quad \text{and} \quad \mathbf{L}^{-1} = \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

This gives over $\mathbb{F}_2$,

$$\mathbf{D} = \mathbf{L}\mathbf{B}\mathbf{L}^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \cdot \mathbf{L}^\top = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

But over $\mathbf{Z}_4$, we get

$$\mathbf{L}\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 2 & 2 & 2 \end{bmatrix}, \text{ which times } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \mathbf{D}' \equiv \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

The presence of a 2 in the lower-right corner of $\mathbf{D}'$, corresponding to a 1×1 block in the diagonal matrix $\mathbf{D}$, signals a cancellation in the 0-1 assignments $a \in K^n$ giving $q(a) = 0$ versus those giving $q(a) = 2$. That is, $N_0(q) - N_2(q) = 0$. In Section 6.9.1 we will call the simple triangle graph a "net-zero" graph.

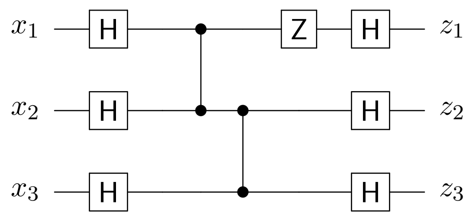
Now, however, let us define $q' = q + 2x_1^2$. This corresponds to adding a self-loop at node 1 to the triangle graph. This goes into the vector $\vec{v}$ and does not change $\mathbf{B}$ or the decomposition. At the end, however, we first get that over $\mathbf{Z}_4$, $\mathbf{A}' = \mathbf{P}^\top\mathbf{A}\mathbf{P}$ is no longer the same as $\mathbf{A}$: it moves the 2 from the upper left corner to the center. Then we get

$$\mathbf{L}\mathbf{A}' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 1 \\ 2 & 0 & 2 \end{bmatrix}, \text{ which times } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix} = \mathbf{D}' \equiv \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

There is a 2 on the main diagonal but it is tucked within a 2×2 block of $\mathbf{D}$. Here in fact we have $N_0(q') = 6$ and $N_2(q') = 2$.

An example of an alternating form q'' with $N_2(q'') > N_0(q'')$ is $q'' = 2x_1^2 + 2x_2^2 + 2x_1x_2$, which corresponds to a single edge with a self-loop at each end. Replacing each self-loop by a triangle yields a 6-node simple undirected graph with $N_0 = 28$ and $N_2 = 36$. We will show that when $N_0 \neq N_2$ in the alternating case, the absolute difference is a simple function of the rank r of $\mathbf{B}$ over $\mathbb{F}_2$.

Example 2. Consider a more detailed example.



From Section 6.3, Hadamard gates will introduce "nondeterministic varibles" which we will use y_i's in the example. However, the introduced nondeterministic variables by the three Hadamard gates at right are immediately equated to the output variables z_1, z_2, z_3. Hence we can skip those and only variables y_1, y_2, y_3 from the three Hadamard gates at left. The corresponding quadratic form is $q(y_1, y_2, y_3) = 2x_1y_1 + 2x_2y_2 + 2x_3y_3 + 2y_1y_2 + 2y_2y_3 + 2y_1^2 + 2y_1z_1 + 2y_2z_2 + 2y_3z_3$ with x_i's and z_i's being constants. Now consider input $|x_1x_2x_3\rangle = |000\rangle$ and output $|z_1z_2z_3\rangle = |000\rangle$. This gives an alternating quadratic form $q(y_1, y_2, y_3) = 2y_1y_2 + 2y_2y_3 + 2y_1^2$ and

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

which is the adjacency matrix of the path graph of length 2 on $n = 3$ vertices. Gaussian elimination does not need any prior swaps, so we have over $\mathbb{F}_4$,

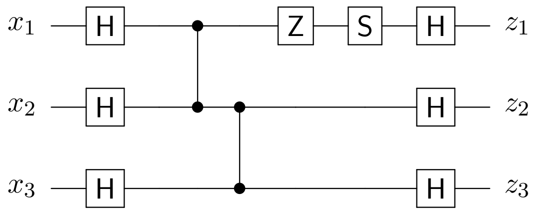
$$\mathbf{D} = \mathbf{LAL}^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{L}^\top = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix}.$$

But

$$\mathbf{D} = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \equiv \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \mathbf{D}'$$

because $q'(y) = y^\top \mathbf{D} y = y^\top \mathbf{D}' y$ over $\mathbb{F}_4$. The 2 at upper left does not zero out the amplitude since it is within a 2×2 blocks. However, the 2 at lower right constitutes a 1×1 block. By our counting lemmas 6.10 and 6.11, it signifies that output $|000\rangle$ is not a possible outcome. Here in fact we have $N_0(q') = 4$ and $N_2(q') = 4$.

Example 3. To make $|000\rangle$ possible in the above example, we can apply phase gate S after Z:



Analogously, after substitution we have the quadratic form $q(y_1, y_2, y_3) = 2y_1y_2 + 2y_2y_3 + 3y_1^2$ which becomes non-alternating. It gives

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

and

$$\mathbf{D} = \mathbf{LAL}^\top = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{L}^\top = \begin{bmatrix} 3 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix}.$$

Again we can ignore the off-diagonal 2's and have

$$\mathbf{D} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix} \equiv= \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Since this is non-alternating and no 2 on the main diagonal, we know that the amplitude is non-zero. Lemma 6.10 and 6.12 gives the amplitude as $\frac{2-2i}{8} = \frac{1-i}{4}$ and so the probability of the output $|000\rangle$ is $\frac{1}{8}$.

6.6 Main Results

Given any n-qubit stabilizer circuit C of size s with h nondeterministic gates, we can obtain its associated quadratic form q_C in $O(s)$ time via the process in Section 6.3. This form has variables $\vec{x} = x_1, \dots, x_n$ for inputs, $\vec{z} = z_1, \dots, z_n$ for outputs, and $y_1, \dots, y_h$ for nondeterministic variables (*wlog.* all coming from h Hadamard gates). It may also have the variables called "w_j" in Section 6.3, but those are introduced only to equate the final annotation term on a qubit line j with the output variable z_j without thereby forcing a value restriction for nondeterministic variable(s) on that line, and so preserve $2^{h/2}$ as the value of the magnitude divisor R in (Equation (6.2)). We can either treat w_j as forced by z_j without changing R, or avoid introducing w_j by reducing R. Since the circuits are allowed to have initial X gates on some lines, treating $\vec{x} = (0, \cdots, 0)$ loses no generality. For any output $\vec{b} = (b_1, \cdots, b_m)$, the quadratic form then becomes

$$\begin{aligned} q(\vec{y}, \vec{b}) &= (\sum \alpha_i y_i + \sum 2y_i y_j) + \sum 2y_i b_j \mod 4 \\ &= \vec{y}^\top \mathbf{A} \vec{y} + \vec{y}^\top 2\mathbf{\Delta} \vec{y} \mod 4 \end{aligned}$$

in the $\vec{y}$ variables only. Here $\mathbf{\Delta}$ is a diagonal matrix with $\mathbf{\Delta}_{i,i} = b_j$. Because we will have $h = \Theta(n)$ for the most part, we still refer to "n" to denote the number of variables in quadratic forms.

Finally, we also fix the outputs b_j all to be 0. We denote by $\vec{N} = (N_0, N_1, N_2, N_3)$ the resulting distribution of values of q_C over the 2^h assignments to $\vec{y}$. Reviewing the discussion surrounding Equation (6.2) in Section 6.3, we can abbreviate the numerator of the amplitude by

$$a_0(\vec{N}) = N_0 - N_2 + i(N_1 - N_3). \tag{6.7}$$

We use the N_c and a_0 notation generally for linear and quadratic forms f without reference to their coming from a quantum circuit. Then a_0 gives the value of the exponential sum $\sum_x i^{f(x)}$.

Now the present the main lemmas that underlie the main theorems. Their proofs are postponed to Section 6.7.

Lemma 6.10. *For any linear function $f(x_1, \cdots, x_n) = \sum_{i=1}^n a_i x_i$ over $\mathbb{Z}_4$, $|N_0 - N_2|$ and $|N_1 - N_3|$ are 0 or a power of 2.*

Lemma 6.11. *For any $\mathbb{Z}_4$-valued alternating quadratic form $f : V \to \mathbb{Z}_4$ of rank r, there is a basis of V over which f can be rewritten as*

$$f(\vec{x}) = 2\sum_{j=1}^{g} x_{2j-1}x_{2j} + 2\sum_{i=1}^{n} w_i x_i$$

for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$, and

$$N_0 - N_2 = 0 \text{ or } (-1)^k 2^{n-g},$$

where $2g = r$ and k is the number of (w_{2j-1}, w_{2j})-pairs in f such that $(w_{2j-1}, w_{2j}) = (1, 1)$ for $j \in \{1, \cdots, g\}$. Also $N_1 = N_3 = 0$.

Lemma 6.12. *For any $\mathbb{Z}_4$-valued non-alternating quadratic form $f : V \to \mathbb{Z}_4$ of rank r, there is a basis of V over which f can be rewritten as*

$$f(\vec{x}) = \sum_{j=1}^{r} x_j + 2\sum_{i=1}^{n} w_i x_i = \sum_{j=1}^{r}(1+2w_j)x_j + 2\sum_{i=r+1}^{n} w_i x_i$$

for some $\vec{w} = (w_1, \cdots, w_n)$. Define c to be the number of w_i's such that $w_i = 0$ with $i \in \{r+1, \cdots, n\}$ and d to be the number of pairs such that $(1+2w_j, 1+2w_{j'}) = (1,3)$ with $j, j' \in \{1, \cdots, r\}$. Also let $m = n - c - 2d$ and rewrite $m = 4a + b$, and define η such that $\eta = 0$ if the rest m-many coefficients are all 1's but $\eta = 1$ if they are all 3's. Then the differences $N_0 - N_2$ and $N_1 - N_3$ take one of the following values:

- *if $b = 0$, then $N_0 - N_2 = (-1)^a 2^{(n+c)/2}$, $N_1 - N_3 = 0$;*
- *if $b = 1$, then $N_0 - N_2 = (-1)^a 2^{(n+c-1)/2}$, $N_1 - N_3 = (-1)^{a+\eta} 2^{(n+c-1)/2}$;*
- *if $b = 2$, then $N_0 - N_2 = 0$, $N_1 - N_3 = (-1)^{a+\eta} 2^{(n+c)/2}$;*
- *if $b = 3$, then $N_0 - N_2 = (-1)^{a+1} 2^{(n+c-1)/2}$, $N_0 - N_2 = (-1)^{a+\eta} 2^{(n+c-1)/2}$.*

The connection between rank and solution counting is expressed by our main theorem about quadratic forms after the normalization process in Lemmas 6.10 to 6.12 is applied:

Theorem 6.13. *Given any normalized classical quadratic form f in n variables, we can compute N_0, N_1, N_2, N_3 and hence $a_0(\vec{N})$ in time $O(n)$. Furthermore, $|a_0(\vec{N})|^2$ is either 0 or 2^{2n-r} where r is the rank of f.*

This means that the bulk of the computing time for the whole process goes into the decomposition in Lemma 6.9, which is used to compute the normal forms asserted in Corollary 6.8. After that, the up-to-n^2 denseness of the original form does not matter and the computation needs only $O(n)$ time.

Proof. We show this separately for the alternating and non-alternating cases. By Corollary 6.8, a normalized alternating quadratic form is of the form

$$f(\vec{x}) = 2\sum_{j=1}^{r/2} x_{2j-1}x_{2j} + 2\sum_{i=1}^{n} w_i x_i \pmod 4,$$

for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$. It is easy to see that $N_1 - N_3$ is always zero since there is no assignment to $\vec{x} = (x_1, \cdots, x_n)$ that would give $f(\vec{x}) = 1$ or 3. Lemma 6.11 gives out

$$N_0 - N_2 = 0 \text{ or } (-1)^k 2^{n-g},$$

which can be done in time $O(n)$. Hence if this is non-zero, then we have

$$a_0(\vec{N}) = (-1)^k 2^{n-g},$$

and

$$|a_0(\vec{N})|^2 = 2^{2n-r}.$$

Similarly, a normalized non-alternating quadratic form is written as

$$f(\vec{x}) = \sum_{j=1}^{r} x_j + 2\sum_{i=1}^{n} w_i x_i = \sum_{j=1}^{r}(1+2w_j)x_j + 2\sum_{i=r+1}^{n} w_i x_i \pmod 4,$$

for some $\vec{w} = (w_1, \cdots, w_n) \in \{0,1\}^n$. Things become trivial if $2w_i = 2$ for some $i \in \{r+1, \cdots, n\}$. This makes $N_0 - N_2 = N_1 - N_3 = 0$.

Now assume $N_0 - N_2$ and $N_1 - N_3$ are not both zero at the same time. Then we can derive from Lemma 6.12 that

$$a_0(\vec{N}) = N_0 - N_2 + i(N_1 - N_3)$$

takes one of the following values:

- if $b = 0$, then $a_0(\vec{N}) = (-1)^a 2^{(n+c)/2}$;
- if $b = 1$, then $a_0(\vec{N}) = (-1)^a 2^{(n+c-1)/2} + i(-1)^{a+\eta} 2^{(n+c-1)/2}$;
- if $b = 2$, then $a_0(\vec{N}) = i(-1)^{a+\eta} 2^{(n+c)/2}$;
- if $b = 3$, then $a_0(\vec{N}) = (-1)^{a+1} 2^{(n+c-1)/2} + i(-1)^{a+\eta} 2^{(n+c-1)/2}$,

where a, b and c are as defined in Lemma 6.12. Note that $c = n - r$. Together we have

$$|a_0(\vec{N})|^2 = 2^{2n-r},$$

and again this can be computed in $O(n)$ time. □

Now we rejoin the process of evaluating the stabilizer circuit C. It will normalize q_C to q' in one of the two forms in Corollary 6.8, which will give a matrix $\mathbf{D}'$ such that $q'(\vec{y}) = \vec{y}^\top \mathbf{D}' \vec{y}$. With such $\mathbf{D}'$, the acceptance probability can be derived directly by Theorem 6.13.

Now we refine the statement of our main results described in Section 6.1 and rephrase our main theorem as follows, but split (b) into two pieces, proving part (b1) here and part (b2) in the next section.

Theorem 6.14 (Main Theorem). *(a) Strong simulation of n-qubit stabilizer circuits C with h nondeterministic single-qubit gates on standard-basis inputs (amplitude as well as the probability) is in time $O(s + n + h^\omega)$ where $2 \leq \omega < 2.3729$.*

(b1) Computing $n \times n$ matrix rank over $\mathbb{F}_2$ reduces in linear time to computing one instance of the strong simulation probability $|\langle 0^n| \; C \; |0^n\rangle|^2$.

(b2) Computing the strong simulation probability $p = |\langle 0^n| \; C \; |0^n\rangle|^2$ reduces in linear time to computing one instance of $n \times n$ matrix rank over $\mathbb{F}_2$ on the promise that $p > 0$.

Proof of (a) and (b1). (a) Let C be given, take $\mathbf{A}$ to be the matrix over $\mathbf{Z}_4$ of its classical quadratic form q_C, and take $\mathbf{B}$ be the associated symmetric matrix over $\mathbb{F}_2$. By Lemma 6.9 and the algorithm of [DP18] there is a decomposition $\mathbf{B} = \mathbf{PLDL}^\top\mathbf{P}^\top$ over $\mathbb{F}_2$ that is computable in $O(n^\omega)$ time such that $\mathbf{D}$ is diagonal (in the non-alternating case) or 2×2 block-diagonal (in the alternating case) and equals the matrix $\mathbf{D}$ in Theorem 6.7. This also computes the rank r of $\mathbf{B}$. Then compute $\mathbf{D'} = \mathbf{L}^{-1}\mathbf{P}^\top\mathbf{AP}(\mathbf{L}^{-1})^\top$ over $\mathbf{Z}_4$ which again takes $O(n^\omega)$ time. By Lemma 6.9(d), $\mathbf{D'}$ and $\mathbf{D}$ yield the vector $\vec{w}$ in the normal form of Corollary 6.8 for q_C. Then Theorem 6.13 yields not only the probability $p = |\langle 0^n| \; C \; |0^n\rangle|^2$ but also the entire distribution of phases as powers of i, and hence yield the amplitude $\langle 0^n| \; C \; |0^n\rangle$.

(b1) To compute the rank r of an $n \times n$ matrix over $\mathbb{F}_2$, make an equivalent symmetric matrix $\mathbf{A}$ by the block-transpose trick in the introduction. Not only is $\mathbf{A}$ alternating but it is the adjacency matrix of a bipartite graph $G = (V, V', E)$. To see that the corresponding graph state circuit C gives $p = |\langle 0^n| \; C \; |0^n\rangle|^2 > 0$, consider any assignment a to the variables in V. This reduces q_C to a linear form $2\ell(x')$ of the variables x' corresponding to nodes of the other partition. If $\ell(x')$ vanishes modulo 2, then all extensions of a to a' on x' contribute 0 modulo 4. Otherwise, $2\ell(x')$ has a nonzero term $2x'_i$ for some i. Assignments a' to x' pair off with canceling contributions 0 and 2 according to the value a'_i of x'_i. Thus there are never more values of 2 than 0. Finally, the all-zero assignment to x makes $\ell(x')$ vanish, so the difference between the numbers of 0 values and 2 values is positive. Thus the normal form for q_C with input and output 0^n cannot have global cancellation, so r is a simple function of p. □

To get the converse simulation in (b2) we must consider the non-alternating case, which arises when the stabilizer circuit C has an odd number of S or S^* gates on some qubit line(s), and allow for the possibility $\langle 0^n| \; C \; |0^n\rangle = 0$. The algorithm for amplitude in the non-alternating case needs knowledge of individual entries in the normal form over $\mathbf{Z}_4$ besides the rank r of q_C.

6.7 Proofs for Important Lemmas

Proof for Lemma 6.10. The given $f(x_1, \cdots, x_n)$ will fall into one of the following cases:

(a) If some $a_j = 0$, it is safe to drop this j-th variable x_j since $\sum_{i=1}^n a_i \cdot x_i \mod 4 = \sum_{i=1, i\neq j}^n a_i \cdot x_i \mod 4$. Define N'_0, N'_1, N'_2, N'_3 with respect to $\vec{x}' = (x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n)$. We can see that $N_i = 2N'_i$ for $i = 0, 1, 2, 3$;

(b) If some $a_j = 2$, then for any $\vec{x}_0 = (x_1, \cdots, x_{j-1}, 0, x_{j+1}, \cdots, x_n)$, it can be paired with $\vec{x}_1$ such that $f(\vec{x}_1) = f(x_1, \cdots, x_{j-1}, 1, x_{j+1}, \cdots, x_n) = f(\vec{x}_0) + 2 \mod 4$. That is, if $f(\vec{x}_0) = 0$, then $f(\vec{x}_1) = 2$, and vice versa. Same analysis goes to N_1 and N_3. Hence, the two differences are zero in this case;

(c) If some $a_j = 1$ and some $a_k = 3$ (without loss of generality, assume $j \leq k$), then for any $\vec{x}_{10} = (x_1, \cdots, x_{j-1}, 1, x_{j+1}, \cdots, x_{k-1}, 0, x_{k+1}, \cdots, x_n)$ and $f(\vec{x}_{10}) = \sum_{i=1, i\neq j, i\neq k}^n a_i \cdot x_i + 1 \mod 4$, we have $f(\vec{x}_{01}) = \sum_{i=1, i\neq j, i\neq k}^n a_i \cdot x_i + 3 \mod 4$, which will cancel in the differences.

While for $\vec{x}_{00} = (x_1, \cdots, x_{j-1}, 0, x_{j+1}, \cdots, x_{k-1}, 0, x_{k+1}, \cdots, x_n)$ and $f(\vec{x}_{00}) = \sum_{i=1, i\neq j, i\neq k}^n a_i \cdot x_i \mod 4$, $f(\vec{x}_{11}) = \sum_{i=1, i\neq j, i\neq k}^n a_i \cdot x_i + 4 \mod 4 = f(\vec{x}_{00})$. Hence, by dropping both j-th and k-th variables (similar to case 1) and defining N'_i with respect to $\vec{x}' = (x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_{k-1}, x_{k+1}, \cdots, x_n)$, $N_i = 2N'_i$ for $i = 0, 1, 2, 3$;

(d) If all x_j's are 1, then for $i = 0, 1, 2, 3$, we have

$$N_i = \sum_{m\geq 0} \binom{n}{4m+i}.$$

Then Lemma 6.15 gives that both differences are powers of 2.

(e) If all x_j's are 3, then

$$N_0 = \sum_{m\geq 0} \binom{n}{4m}, \quad N_2 = \sum_{m\geq 0} \binom{n}{4m+2},$$

$$N_1 = \sum_{m\geq 0} \binom{n}{4m+3}, \quad N_3 = \sum_{m\geq 0} \binom{n}{4m+1}.$$

Then it can be reduced to case 4 and hence both differences are powers of 2.

Note that the above procedures can be applied to a given $f(\vec{x})$ recursively. Overall, the statement holds. $\square$

Lemma 6.15.

$$\sum_{r\geq 0}\binom{n}{4r}-\sum_{r\geq 0}\binom{n}{4r+2}$$

$$\sum_{r\geq 0}\binom{n}{4r+1}-\sum_{r\geq 0}\binom{n}{4r+3}$$

are either 0 or a power-of-2.

Proof. It is known that with ω be the d-th root of unity,

$$\sum_{r\geq 0}\binom{n}{dr+c}=\frac{1}{d}\sum_{j=0}^{d-1}\omega^{-jc}(1+\omega^j)^n$$

where $0\leq c<d$. By simple substitutions with $d=4$ and $a=0,1,2,3$, we get

$$\begin{aligned}g_0&=\sum_{r\geq 0}\binom{n}{4r}-\sum_{r\geq 0}\binom{n}{4r+2}=\frac{1}{2}(1+\omega^n)(1+\omega^3)^n,\\ g_1&=\sum_{r\geq 0}\binom{n}{4r+1}-\sum_{r\geq 0}\binom{n}{4r+3}=\frac{1}{2}\omega^{-1}(\omega^n-1)(1+\omega^3)^n.\end{aligned}$$

Rewrite $n=4a+b$ with $a,b\in\mathbb{Z}$ and $0\leq b<4$. Let $g_0=\sum_{r\geq 0}\binom{n}{4r}-\sum_{r\geq 0}\binom{n}{4r+2}$ and $g_1=\sum_{r\geq 0}\binom{n}{4r+1}-\sum_{r\geq 0}\binom{n}{4r+3}$. It is easy to verify that $(1+\omega^3)^4=-4$ and hence we can rewrite

$$\begin{aligned}g_0&=\frac{1}{2}(-4)^a(1+\omega^b)(1+\omega^3)^b,\\ g_1&=\frac{1}{2}(-4)^a\omega^3(\omega^b-1)(1+\omega^3)^b.\end{aligned}$$

Now we can analysis them case by case.

I. $b=0$: $g_0=(-4)^a$ and $g_1=0$;

II. $b=1$: $g_0=(-4)^a$ and $g_1=(-4)^a$;

III. $b=2$: $g_0=0$ and $g_1=2\cdot(-4)^a$;

IV. $b=3$: $g_0=(-2)(-4)^a$ and $g_1=2\cdot(-4)^a$. □

Proof for Lemma 6.11. Since f is alternating, by Corollary 6.8 and Theorem 6.7, f has even rank r and

$$f(\vec{x})=2\sum_{j=1}^{r/2}x_{2j-1}x_{2j}+2\sum_{i=1}^{n}w_ix_i,$$

for some basis for V over K and for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$. Let $r = 2g$ for some $g \in \mathbb{Z}$ and we can further rewrite it as

$$f(\vec{x}) = 2\sum_{j=1}^{g}(x_{2j-1} + w_{2j})(x_{2j} + w_{2j-1}) + 2\sum_{i=r+1}^{n} w_i x_i - 2\sum_{j=1}^{g} w_{2j-1}w_{2j}$$

Without loss of generality, we first look at the variable pair (x_1, x_2) and its coefficient pair (w_1, w_2). Denote

$$f'(\vec{x}) = 2\sum_{j=2}^{g}(x_{2j-1} + w_{2j})(x_{2j} + w_{2j-1}) + 2\sum_{i=r+1}^{n} w_i x_i - 2\sum_{j=2}^{g} w_{2j-1}w_{2j},$$

and write

$$f(\vec{x}) = f'(\vec{x}) + 2(x_1 + w_2)(x_2 + w_1) + 2w_1w_2.$$

Note that $f'(\vec{x})$ only depends on $(x_3, \cdots, x_n)$, that is, $f'(\vec{x}) = f(x_3, \cdots, x_n)$. There are only four cases to consider for $h(x_1, x_2) = 2(x_1 + w_2)(x_2 + w_1)$:

- $(w_1, w_2) = (0, 0)$: $h(x_1, x_2) = 2x_1x_2$, and $h(0, 0) = 0, h(0, 1) = 0, h(1, 0) = 0, h(1, 1) = 2$;
- $(w_1, w_2) = (1, 0)$: $h(x_1, x_2) = 2x_1(1 + x_2)$, and $h(0, 0) = 0, h(0, 1) = 0, h(1, 0) = 2, h(1, 1) = 0$;
- $(w_1, w_2) = (0, 1)$: $h(x_1, x_2) = 2(1 + x_1)x_2$, and $h(0, 0) = 0, h(0, 1) = 2, h(1, 0) = 0, h(1, 1) = 0$;
- $(w_1, w_2) = (1, 1)$: $h(x_1, x_2) = 2(1 + x_1)(1 + x_2)$, and $h(0, 0) = 2, h(0, 1) = 0, h(1, 0) = 0, h(1, 1) = 0$.

Define $Q_i^{00} = \{\vec{x} \in V | f(\vec{x}) = i \mod 4 \text{ and } x_1 = 0, x_2 = 0\}$ and similarly $Q_i^{01}, Q_i^{10}, Q_i^{11}$. Then we have $Q_i = Q_i^{00} \cup Q_i^{01} \cup Q_i^{10} \cup Q_i^{11}$. Also define $S_i^{00} = \{\vec{x} \in V | f'(\vec{x}) = i \mod 4 \text{ and } x_1 = 0, x_2 = 0\}$ and analogously $S_i^{01}, S_i^{10}, S_i^{11}$.

Note that $f'(\vec{x})$ only depends on $(x_3, \cdots, x_n)$. Let $S_i' = \{(x_3, \cdots, x_n) | f'(x_3, \cdots, x_n) = i\}$, and we have $|S_i'| = |S_i^{00}| = |S_i^{01}| = |S_i^{10}| = |S_i^{11}|$. Now analyze the above four cases separately:

- $(w_1, w_2) = (0, 0)$: we have
$$f(\vec{x}) = f'(\vec{x}) + 2x_1x_2,$$

If $\vec{x}_{00} \in Q_i^{00}$, then $f(\vec{x}_{00}) = f'(\vec{x}_{00}) = i$, same for Q_i^{01}, Q_i^{10}, while if $\vec{x}_{11} \in Q_i^{11}$, then

$f(\vec{x}_{11}) = f'(\vec{x}_{11}) + 2$ and hence $f'(\vec{x}_{11}) = i + 2$. Now for some $c \in \{0, 1\}$,

$$\begin{aligned} N_c - N_{c+2} &= |Q_c^{00}| + |Q_c^{01}| + |Q_c^{10}| + |Q_c^{11}| - (|Q_{c+2}^{00}| + |Q_{c+2}^{01}| + |Q_{c+2}^{10}| + |Q_{c+2}^{11}|) \\ &= |S_c^{00}| + |S_c^{01}| + |S_c^{10}| + |S_{c+2}^{11}| - (|S_{c+2}^{00}| + |S_{c+2}^{01}| + |S_{c+2}^{10}| + |S_c^{11}|) \\ &= 3|S'_c| + |S'_{c+2}| - (3|S'_{c+2}| + |S'_c|) \\ &= 2(|S'_c| - |S'_{c+2}|). \end{aligned}$$

- $(w_1, w_2) = (1, 0)$: by the similar analysis, $|Q_c| - |Q_{c+2}| = 2(|S'_c| - |S'_{c+2}|)$.
- $(w_1, w_2) = (0, 1)$: $|Q_c| - |Q_{c+2}| = 2(|S'_c| - |S'_{c+2}|)$.
- $(w_1, w_2) = (1, 1)$: $|Q_c| - |Q_{c+2}| = -2(|S'_c| - |S'_{c+2}|)$.

Hence, we can reduce the counting of $|Q_c| - |Q_{c+2}|$ over $(x_1, \cdots, x_n)$ to the counting of $|S'_c| - |S'_{c+2}|$ over $(x_3, \cdots, x_n)$, and gradually after g-many such reduction, we can derive

$$|Q_c| - |Q_{c+2}| = (-1)^m 2^g (|Q'_c| - |Q'_{c+2}|),$$

where $Q'_c = \{(x_{r+1}, \cdots, x_n) | 2\sum_{i=r+1}^n w_i x_i = c\}$ and m is the number of (w_{2j-1}, w_{2j}) pairs in $f(\vec{x})$ such that $(w_{2j-1}, w_{2j}) = (1, 1)$, hence $2w_{2j-1}w_{2j} = 2$.

Now it is left to argue that $\big||Q'_c| - |Q'_{c+2}|\big|$ is either zero or a power of 2. Let $q(x) = \sum_{i=r+1}^n 2w_i x_i$. Since $w_i \in \{0, 1\}$, $q(x)$ is linear with coefficient from $\{0, 2\}$. Then we can reduce it to the 1st and 2nd cases in Lemma 6.10. In the 2nd case, it gives that $|Q'_c| - |Q'_{c+2}| = 0$ if $w_i = 1$ for some $i \in \{r + 1, \cdots, n\}$. Now assume non-zero case. Then we have $w_i = 0$ for all $i \in \{r + 1, \cdots, n\}$, which gives

$$|Q'_c| - |Q'_{c+2}| = (-1)^m 2^g 2^{n-r} = (-1)^m 2^{n-g},$$

and hence it completes the proof. □

Proof for Lemma 6.12. Since f is non-alternating, by Corollary 6.8, there exists a basis for V over K, determining the coordinates $(x_1, \cdots, x_n)$, such that

$$f(\vec{x}) = \sum_{j=1}^r x_j + 2\sum_{i=1}^n w_i x_i,$$

for some $\vec{w} = (w_1, \cdots, w_n) \in K^n$. By rearranging, we have

$$f(\vec{x}) = \sum_{j=1}^r (1 + 2w_j) x_j + 2\sum_{i=r+1}^n w_i x_i = \sum_{j=1}^r w'_j x_j + 2\sum_{i=r+1}^n w_i x_i,$$

where $w'_j = 1 + 2w_j$. Note that w'_j can only be 1 or 3. Then we can reduce it to the 2nd, 3rd, 4th and 5th cases in Lemma 6.10.

The 2nd case gives the trivial case where both $N_0 - N_2$ and $N_1 - N_3$ are zero. Now assume non-zero case. Then we have $w'_i, w_i \in \{0, 1, 3\}$.

Define c to be the number of w_i's such that $w_i = 0$ with $i \in \{r+1, \cdots, n\}$ and d to be the number of pairs such that $(1 + 2w_j, 1 + 2w_{j'}) = (1, 3)$ with $j, j' \in \{1, \cdots, r\}$. Also let $m = n - c - 2d$ and rewrite $m = 4a + b$, and define η such that $\eta = 0$ if the rest m-many coefficients are all 1's but $\eta = 1$ if they are all 3's. Then the differences $N_0 - N_2$ and $N_1 - N_3$ are taking one of the following values:

- if $b = 0$, then $N_0 - N_2 = (-1)^a 2^{(n+c)/2}$, $N_1 - N_3 = 0$;
- if $b = 1$, then $N_0 - N_2 = (-1)^a 2^{(n+c-1)/2}$, $N_1 - N_3 = (-1)^{a+\eta} 2^{(n+c-1)/2}$;
- if $b = 2$, then $N_0 - N_2 = 0$, $N_1 - N_3 = (-1)^{a+\eta} 2^{(n+c)/2}$;
- if $b = 3$, then $N_0 - N_2 = (-1)^{a+1} 2^{(n+c-1)/2}$, $N_0 - N_2 = (-1)^{a+\eta} 2^{(n+c-1)/2}$. $\square$

6.8 More Properties from the Simulation

Note that Lemmas 6.10, 6.11, and 6.12 and the proof method of Theorem 6.13 apply to more general input $\vec{a}$ and output $\vec{b}$ as well, so that we have the following supplementary result:

Theorem 6.16. *Given a stabilizer circuit C and its quadratic form $q_C(\vec{y}, \vec{z})$, assume we know $\mathbf{Q}, \mathbf{D}_1$ and $\mathbf{D}_2$ with entries in $\mathbb{F}_2$ such that $\vec{y}^\top \mathbf{Q}^\top \mathbf{A} \mathbf{Q} \vec{y} = \vec{y}^\top (\mathbf{D_1} + 2\mathbf{D}_2)\vec{y}$ where*

- *if q_C is alternating, $\mathbf{D}_2$ is a diagonal matrix with entries in $\{0, 1\}$ and $\mathbf{D}_1 = \mathbf{M}_1 \oplus \cdots \oplus \mathbf{M}_g$ has even rank $r = 2g$ over $\mathbb{F}_2$;*
- *if q_C is non-alternating, $\mathbf{D}_1$ and $\mathbf{D}_2$ are both diagonal matrices with entries in $\{0, 1\}$.*

Then we can compute $|\langle \vec{b}|\ C\ |\vec{0}\rangle|^2$ for any output vector $\vec{b}$ to the circuit in $O(en)$ time where $n = |\vec{y}|$ and e is the number of ones in $\vec{y}$.

Proof. Assume $\mathbf{Q} = (Q_{i,j})$ with $Q_{i,j} \in \mathbb{F}_2$ and take any output vector $\vec{b}$. Then $q_C(\vec{y}, \vec{b}) = \vec{y}^\top \mathbf{A} \vec{y} + \vec{y}^\top \mathbf{\Delta} \vec{y}$ and we have

$$\begin{aligned}\vec{y}^\top \mathbf{Q}^\top \mathbf{A} \mathbf{Q} \vec{y} + \vec{y}^\top \mathbf{Q}^\top \mathbf{\Delta} \mathbf{Q} \vec{y} &= \vec{y}^\top (\mathbf{D_1} + 2\mathbf{D}_2)\vec{y} + \sum_i 2\vec{y}^\top \mathbf{E}_i \vec{y} \\ &= \vec{y}^\top \mathbf{D_1} \vec{y} + \vec{y}^\top 2(\mathbf{D}_2 + \sum_i \mathbf{E}_i)\vec{y}\end{aligned}$$

where E_i is a diagonal matrix $diag(Q_{i,1}, \cdots, Q_{i,n})$ for i such that $\Delta_{i,i} = 1$ and $\mathbf{D}_1$ varies depending on whether it is alternating or non-alternating. Then each $E_i = diag(Q_{i,1}, \cdots, Q_{i,n})$ can be obtained in $O(n)$ time given the matrix $\mathbf{Q}$.

We also know that in both the alternating and non-alternating cases, the output probability $|\langle\vec{b}|\ C\ |\vec{0}\rangle|^2$ is determined by the rank of $\mathbf{D_1}$ if $|\langle\vec{b}|\ C\ |\vec{0}\rangle|^2 \neq 0$. Now we will show that with the knowledge of such $\mathbf{Q}$, we can tell $|\langle\vec{b}|\ C\ |\vec{0}\rangle|^2 = 0$ in $O(en)$ time.

First suppose $q_C(\vec{y},\vec{z})$ is alternating and $n = |\vec{y}|$, then for output $\vec{b}$ we can rewrite

$$\vec{y}^\top \mathbf{D_1}\vec{y} + \vec{y}^\top 2(\mathbf{D}_2 + \sum_i \mathbf{E}_i)\vec{y} = \sum_{j=1}^{g} 2y_{2j-1}y_{2j} + \sum_{i=1}^{n} 2w_i y_i \mod 4,$$

where $w_i \in \{0,1\}$.

Once we finish updating the above equation (which takes $O(en)$ time), we can by Lemma 6.11, get the value $\langle\vec{b}|C|\vec{0}\rangle$ and identify if $|\langle\vec{b}|C|\vec{0}\rangle|^2 = 0$ which happens when $w_i = 0$ for some $i \in \{r+1,\cdots,n\}$. Analogously, this also can be done in $O(en)$ time by Lemma 6.12 for non-alternating cases. □

Now consider a graph $G = (V,E)$ and its adjacency matrix $\mathbf{A}_G$. Assume we are also given matrix $\mathbf{P}$ such that $\mathbf{P}^\top\mathbf{A}_G\mathbf{P} = \mathbf{D}$ where $\mathbf{D}$ is in normalized form. Let $\mathbf{E}_i$ be the matrix with only the (i,i)-th entry being 1 and others 0. Also let $\mathbf{E}_{\vec{v}} = diag(v_1,\cdots,v_n)$ with $\vec{v} = (v_1,\cdots,v_n)$. It is easy to check that $\mathbf{P}^\top 2\mathbf{E}_i\mathbf{P}$ is again a diagonal matrix. More precisely,

$$\mathbf{P}^\top 2\mathbf{E}_i\mathbf{P} = 2diag(P_{i,1}\cdots,P_{i,n}) \mod 4,$$

and we have

$$\mathbf{P}^\top 2\mathbf{E}_{\vec{v}}\mathbf{P} = 2\sum_{i:v_i=1} diag(P_{i,1}\cdots,P_{i,n}) \mod 4.$$

Suppose the vector set $\{\vec{v_i}\}$ with cardinality $\mathsf{rank}(\mathbf{A}_G)$ over $\mathbb{F}_2$ such that

$$\mathbf{P}^\top 2\mathbf{E}_{\vec{v_i}}\mathbf{P} = 2\mathbf{E}_i.$$

With Lemma 6.10, Lemma 6.11 and Lemma 6.12, we can check that the vector space spanned by $\{\vec{v_i}\}$ over $\mathbb{F}_2$ gives all the vector $\vec{b}$ such that $\langle\vec{b}|C_G|0^n\rangle \neq 0$. The proof can be extended directly from the proof for those lemmas. Hence we have the following:

Lemma 6.17. *Given an adjacency matrix* $\mathbf{A}$ *for a stabilizer circuit, the set of all outputs of non-zero amplitudes form a vector space (affine space) of dimension* $\mathsf{rank}(\mathbf{A})$ *over* $\mathbb{F}_2$.

Proof. For convenience, consider an adjacency matrix $\mathbf{A}$ for a bipartite graph. Assume given matrices $\mathbf{P}$ and $\mathbf{D}$ such that $\mathbf{P}^\top\mathbf{A}\mathbf{P} = \mathbf{D}$ over $\mathbb{Z}_4$ where $\mathbf{D}$ is a normalized form (might be with 0/2 on the diagonal). Say the rank of $\mathbf{D}$ is r. Note that this type of stabilizer circuits always has non-zero amplitude on output 0^n. Then $\mathbf{D}$ has no non-zero entries on the diagonal from $(r+1)$-th through n-th positions by Lemma 6.11, since $\mathbf{D}$ corresponds to output 0^n. For other output $\vec{b}$, we have

$$\mathbf{P}^\top(\mathbf{A}+2\mathbf{E}_{\vec{b}})\mathbf{P} = \mathbf{D} + 2\sum_{i:b_i=1} diag(P_{i,1}\cdots,P_{i,n}) = \mathbf{D}_{\vec{b}} \mod 4.$$

Again the amplitude on $\vec{b}$ is non-zero only when there is no non-zero entries on the diagonal of $\mathbf{D}_{\vec{b}}$ from $(r+1)$-th through n-th positions. We can see that for any two non-zero amplitude $\vec{b_1}$ and $\vec{b_2}$, the matrix $\mathbf{D} + 2\sum_{i:b_{1,i}=1} diag(P_{i,1}\cdots,P_{i,n}) + 2\sum_{i:b_{2,i}=1} diag(P_{i,1}\cdots,P_{i,n})$ will never have non-zero entries on the diagonal from $(r+1)$-th through n-th positions. Thus the output $\vec{b_1}+\vec{b_2}$ is of non-zero amplitude as well. Hence the set of outputs with non-zero amplitude forms a subgroup over $\mathbb{F}_2^n$ with 0^n be the identity. □

6.9 Interpretations and Conclusions

We have improved the asymptotic running time for strong simulation of n-qubit stabilizer circuits (with typical size and nondeterminism) from $O(n^3)$ to $O(n^\omega)$. We have also shown a linear time reduction from matrix rank over $\mathbb{F}_2$ to strong simulation. One interpretation of the latter is:

The time gap between weak and strong simulation for stabilizer circuits cannot be closed unless $n\times n$ matrix rank over $\mathbb{F}_2$ is computable in $O(n^2)$ time.

The direction from the quantum simulation to matrix rank comes close to establishing a complete equivalence of them, especially for the simulation probability p. Via analysis of "self-dual" forms we have reduced the probability computation to the alternating case, in which by Lemma 6.11 and Theorem 6.13 we get a simple expression whose absolute value depends only on the rank and whether $p = 0$. That puts focus on the complexity of deciding whether p, or equivalently the amplitude $a = \langle 0^n|\ C\ |0^n\rangle$, is zero, specifically in the alternating case where a is always real.

Besides, we find further connections to graph theory and matroid theory as discussed below:

- In Section 6.9.1, we define a new class of graphs, named *net-zero graph*, which is naturally connected to the problem of whether $\langle 0^n|\ C\ |0^n\rangle$ is zero or not.
- Section 6.9.2 shows that $\langle 0^n|\ C\ |0^n\rangle$ is indeed an instance of *general Tutte invariants.*

We have shown tight connections to the fundamental problems of counting solutions to quadratic forms f over $\mathbb{F}_2$ and $\mathbf{Z}_4$. For $\mathbb{F}_2$ we get that $2f$ is an alternating form over $\mathbf{Z}_4$

with the same solution count over $\{0,1\}^n$, so the near-equivalence to matrix rank applies. In any event we have reduced the $\mathbb{F}_2$ case to matrix multiplication in a way that improves the $O(n^3)$ running time stated in [EK90] to $O(n^\omega)$. For binary solution counting of non-alternating classical quadratic forms over $\mathbf{Z}_4$, we obtain $O(n^\omega)$ runtime via methods that multiply matrices as well as compute rank.

When the non-Clifford gate CS is added to create a universal set, the quadratic forms over $\mathbf{Z}_4$ have terms xy or $3xy$. They are no longer classical and the connection to $\mathbb{F}_2$ exploited by [Sch09] no longer applies. No such connection can apply, nor any extension of the algorithm in [DP18] *a-fortiori*, unless $\mathsf{BQP} = \mathsf{P}$. There is also the sharp dichotomy theorem of [CLX14] that solution counting for these forms over all of $\mathbf{Z}_4^n$ is in polynomial time, but over $\{0,1\}^n$ it is $\#\mathsf{P}$-complete. This extends to affine versus non-affine forms over $\mathbf{Z}_K^n$, $K = 2^k$. Deeper understanding of *why* the dichotomy operates may illuminate exactly which elements of quantum computations create hardness for classical emulation (for this, see [Bac17, Bac18]).

Nevertheless, perhaps these techniques can apply to heuristic or approximative methods on general quantum circuits. The polynomial translation in [RCG18] applies to quantum circuits of all common gate types. There are questions about analyzing circuits that are "mostly Clifford" or those from the Clifford plus T libraries that try to minimize the latter gates, of which we mention[BG16, MFIB18, BBC$^+$19]. For example, are there reasonably-tight bounds for the numbers of the non-Clifford gates required to compute certain functions that can be obtained efficiently by algebraic means, without resort to exhaustive search?

A closer look into Lemma 6.11 and Lemma 6.12 suggests that the probability of a specific output or the distribution over the entire output set can serve as a metric to test whether two given quantum stabilizer circuits are (not) *equivalent*. Let h_0^i, h_1^i be two corresponding h_0, h_1 differences for circuit C_i/quadratic form $f_i(\vec{x})$. Then we can define the following two concepts accordingly.

Definition 6.18. *Given two quantum circuits C_1 and C_2, we call C_1 and C_2 are weakly equivalent, denoted by $C_1 \overset{w}{\approx} C_2$ if*

$$|\langle\vec{z}|\; C_1\; |\vec{a}\rangle|^2 = |\langle\vec{z}|\; C_2\; |\vec{x}\rangle|^2$$

for a fixed input $\vec{a}$ and all possible outputs $\vec{z}$.

Recall from Section 6.1 the definition of $N_j(q(y))$ as the number of arguments y giving $q(y) = j$.

Definition 6.19. *Given two quantum circuits C_1 and C_2, we call C_1 and C_2 are strongly equivalent, denoted by $C_1 \overset{s}{\approx} C_2$ if for all $\vec{a}$ and $\vec{b}$, the amplitudes of C_1 and C_2 are the same,* *that is,*

$$|N_j(Q_1(\vec{a},\vec{y},\vec{b}))| = |N_j(Q_2(\vec{a},\vec{y},\vec{b}))|.$$

Now consider $C_1 \overset{s}{\approx} C_2$ for two given stabilizer circuits. The corresponding $Q_1(\vec{x},\vec{y},\vec{z})$ and $Q_2(\vec{x},\vec{y},\vec{z})$ will be of the forms as stated in Section 6.6. Without loss of generality, assume $\vec{0}$. Note that the resulting $Q_1(\vec{y},\vec{z})$ and $Q_2(\vec{y},\vec{z})$ can be associated with two graphs. Each graph has two sets of nodes $\vec{y}$ and $\vec{z}$. The nodes in $\vec{y}$ can be connected by edges in any way, while there is no edge between nodes among $\vec{z}$ and each node is connected by exactly one node from $\vec{y}$ without overlapping node. Hence, this should be a strict class among graphs and this gives out another interesting question, *does $C_1 \overset{s}{\approx} C_2$ implies that their associated graphs are isomorphic?* If this is true, we will have, $C_1 \overset{s}{\approx} C_2$ if and only if their associated graphs are isomorphic. Note that $C_1 \overset{s}{\approx} C_2$ says $|N_i(Q_1)| = |N_j(Q_2)|$ for all possible outputs, which are exponentially many. We ask:

- If $|N_j(Q_1)| = |N_j(Q_2)|$ for all possible outputs, does $Q_1(\vec{y},\vec{b}) \overset{\mathbb{F}_2}{\sim} Q_2(\vec{y},\vec{b})$ for all possible $\vec{b}$?
- For the case where $C_1 \overset{w}{\approx} C_2$, can we pose a similar question, but in terms of *rank*?

6.9.1 Net-Zero Graphs

The alternating case comes down to graph-state circuits C_G and can be framed in terms apart from quantum computing. Consider black/white two-colorings (not necessarily proper) of the n vertices of G, and count the number of edges whose two nodes are both colored black. Call those B-B edges. Define c_0 to be the count of colorings that make an even number of B-B edges and $c_1 = 2^n - c_0$ to be the count of colorings that make an odd number of B-B edges. The following is called $a(G)$ for "amplitude" and divides by 2^n not $2^{n/2}$ because C_G has $2n$ Hadamard gates.

$$a(G) = \frac{c_0 - c_1}{2^n}.$$

Definition 6.20. *Call an undirected graph G* net-zero *if $a = 0$,* net-positive *if $a > 0$, and* net-negative *if $a < 0$.*

The connection between net-zero graphs and stabilizer circuits is bridged via classical quadratic forms over $\mathbb{Z}_4$. Following the conversion rules in Section 6.3 (and an example in Section 6.5), the connection shows that

$$\langle 0^{\otimes n}|\ C\ |0^{\otimes n}\rangle \text{ is zero if and only if the associated graph } G \text{ is net-zero.}$$

The following proposition collects some basic facts:

Proposition 6.21. *(a) Every odd cycle graph is net-zero.*

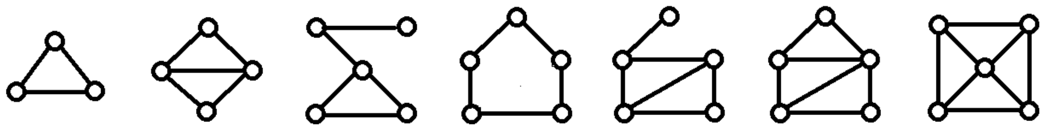
(b) Every bipartite graph is net-positive.

(c) A graph is net-zero if and only if one of its connected components is net-zero.

(d) If G is net-zero, then the graph G' obtained by attaching a new node v only to one existing node u, then attaching a second new node w only to v, is also net-zero.

Proof. Part (a) follows because every coloring has an even number of B-W edges. Hence the number of monochrome edges is odd, and so complementing the coloring flips the parity between B-B and W-W edges. Part (b) was part of the proof of Theorem 6.14(b1). Part (c) is intuitive from how the quantum state is a tensor product over the connected components, so the events of all-0 output on each component are independent. The proof of (d) is that whether u is colored black or white, exactly one of the four colorings of v and w creates one more B-B edge. Thus $|\langle 0^{n+2}|\ C_{G'}\ |0^{n+2}\rangle|^2$ is directly proportional to $|\langle 0^n|\ C_G\ |0^n\rangle|^2$. □

The smallest net-zero graph is the triangle graph. The graph made by attaching a second triangle is net-zero, as is the graph made by attaching a triangle to any of the latter's four outer edges. As observed at the end of section 6.4, the six-node graph consisting of two triangles connected by an edge is net-negative. Here are the connected net-zero graphs of 3, 4, and 5 nodes:



An isolated self-loop is net-zero, while an edge with two self-loops is net-negative. This includes the quadratic forms produced by Lemma 5.2 in our paper [GR19] and we conclude:

Corollary 6.22. *If net-zero graphs of n nodes with self-loops allowed are recognizable in $O(n^2)$ time then computing $|\langle 0^n|\ C\ |0^n\rangle|$ for stabilizer circuits C (of $O(n^2)$ size with $O(n)$ nondeterminism) is $O(N)$-time equivalent to computing $n \times n$ matrix rank, where $N = n^2$.*

6.9.2 Representation via General Tutte Invariant

The above concept of net-zero graphs can further be related to matroid theory. More precisely, they extend to graphs with *circles*, which are isolated loops without a vertex and contribute a multiplicative -1, and more generally to *graphical 2-polymatroids* with *rank function* $f_G(A)$ defined for any set $A \subseteq E$ to be the total number of vertices touched by edges in A.

We follow [Nob06] to give the definition of matroids. A *matroid* is defined by a set U and a function f from finite subsets of U to $\mathbb{N}$ that obeys the following rules:

I. $f(\emptyset) = 0$;

II. for all $a \in U$, $f(\{a\}) \leq 1$;

III. if $A \subseteq B$ then $f(A) \leq f(B)$; and

IV. if $A \subseteq B$ and $c \notin B$ then $f(A \cup \{c\}) - f(A) \geq f(B \cup \{c\}) - f(B)$.

The notion of *rank* in an ordinary vector space obeys these axioms, where we may identify a matrix with its set A of row-vectors. The third axiom says that if B includes all the vectors in A then its rank cannot be lower, and the fourth says that if adding a vector c to B increases its rank—meaning c is independent of B—then it is also independent of A and so the rank of $A \cup \{c\}$ likewise goes up (by 1). Thus matroids abstract the notions of rank and linear independence in vector spaces.

The definition of a *polymatroid* simply wipes out rule 2. OK, a k-polymatroid replaces it by the rule that for all singleton sets $\{a\}$ have $f(\{a\}) \leq k$. An important kind of 2-polymatroid springs from the following idea:

> The "f-rank" of a subset A of the edges in a graph G is the number of vertices collectively touched by edges in A.

In a simple undirected graph, every edge has f-rank 2. In graphs with self-loops, however, the loops have rank 1. We can also allow the universe U to include members of f-rank 0. Those are visualized as loops without a vertex and called *circles*. We could also visualize edges of rank 1 that stick out from a vertex v into empty space, but those are formally the same as loops at v. This is how Noble defines a *graphic(al) 2-polymatroid*.

Then $a(f_G)$ becomes a *generalized Tutte invariant* (see [OW93, Nob06]) with parameters

$$(r, s, t; a, b, c, d; m, n) = (1/2, -1, 0; 1, -1, 1, -1/2; 1, -1/2).$$

This gives

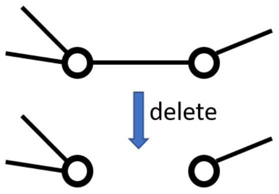
$$a(G) = \left(-\frac{1}{2}\right)^{n/2} S(f_G; -\sqrt{2}i, \sqrt{2}i), \quad \text{where} \quad S(f; x, y) = \sum_{A \subseteq E} x^{f(E)-f(A)} y^{2|A|-f(A)},$$

by the main theorem of [OW93]. This in turn further simplifies to

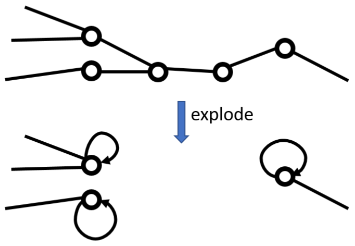
$$a(G) = \sum_{A \subseteq E} \frac{(-2)^{|A|}}{2^{f_G(A)}}.$$

Noble [Nob06] shows that computing $S(f_G; x, y)$ is #P-hard for any constant rational x, y whenever $xy \neq 1$. The complex irrational point $(-\sqrt{2}i, \sqrt{2}i)$ has $xy = 2$ but evades his proof because having $y^2 = -2$ makes a denominator vanish. Other connections between quantum graph states and matroids have been shown by Sarvepalli [Sar14], and there is scope for further development along these lines.

Before showing how to get the desired formula $a(G) = \left(-\frac{1}{2}\right)^{n/2} S(f_G; -\sqrt{2}i, \sqrt{2}i)$, following [Nob06], we need to define two operations on graphs: *edge-deletion* and *edge-explosion*.



(a) Deleting an edge e: $G \backslash e$



(b) Exploding an edge e: G/e

Figure 6.2: Deletion and Explosion.

Edge-deletion $G \backslash e$ simply removes edge e from the graph. While in edge-explosion G/e, two vertices disappear. Not only does the edge $e = (u, v)$ disappear, but any other edge incident to u or v from a vertex $w \neq u, v$ gets "recoiled" into a loop at w.

Following Noble's paper, let $\mathcal{M}$ denotes the class of all graphic 2-polymatroids and let $U_{0,1}, U_{1,1}$ and $U_{2,1}$ are the graphic polymatroids with precisely one edge e, which is respectively a circle, loop or edge between two vertices. An edge e is called a separator in G if and only if the endpoints of e and the set of endpoints of edges in $E \backslash e$ are disjoint, that is, e is an isolated edge.

Then $\phi : \mathcal{M} \to \mathbb{C}$ is said to be a generalized Tutte invariant (for graphic 2-polymatroids) if there exist constants $(r, s, t, a, b, c, d, m, n) \in \mathbb{C}^9$ such that

$$\phi(U_{2,1}) = r, \phi(U_{0,1}) = s, \phi(U_{1,1}) = t,$$

and for any graphic 2-polymatroid (E, f),

$$\phi(f) = \phi(f \backslash (E \backslash e))\phi(f \backslash e) \quad \text{if } e \text{ is a separator of } f;$$

and if e is not a separator,

$$\phi(f) = \begin{cases} a\phi(f\backslash e) + b\phi(f/e), & \text{if } f(E\backslash e) = f(E) \text{ and } f(e) = 1; \\ c\phi(f\backslash e) + d\phi(f/e), & \text{if } f(E\backslash e) = f(E) - 1 \text{ and } f(e) = 2; \\ m\phi(f\backslash e) + n\phi(f/e), & \text{if } f(E\backslash e) = f(E) \text{ and } f(e) = 2. \end{cases}$$

Before plugging into the above equations, we need a 2-term recursion formula based on edge-deletion and edge-explosion. Define $c(G) = c_0 - c_1$ and then $a(G) = c(G)/2^{|V|}$. Consider a connected graph $G = (V, E)$ and an edge $e = (u, v) \in E$. The recursion formula can be worked out as follows:

$$\begin{aligned} c(G) &= \underset{u,v\text{ are both black}}{c(G)} + \underset{\text{at least one of } u,v \text{ is white}}{c(G)} \\ &= \underset{u,v\text{ are both black}}{c(G)} + \underset{\text{at least one of } u,v \text{ is white}}{c(G\backslash e)} \\ &= \underset{\text{at least one of } u,v \text{ is white}}{c(G\backslash e)} - \underset{u,v\text{ are both black}}{c(G\backslash e)} \\ &= c(G\backslash e) - \underset{u,v\text{ are both black}}{2c(G\backslash e)} \\ &= c(G\backslash e) - 2c(G/e), \end{aligned}$$

and in turn

$$a(G) = \frac{c(G\backslash e)}{2^{|V|}} - \frac{2}{4} \cdot \frac{c(G/e)}{2^{|V|-2}} = a(G\backslash e) - \frac{1}{2}a(G/e).$$

If e is a self-loop, we can use a similar argument to derive

$$a(G) = a(G\backslash e) - a(G/e).$$

Now we can start working out those constants for $\phi(G) = a(G)$. Recall that the rank value $f(A)$ in a graph G is the number of distinct endpoints of edges in set A.

$$\begin{aligned} r &= a(U_{2,1}) = \frac{1}{2} \\ s &= a(U_{0,1}) = -1 \\ t &= a(U_{1,1}) = 0. \end{aligned}$$

If e is not a separator, we only have two cases to consider–if e is a self-loop or an edge connecting two vertices:

- if $f(E\backslash e) = f(E)$ and $f(e) = 1$, then this edge e is a self-loop and from above we have $a = 1, b = -1$;
- if $f(E\backslash e) = f(E) - 1$ and $f(e) = 2$, or $f(E\backslash e) = f(E)$ and $f(e) = 2$, then e is an ordinary edge making $c = m = 1$ and $d = n = -\frac{1}{2}$.

Overall, this gives the set of parameters and hence the desired formula for $a(G)$ as stated above.

Chapter 7

Attack on Matrix Rank over $\mathbb{F}_2$

The result of **Chapter** 6 gives some insights towards a possible breakthrough on the problem of computing the rank of an $m \times n$ (dense) matrix over $\mathbb{F}_2$. The best known time for computing rank over any field is $O(n^\omega)$, and no better time is known over $\mathbb{F}_2$ in particular. Although rank reduces to matrix multiplication, they are not known to be equivalent. So it is possible that computing rank might be in $\tilde{O}(n^2)$ time where equivalently $\omega = 2$. The idea of possibly putting rank in $o(n^\omega)$ time came from combining the quadratic form analysis (in **Chapter** 6) with Fourier analysis.

We prove a few special cases. The results of this chapter have not yet been incorporated into a paper for submission. First we review the top-level algorithms and concepts.

7.1 Motivation

Recall the following algorithm for computing the rank r of an $n \times n$ matrix $\mathbf{A}_0$ over the field $\mathbb{F}_2$:

I. Form the symmetric block matrix $\mathbf{A} = \begin{bmatrix} 0 & \mathbf{A}_0 \\ \mathbf{A}_0^\top & 0 \end{bmatrix}$.

II. Form the quantum *graph state* circuit $C_\mathbf{A}$ for the bipartite graph with adjacency matrix $\mathbf{A}$.

III. Calculate $p =$ the quantum probability that $C_\mathbf{A}(0^{2n}) = 0^{2n}$. The bipartite case assures $p > 0$.

IV. Output $r = \log_2(1/\sqrt{p})$.

Note that step 3 dominates the runtime, which is a counting problem over a graph.

Consider the following 2×4 matrix

$$\mathbf{A}_0 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

and the corresponding bipartite graph $G = (V, E)$:

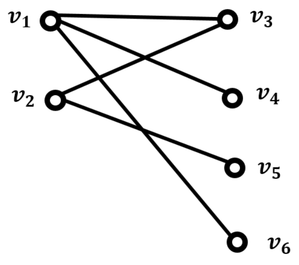


Figure 7.1

Associate the graph with the polynomial

$$q(\vec{x}) = \sum_{(i,j)\in E} x_i x_j = x_1x_4 + x_1x_5 + x_1x_7 + x_2x_4 + x_2x_6 \mod 2,$$

and define s_0 to be the cardinality of the set $\{\vec{x} : q(\vec{x}) = 0 \mod 2\}$ and similarly, s_1 be that for $\{\vec{x} : q(\vec{x}) = 1 \mod 2\}$. Then the wanted value in step 3 is

$$p = \frac{s_0 - s_1}{2^{|V|}} = \frac{\sum_{\vec{x}\in\mathbb{F}_2^n} f(\vec{x})}{2^n},$$

where $|V| = n$ and

$$f(\vec{x}) = (-1)^{q(\vec{x})}.$$

Hence, the problem of rank computing is reduced to the solution counting on $f(\vec{x})$.

To get more intuitions, we can consider the corresponding quantum circuit for this bipartite graph. In general, every graph can be converted into a quantum (stabilizer) circuit. Recall that given an input, the outputs of a quantum circuit are associated with amplitudes such that the probability of "*seeing*" an output is the square norm of its amplitude. The above graph is transformed into the following circuit in Figure 7.2, where the H's are Hadamard gates and the dots on different horizontal lines being connected by a vertical string are control-Z gates (which are the edges in the graph). This circuit has all-zero input, and we are concerned with what are the output vectors with non-zero amplitudes. A nice fact (discussed in Section 6.8) is that *the set of output vectors with non-zero amplitude forms an (affine) subspace.*

We can thus relate the rank to the dimension of this (affine) subspace S. To be sure, the rank is already definable as the dimension of the space R spanned by the rows of the

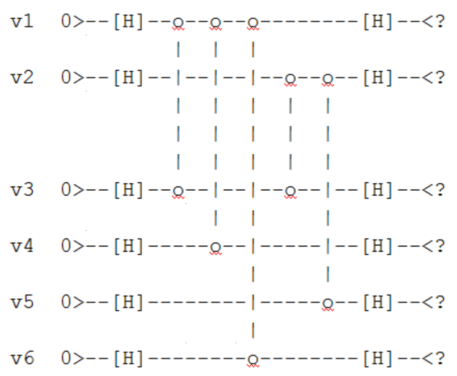


Figure 7.2

original matrix **A**. The point of using S is that we might be able to attack the problem of its dimension better using Fourier analysis iteratively/recursively.

7.2 Insights from Fourier Analysis

Now think of $f(\vec{x})$ as a function $f : \mathbb{F}_2^n \to \mathbb{C}$ with $\mathbb{F}_2^n$ being a finite group of size 2^n and define the character of $\mathbb{F}_2^n$ to be $\psi_a(\vec{x}) = (-1)^{\vec{a}\odot\vec{x}}$. By Fourier analysis, we have

$$p = \mathop{\mathbb{E}}_{\vec{x}\in\mathbb{F}_2^n}[f(\vec{x})] = \mathop{\mathbb{E}}_{\vec{x}\in\mathbb{F}_2^n}[f(\vec{x})\cdot\psi_0(\vec{x})] = \hat{f}(0),$$

where $\hat{f}$ is a Fourier coefficient of f, and a nice property–Plancherel Identity–can come in to play. More precisely, let

$$f(\vec{x}) = (-1)^{q(\vec{x})} = (-1)^{x_1x_4+x_1x_5+x_1x_7}(-1)^{x_2x_4+x_2x_6} = f_1(\vec{x})\cdot f_2(\vec{x})$$

and define inner product between two functions $h, g : \mathbb{F}_2^n \to \mathbb{C}$ to be $\langle h, g\rangle = \mathbb{E}_{x\in\mathbb{F}_2^n}[h(\vec{x})\overline{g(\vec{x})}]$. Then

$$\hat{f}(0) = \mathop{\mathbb{E}}_{\vec{x}\in\mathbb{F}_2^n}[f(\vec{x})] = \mathop{\mathbb{E}}_{\vec{x}\in\mathbb{F}_2^n}[f_1(\vec{x})f_2(\vec{x})] = \langle f_1, f_2\rangle$$

and Plancherel Identity gives

$$\langle f_1, f_2\rangle = \sum_{a\in\mathbb{F}_2^n}\hat{f}_1(\vec{a})\hat{f}_2(\vec{a}).$$

That says that the counting problem over f can be broken into two smaller counting problems (over two smaller subgraphs).

Can this help design an iterative/recursive algorithm for the counting problem with $o(n^\omega)$ time?

If so, we get an algorithm computing rank of a matrix over $\mathbb{F}_2$ with less time than matrix multiplication.

Define $H_1 = \{\vec{a} : \hat{f}_1(\vec{a}) \neq 0\}$ and $H_2 = \{\vec{a} : \hat{f}_2(\vec{a}) \neq 0\}$. The sum becomes $\sum_{a \in H_1 \cap H_2} \hat{f}_1(\vec{a})\hat{f}_2(\vec{a})$. Indeed, both H_1 and H_2 are subspaces (subgroups) of $\mathbb{F}_2^n$ (discussed in Section 6.8), meaning H_1 and H_2 can be enumerated with two sets of bases. Moreover, $\hat{f}_1(\vec{a}) = \pm\frac{1}{2^{h_1}}$ for all $\vec{a} \in H_1$ and $2h_1$ equals the rank of the adjacency matrix of the corresponding subgraph (which is illustrated in Figure 7.3); analogous for $\hat{f}_2$ and H_2. This also means that $\dim(H_1) = 2h_1$ and $\dim(H_2) = 2h_2$.

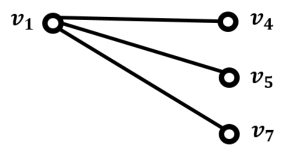


Figure 7.3

For our example, $\dim(H_1) = \dim(H_2) = 2$. Let $H_1 = \mathsf{span}\{\vec{v_1}, \vec{v_2}\}$ and $H_2 = \mathsf{span}\{\vec{w_1}, \vec{w_2}\}$.

$$\begin{aligned} p &= \hat{f}_1(0)\hat{f}_2(0) + \hat{f}_2(\vec{v_1})\hat{f}_2(\vec{v_1}) + \hat{f}_1(\vec{v_2})\hat{f}_2(\vec{v_2}) + \hat{f}_1(\vec{v_1} + \vec{v_2})\hat{f}_2(\vec{v_1} + \vec{v_2}) \\ &= \hat{f}_1(0)\hat{f}_2(0) + \hat{f}_2(\vec{w_1})\hat{f}_2(\vec{w_1}) + \hat{f}_1(\vec{w_2})\hat{f}_2(\vec{w_2}) + \hat{f}_1(\vec{w_1} + \vec{w_2})\hat{f}_2(\vec{w_1} + \vec{w_2}). \end{aligned}$$

There will be three cases to consider (more details in Section 7.4): $\dim(H_1 \cap H_2) = 0, 1$ or 2. Hence now, the question is further reduced to:

Given two sets of bases for two subspaces H_1 and H_2, compute $H_1 \cap H_2$.

From the discussion in Section 7.4, if $H_1 \cap H_2$ is known, $\hat{f}(0)$ could be computed efficiently. So is the basis for the subspace where $\hat{f} \neq 0$.

7.3 Fourier Analysis on Finite Groups

Let G be a finite abelian group. a character of G is simply a homomorphism ψ from G to the multiplicative group of the complex numbers $\mathbb{C}^* : \psi(a + b) = \psi(a)\psi(b)$, and $\psi(-a) = \frac{1}{\psi(a)}$. Since G is finite, we have that every element in the image of ψ is a root of unity, and thus $\frac{1}{\psi(a)} = \overline{\psi(a)}$.

Characters form a group under multiplication. Define the dual group of G to be the group $\widehat{G}$ of all characters of G. Let ψ_0 be the trivial character, which maps all of G to 1; this is the identity element of $\widehat{G}$.

Examples. Let $G = \mathbb{Z}_p$ (p need not be prime) and $\omega_p = e^{\frac{2\pi i}{p}}$. For $a \in \mathbb{Z}_p$, define $\psi_a : G \to \mathbb{C}$ by:

$$\psi_a(x) = \omega_p^{ax}.$$

Here we can see that ω_p is the primitive p-th root of unity. Then $\widehat{G} = \{\psi_a | a \in \mathbb{Z}_p\}$. In our case, $G = \mathbb{Z}_{2^n}$, and for $a \in \mathbb{Z}_{2^n}$, define $\psi_a : G \to \mathbb{C}$ by:

$$\psi_a(x) = (-1)^{a \odot x} = (-1)^{\sum_{i=1}^{n} a_i x_i}.$$

Then $\widehat{G} = \{\psi_a | a \in \mathbb{Z}_{2^n}\}$.

Inner Product. For two complex-valued functions f, g on G, define the inner product to be

$$\langle f, g \rangle = \frac{1}{|G|} \sum_{a \in G} f(a)\overline{g(a)} = \mathop{\mathbb{E}}_{a \in G}[f(a)\overline{g(a)}].$$

Now we can see that every function $f : G \to \mathbb{C}$ can be written as a linear combination of characters of G.

Lemma 7.1. *Every $f : G \to \mathbb{C}$ has the following expression:*

$$f(x) = \sum_{a \in \hat{G}} \hat{f}(a)\psi_a(x),$$

where

$$\hat{f}(a) = \langle f, \psi_a \rangle = \mathop{\mathbb{E}}_{x \in G}[f(x)\overline{\psi_a(x)}].$$

Lemma 7.2 (Plancherel Identity.)**.** *Let $f, g : G \to \mathbb{C}$, then:*

$$\langle f, g \rangle = |G| \langle \hat{f}, \hat{g} \rangle = \sum_{a \in \hat{G}} \hat{f}(a)\overline{\hat{g}(a)}.$$

Proof.

$$\begin{aligned}
\langle f(x), g(x)\rangle &= \mathop{\mathbb{E}}_{x\in G}[\left(\sum_{a_1\in\hat{G}} \hat{f}(a_1)\psi_{a_1}(x)\right)\left(\sum_{a_2\in\hat{G}} \hat{g}(a_1)\psi_{a_2}(x)\right)] \\
&= \mathop{\mathbb{E}}_{x\in G}[\sum_{a_1,a_2\in\hat{G}} \hat{f}(a_1)\psi_{a_1}(x)\overline{\hat{g}(a_2)\psi_{a_2}(x)}] \\
&= \sum_{a_1,a_2\in\hat{G}} \mathop{\mathbb{E}}_{x\in G}[\hat{f}(a_1)\psi_{a_1}(x)\overline{\hat{g}(a_2)\psi_{a_2}(x)}] \\
&= \sum_{a_1,a_2\in\hat{G}} \hat{f}(a_1)\overline{\hat{g}(a_2)} \mathop{\mathbb{E}}_{x\in G}[\psi_{a_1}(x)\overline{\psi_{a_2}(x)}] \\
&= \sum_{a_1,a_2\in\hat{G}} \hat{f}(a_1)\overline{\hat{g}(a_2)}\mathbf{1}_{a_1=a_2} \\
&= \sum_{a\in\hat{G}} \hat{f}(a)\overline{\hat{g}(a)} \\
&= |\hat{G}| \mathop{\mathbb{E}}_{a\in\hat{G}}[\hat{f}(a)\overline{\hat{g}(a)}] \\
&= |G|\langle\hat{f},\hat{g}\rangle.
\end{aligned}$$

□

Lemma 7.3 (Parseval Identity.). *Let $f : G \to \mathbb{C}$, then*

$$\langle f, f\rangle = \|f\|_2^2 = \sum_{a\in\hat{G}} |\hat{f}(a)|^2.$$

7.4 Base Example

Consider a base case Figure 7.4: an n-node bipartite graph $G = (V, E)$ with two nodes on LHS and arbitrary many nodes on RHS. Then the corresponding quadratic form will be

$$q(x) = 2x_1 \sum_{(1,i)\in E} x_i + 2x_2 \sum_{(2,i)\in E} x_i \mod 4.$$

Moreover in bipartite cases, it is equivalent to

$$q(x) = x_1 \sum_{(1,i)\in E} x_i + x_2 \sum_{(2,i)\in E} x_i \mod 2.$$

Hereafter, we identify $q(x)$ with the one over $\mathbb{F}_2$. Define

$$f(x) = (-1)^q(x).$$

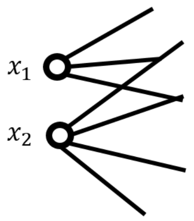


Figure 7.4

Now let

$$g_1(x) = (-1)^{x_1 \sum_{(1,i)\in E} x_i}, \quad g_2(x) = (-1)^{x_2 \sum_{(2,i)\in E} x_i}.$$

We have $f(x) = g_1(x)g_2(x)$.

Note that the amplitude $\langle 0^n|C_G|0^n\rangle = \hat{f}(0)$ now equals $\frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n} g_1(x)g_2(x)$, and more precisely, it is the following:

$$\hat{f}(0) = \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n} g_1(x)g_2(x) = \langle g_1, g_2\rangle = \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\hat{g_2}(a),$$

where the last equality follows Plancherel Identity. More generally, we have

$$\begin{aligned}\hat{f}(b) &= \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n} g_1(x)g_2(x)\psi_b(x)\\ &= \langle g_1, g_2\cdot\psi_b\rangle\\ &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\widehat{g_2\cdot\psi_b}(a)\\ &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\langle g_2\cdot\psi_b, \psi_a\rangle\\ &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\langle g_2, \psi_{b+a}\rangle\\ &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\widehat{g_2}(b+a).\end{aligned}$$

Apparently, we also have $\hat{f}(b) = \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(b+a)\widehat{g_2}(a)$ from commutativity.

Note that the rank of the adjacency matrix for component defined by g_1 is 2, and so is for g_2. Now suppose $V = \mathsf{span}(\{\vec{v_1}, \vec{v_2}\})$ (as defined and discussed in Section 6.8) for g_1 such that $\hat{g_1}(c_1\vec{v_1} + c_2\vec{v_2}) \neq 0$ with $c_i \in \mathbb{F}_2$, and $W = \mathsf{span}(\{\vec{w_1}, \vec{w_2}\})$ for g_2. Without loss of generality, assume $\hat{g_1}(0) = \hat{g_1}(\vec{v_1}) = \hat{g_1}(\vec{v_2}) > 0$ and $\hat{g_1}(v_1 \vec{+} \vec{v_2}) < 0$. Same for g_2. Now we have

$$\begin{aligned}\hat{f}(b) &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\widehat{g_2}(b+a)\\ &= \frac{1}{2}\hat{g}_2(\vec{b}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2}).\end{aligned}$$

There will be three different cases to consider:

I. $V \perp W$

II. $\dim(V \cap W) = 1$

III. $\dim(V \cap W) = 2$, i.e., $V = W$

Case (1): $V \perp W$.

$$\hat{f}(\vec{0}) = \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{0}+\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{0}+\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2}) = \frac{1}{4} = \frac{1}{2^{r/2}},$$

because $\hat{g}_2(\vec{v_1}) = \hat{g}_2(\vec{v_2}) = \hat{g}_2(\vec{v_1}+\vec{v_2}) = 0$ and $\hat{g}_2(\vec{0}) = \frac{1}{2}$. Hence, the total rank is 4. We can also see that the basis set $\{\vec{v_1}, \vec{v_2}, \vec{w_1}, \vec{w_2}\}$ gives the vector space $(V \cup W)$ such that $\hat{f}(\vec{b}) \neq 0, \forall \vec{b} \in V \cup W$.

Case (2): $\dim(V \cap W) = 1$. Let $\vec{t}$ be the basis vector of the intersection space. We do a case-by-case analysis to all the possible situations as listed below:

(a) $\vec{t} = \vec{v_1} = \vec{w_1}$: rank = 2, basis $\{\vec{w_1}, \vec{w_2}+\vec{v_2}\}$

(b) $\vec{t} = \vec{v_2} = \vec{w_2}$: rank = 2, basis $\{\vec{w_2}, \vec{w_1}+\vec{v_1}\}$

(c) $\vec{t} = \vec{v_1} = \vec{w_2}$: rank = 2, basis $\{\vec{w_2}, \vec{w_1}+\vec{v_2}\}$

(d) $\vec{t} = \vec{v_2} = \vec{w_1}$: rank = 2, basis $\{\vec{w_1}, \vec{w_2}+\vec{v_1}\}$

(e) $\vec{t} = \vec{v_1}+\vec{v_2} = \vec{w_1}$: not possible

(f) $\vec{t} = \vec{v_1}+\vec{v_2} = \vec{w_2}$: not possible

(g) $\vec{t} = \vec{v_1} = \vec{w_1}+\vec{w_2}$: not possible

(h) $\vec{t} = \vec{v_2} = \vec{w_1}+\vec{w_2}$: not possible

(i) $\vec{t} = \vec{v_1}+\vec{v_2} = \vec{w_1}+\vec{w_2}$: rank = 2, basis $\{\vec{w_1}+\vec{w_2}, \vec{w_1}+\vec{v_1}\}$ or basis $\{\vec{w_1}+\vec{v_1}, \vec{w_1}+\vec{v_2}\}$

Consider the situation: (2.a) $\vec{t} = \vec{v_1} = \vec{w_1}$.

$$\hat{f}(\vec{b}) = \frac{1}{2}\hat{g}_2(\vec{b}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2}).$$

By plugging in $\vec{b} = \vec{0}$, we get $\hat{f}(\vec{0}) = \frac{1}{2}$ since $\hat{g}_2(\vec{v}_2) = \hat{g}_2(\vec{v}_1+\vec{v}_2) = 0$ by the fact that $\vec{v}_2 \in W$. Hence, rank remains to be 2.

Now some insights for finding the basis for the space S such that $\hat{f}(\vec{b}) \neq 0 \ \forall \vec{b} \in S$ are (1) if $\vec{b} \in W$, both terms $\hat{g}_2(\vec{b}+\vec{v_2})$ and $\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2})$ will be zero; (2) if $\vec{b} \notin W$, we need $\hat{g}_2(\vec{b}+\vec{v_2}) = -\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2})$ and not equal to zero. It is easy to check that $\{\vec{w_1}, \vec{w_2}+\vec{v_2}\}$ is one valid basis for the space S such that $\hat{f}(\vec{w_1}+\vec{w_2}+\vec{v_2})$ has negative amplitude.

The same analysis can be applied to cases (2.b), (2.c) and (2.d) and derive what it shows above. In more general words, the basis consists of (1) one member from the intersection space and (2) one member produced by summing up the basis outside this intersection space.

Cases (2.e), (2.f), (2,g) and (2.h) are impossible because, for instance in (2.e),

$$\hat{f}(\vec{0}) = \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{v_1}+\vec{v_2}) = \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 0 - \frac{1}{2}\cdot\frac{1}{2} = 0,$$

where contradict the fact that $\hat{f}(\vec{0}) > 0$ for bipartite graphs.

As for the last case: $\vec{t} = \vec{v_1}+\vec{v_2} = \vec{w_1}+\vec{w_2}$. We know the rank is again 2 by $\hat{f}(\vec{0}) = \frac{1}{2}$. Following the same insights as above, we can derive a basis set $\{\vec{w_1}+\vec{w_2}, \vec{w_1}+\vec{v_1}\}$. However, $\hat{f}(\vec{w_1}+\vec{w_2})$ gives negative amplitude, and we would want a basis set $\{\vec{s_1}, \vec{s_2}\}$ such that $\hat{f}(\vec{s_1}+\vec{s_2}) < 0$, because this will be consistent with the fact that both $\hat{g}_1(\vec{v_1}+\vec{v_2})$ and $\hat{g}_2(\vec{w_1}+\vec{w_2})$ are negative. Hence, instead, we take the basis set $\{\vec{w_1}+\vec{v_1}, \vec{w_1}+\vec{v_2}\}$ associated with f.

Case (3): $\dim(V \cap W) = 2$. Let $\{\vec{t_1}, \vec{t_2}\}$ be the basis of the intersection space. We do a case-by-case analysis to all the possible situations as listed below:

(a) $\vec{t_1} = \vec{v_1} = \vec{w_1}, \vec{t_2} = \vec{v_2} = \vec{w_2}$: rank = 0

(b) $\vec{t_1} = \vec{v_1} = \vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_1}$: rank = 0

(c) $\vec{t_1} = \vec{v_1}+\vec{v_2} = \vec{w_1}, \vec{t_2} = \vec{v_1} = \vec{w_2}$: not possible

(d) $\vec{t_1} = \vec{v_1}+\vec{v_2} = \vec{w_1}, \vec{t_2} = \vec{v_2} = \vec{w_2}$: not possible

(e) $\vec{t_1} = \vec{v_1}+\vec{v_2} = \vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_1}$: not possible

(f) $\vec{t_1} = \vec{v_1}+\vec{v_2} = \vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_1}$: not possible

(g) $\vec{t_1} = \vec{v_1} = \vec{w_1}+\vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_1}$: not possible (same as (3.f))

(h) $\vec{t_1} = \vec{v_1} = \vec{w_1}+\vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_2}$: not possible (same as (3.d))

(i) $\vec{t_1} = \vec{v_2} = \vec{w_1}+\vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_1}$: not possible (same as (3.e))

(j) $\vec{t_1} = \vec{v_2} = \vec{w_1}+\vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_2}$: not possible (same as (3.c))

(k) $\vec{t_1} = \vec{v_1}+\vec{v_2} = \vec{w_1}+\vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_1}$: rank = 0 (same as (3.a))

(l) $\vec{t_1} = \vec{v_1} + \vec{v_2} = \vec{w_1} + \vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_2}$: rank = 0 (same as (3.b))

(m) $\vec{t_1} = \vec{v_1} + \vec{v_2} = \vec{w_1} + \vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_1}$: rank = 0 (same as (3.b))

(n) $\vec{t_1} = \vec{v_1} + \vec{v_2} = \vec{w_1} + \vec{w_2}, \vec{t_2} = \vec{v_2} = \vec{w_2}$: rank = 0 (same as (3.a))

Consider (3.a): $\vec{t_1} = \vec{v_1} = \vec{w_1}, \vec{t_2} = \vec{v_2} = \vec{w_2}$.

$$\begin{aligned}\hat{f}(\vec{0}) &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{v_1} + \vec{v_2}) \\ &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{w_1}) + \frac{1}{2}\hat{g}_2(\vec{w_2}) - \frac{1}{2}\hat{g}_2(\vec{w_1} + \vec{w_2}) \\ &= \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} - \frac{1}{2}\cdot(-\frac{1}{2}) \\ &= 1.\end{aligned}$$

This means that the rank becomes 0. We can see that this actually corresponds to the scenario that components g_1 and g_2 have identical structure and hence cause cancellation in the combined stabilizer circuit. The same argument works for case (3.b).

Now for (3.c): $\vec{t_1} = \vec{v_1} + \vec{v_2} = \vec{w_1}, \vec{t_2} = \vec{v_1} = \vec{w_2}$.

$$\begin{aligned}\hat{f}(\vec{0}) &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{v_1} + \vec{v_2}) \\ &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{w_2}) + \frac{1}{2}\hat{g}_2(\vec{w_2} + \vec{w_1}) - \frac{1}{2}\hat{g}_2(\vec{w_1}) \\ &= \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot(-\frac{1}{2}) - \frac{1}{2}\cdot\frac{1}{2} \\ &= 0\end{aligned}$$

contradicting the fact that $\hat{f}(\vec{0}) > 0$ for bipartite graphs. Using this argument on cases (3.d) through (3.f) can lead to the same conclusion on them. Also note that (3.g) through (3.j) are the same as (3.c) through (3.f), respectively. For instance, (3.g) is identical to (3.f) because $\vec{v_1} + \vec{v_2} = \vec{t_1} + \vec{t_2} = \vec{w_2}$. Hence, they will again lead to zero rank.

Case (3.k): $\vec{t_1} = \vec{v_1} + \vec{v_2} = \vec{w_1} + \vec{w_2}, \vec{t_2} = \vec{v_1} = \vec{w_1}$.

$$\begin{aligned}\hat{f}(\vec{0}) &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{v_1} + \vec{v_2}) \\ &= \frac{1}{2}\hat{g}_2(\vec{0}) + \frac{1}{2}\hat{g}_2(\vec{w_1}) + \frac{1}{2}\hat{g}_2(\vec{w_2}) - \frac{1}{2}\hat{g}_2(\vec{w_1} + \vec{w_2}) \\ &= \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} - \frac{1}{2}\cdot(-\frac{1}{2}) \\ &= 1.\end{aligned}$$

Note that this case is exactly (3.a) because $\vec{v_2} = \vec{t_1} + \vec{t_2} = \vec{w_2}$. Similarly, we have that (1) (3.l) corresponds to (3.b); (2) (3.m) to (3.b); (3) (3.n) to (3.a).

7.5 Generalization and Thoughts

As seen above, via Fourier analysis, the problem of computing general matrix rank over $\mathbb{F}_2$ is further reduced to:

Given two sets of bases for two subspaces H_1 and H_2, compute $H_1 \cap H_2$.

Now we are ready to discuss two possible ideas for computing matrix rank over $\mathbb{F}_2$: iterative and recursive.

7.5.1 Possible Iterative Approach

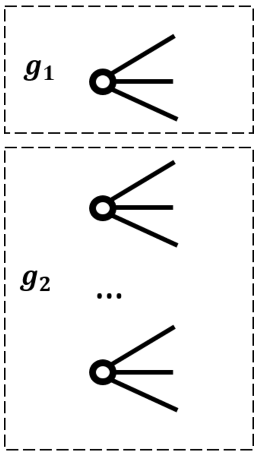


Figure 7.5

Note that if we chop off a bipartite graph as in Figure 7.5, the following formula still works:

$$\begin{aligned}\hat{f}(b) &= \sum_{a\in\mathbb{F}_2^n} \hat{g_1}(a)\widehat{g_2}(b+a)\\ &= \frac{1}{2}\hat{g}_2(\vec{b}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}) + \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_2}) - \frac{1}{2}\hat{g}_2(\vec{b}+\vec{v_1}+\vec{v_2}),\end{aligned}$$

where the vector space $V = \mathsf{span}(\{\vec{v_1}, \vec{v_2}\})$ is associated with g_1 such that $\hat{g}_1(c_1\vec{v_1} + c_2\vec{v_2}) \neq 0$ with $c_i \in \mathbb{F}_2$.

Now the steps of a possible iterative approach for computing matrix rank over $\mathbb{F}_2$ for general matrix will be: (1) write $f(\vec{x}) = \prod_i f_i(\vec{x})$; (2) first "compute and merge" f_1 and f_2, then over that and f_3, and so on.

This approach guarantees that the dimension of the intersection subspace is 0, 1 or 2, which means that $H_1 \cap H_2$ could be easier to compute. However, it will loop $O(n)$ times.

Hence it might only help for special cases such as sparse graphs (matrices). More details of an explicit design of this approach will be one future work.

7.5.2 Possible Recursive Approach

Using the same insight from Fourier analysis, we can also try to design a recursing approach, whose main steps will be: (1) write $f(\vec{x}) = (f_1(\vec{x}) \cdots f_k(\vec{x})) \cdot (f_{k+1}(\vec{x}) \cdots f_{O(n)}(\vec{x}))$ with two equal halves; (2) compute over $(f_1(\vec{x}) \cdots f_k(\vec{x}))$ and $(f_{k+1}(\vec{x}) \cdots f_{O(n)}(\vec{x}))$ recursively.

The main concern is apparently that $H_1 \cap H_2$ could be of large dimension and hence might not be easy to compute. One can ask: (1) what kinds of graphs have a nice division, i.e., easy to compute $H_1 \cap H_2$? (2) can we find a (randomized way) to recurse and efficiently compute $H_1 \cap H_2$?

Chapter 8

Conclusion, Future Research and Speculation

The first main result (in **Chapter** 5) in this thesis is a new logical emulation of general quantum circuits. Our logical approach reduces computing quantum circuits to counting solutions to a Boolean formula. This is potentially a whole paradigm of approaches to emulations. Here are some possible future research problems:

1) *For which types of quantum circuits do the solvers work well?*
   In the experiment using SAT solvers on generated formulas, some quantum circuit instances generated Boolean formulas that are "friendly" (i.e., be efficiently solved) to SAT solvers, while others are not. From those statistics shown Section 5.7, it would be natural to ask what are the types of quantum circuits corresponding to boolean formula instances that are "friendly" to existing SAT solvers.

2) *Can SAT solvers be morphed to understand quantum identities?*
   Quantum identities is a property that a sequence of quantum gates will give an identity operation as a whole. A straightforward example of quantum identities is that two consecutive Hadamard gates cancel out (i.e., producing identity). However, *cachet* and *sharpSAT* both are not able to recognize this phenomenon from the input boolean formulas. If there is no existing SAT solver that can understand quantum identities from input formulas, it would be interesting to investigate how the SAT solver can be designed to identity quantum identities.

3) *Can we identify larger subclasses of #SAT that are polynomial-time solvable?*
   It is known [Got98] that stabilizer circuits can be polynomial-time classically simulated. Now from the perspective of our work, stabilizer circuits correspond to a class of boolean

formulas. Now we can generalize this and ask: *is there a subclass of $\#SAT$ which can be efficiently solved?* and *What properties can be used to characterize the elements in this subclass?* Also note that this serves as a generalization of problem 1) in complexity sense.

4) *Can we classify subclasses whose counting problems correspond exactly to* BQP*?*

   The general intellectual challenge is that people have desired (non-quantum) characterization of BQP, but dichotomy phenomenon [CCLL10, CC12, CGW16] developed by Jin-Yi Cai et al. shows that in myriad cases the complexity jumps from P to $\#P$-complete with nothing in between. Roughly speaking, the dichotomy programme is trying to establish the complete classification of natural counting problems such as those associated to polynomials as being exactly either in P or $\#P$-complete, which conflicts with the goal of characterizing BQP.

   A particularly 'close shave' is that counting the binary solutions of quadratic polynomials when the coefficient of all terms of the form xy is 2 is easy while counting the binary solutions in general is $\#P$-complete. Recently, Cai et al. [CGW17] observed that Clifford gates are indeed a special case of a known tractable class called *affine signatures*, so that dichotomy results give an alternate proof of the Gottesman-Knill Theorem [Got98]. Since counting the satisfying assignments to boolean formulas is somewhat analogous to counting binary solutions, we may ask: *Can we identify subclasses of boolean formulas whose counting problems are in* BQP *but beyond* P*?* or even *is there a dichotomy theorem on boolean formulas between* P *and* BQP*?* If either of the answers is 'yes', it may give intuitions for improving $\#SAT$ solvers?

The second main result (in **Chapter** 6) improved the asymptotic running time for strong simulation of n-qubit stabilizer circuits from $O(n^3)$ to $O(n^\omega)$. This result also shows almost tight and new connections between strong simulation of stabilizer circuits and two bedrock mathematical tasks: *computing matrix rank* and *counting solutions to quadratic polynomials* (both over the field $\mathbb{F}_2$). This says that any improvement on one will imply improvement for the others. Our work also yields an apparently new class of undirected graphs: *net-zero graphs*. The concept further extends to graphs with *circles*, which are isolated loops without a vertex and contribute a multiplicative -1, and more generally to *graphical 2-polymatroids* (more in Section 6.9). These connections also lead to some possible future research problems listed as follows:

1) *What are the tractable subclasses of general Tutte invariant?*

   Noble [Nob06] shows that computing the general Tutte invariant $S(f_G; x, y)$ is $\#\mathsf{P}$-hard for any constant rational x, y whenever $xy \neq 1$. Our case has the quantity

$S(f_G; -\sqrt{2}i, \sqrt{2}i)$ and it can be computed in time $O(n^\omega)$. This complex irrational point $(-\sqrt{2}i, \sqrt{2}i)$ has $xy = 2$ but evades his proof because having $y^2 = -2$ makes a denominator vanish. Hence from the perspective of computational complexity, it would be interesting to know what are the subclasses solvable in polynomial time.

2) *Are there any other further applications of net-zero graphs?*
Net-zero graphs are naturally connected to strong simulation of stabilizer circuits (as shown in Section 6.9). One may ask what other classical or quantum problems this class of graphs can be connected to.

3) *Are the graphs isomorphic if their corresponding stabilizer circuits have identical amplitudes for all outputs of the circuits?*
Our work use the graph-state representation of stabilizer circuits. Given two quantum stabilizer circuits C_1 and C_2. Without loss of generality, assume the inputs to both circuits are all-zero vectors. Each graph has two sets of nodes $\vec{y}$ and $\vec{z}$. The nodes in $\vec{y}$ can be connected by edges in any way, while there is no edge between nodes among $\vec{z}$ and each node is connected by exactly one node from $\vec{y}$ without overlapping node. Hence, this should be a strict class among graphs and this gives out a question, *does that C_1 and C_2 have identical amplitudes for all outputs implies that their associated graphs are isomorphic?* If this is true, we will have an if-and-only-if relation between these two properties.

Besides those discussed above, we also want to speculate about more possible applications of algebraic techniques in quantum computing in the next section. Algebraic techniques have succeeded in many applications in computer science. These techniques seem to have advantages in analyzing polynomial-represented problems.

8.1 Algebraic Geometric Methods and Measuring "Effort"

We want to ask

$$\boxed{\text{What would be a true measure of the effort required to implement a quantum circuit?}}$$

One could imagine a measure built on the notion of coherence and entanglement because they at the same time are sources of quantum advantages and hindrances for physical implementation. However, this would come with the following issues:

- High entanglement does not entail high complexity. For instance, stabilizer circuits can have high entanglement because of Hadamard gates and controlled-NOT gates.

- Entanglement simply is a property of a quantum state not of a circuit.

Hence entanglement by itself is not a useful complexity measure. *What does it mean to define the "entangling capacity" of a quantum circuit?* Nevertheless, there are commonalities between how we would expect certain axioms for entanglement measures $E(C)$ to carry over for circuits and axioms for complexity measures [PV14]. Below is an important one:

- If C_1 and C_2 are disjoint quantum systems, then $E(C_1 \otimes C_2) = E(C_1) + E(C_2)$.

This is the circuit analogue of the axioms called *full additivity* for tensor products of states. This property needs to address another subtle issue: what happens if $C' = C+$ one single-qubit gate. For instance, $C+\mathsf{H}$ cannot always have $E(C+\mathsf{H}) > E(C)$ because $(C+\mathsf{H})+\mathsf{H} = C$.

There is one famous complexity bound on classical circuits which somewhat has this additive property: Strassen's theorem [Str73]. This is the one and only nonlinear lower bound known on classical arithmetic circuits. It is an application of basic algebraic geometry. Hence, we could have an alternative hypothesis: *there is a nonlinearity that can be identified mathematically and that manifests physically as hindrances to maintaining quantum coherence.*

To further motivate our point about the idea of nonlinearity, let us briefly review how Strassen's theorem is proved. For proving his theorem, Strassen used a powerful concept: *geometric degree.*

Geometric degree is also usually called the *degree* of a variety X, which we will denote by $\mathrm{gdeg}(X)$ just for avoiding confusion between this and the degree of a polynomial. In particular, the geometric degree can be defined as the number of points of intersection of X with a general $(n-r)$-plane when X is of dimension r [Gat14, Har13].

The proof of Strassen's lower bound applies Bézout's Theorem. A slightly restricted version of this theoremw can be stated as:

Bézout's Theorem. *Suppose we have r polynomials over an algebraically closed field K. Say $f_i(x_1, \cdots, x_n)$ has degree d_i, for $i = 1, ..., r$. Let $S \subseteq K^n$ denote the set of common zeros of these polynomials. Then S is either infinite or $|S| \leq \prod_{i=1}^{r} d_i$. If the polynomials are algebraically independent and if multiplicities are counted correctly, then the inequality becomes an equality.*

Strassen's lower bound is as follows:

Strassen's Theorem. *Given any a tuple of polynomials $(q_1(x), q_2(x), \cdots, q_n(x))$ of variables $x_1, x_2, \cdots, x_n$, every arithmetic circuit C computing $(q_1, q_2, \cdots, q_n)$ must have size $\Omega(\log(\mathrm{gdeg}(V(q_1, \cdots, q_n))))$.*

Later Walter Baur extended Strassen's result to lower-bound the circuit size of computing a single polynomial $f(x_1, \cdots, x_n)$ with the following key lemma:

Lemma ("Derivative Lemma" [BS83]): *For any polynomial $f(x_1, \cdots, x_n)$ that can be computed by an arithmetic circuit of size L, then there is a circuit of size at most $5L$ that computes the following set of polynomials*

$$\{f, \frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}\}.$$

A simple example is the polynomial $f(x) = x_1^d + x_2^d + \cdots + x_n^d$. By Strassen's lower bound and the above lemma, we have

$$5L(f) \geq L(x_1^{d-1}, \cdots, x_n^{d-1}) \geq \log(\mathrm{gdeg}(V(y_1 - x_1^{d-1}, \cdots, y_n - x_n^{d-1}))) \geq n\log(d-1),$$

where $L(f_1, \cdots)$ is the size of a circuit computing polynomials $(f_1, \cdots)$ and the last inequality comes from the fact that if we restrict $y_i = 1$ for all $i \in [n]$ then there are $(d-1)^n$ complex roots to $\{x_i^{d-1} = 1 : i \in [n]\}$. It is worth pointing out that by taking the first and third terms, the above inequality sequence also gives

$$L(f) = \Omega(\log(\mathrm{gdeg}(\mathsf{MJ}(f)))).$$

Unfortunately, Bézout's inequality also shows that $(d-1)^n$ is the highest possible gdeg one can get not only for this simple f but *any* $f(x_1, \cdots, x_n)$ of degree d. So what is still often regarded as the only known general super-linear complexity lower bound learn us at a fork in the road: either find new ideas needed to boost it or find new areas to apply its ideas. We focus on the latter. It is natural to ask:

What ramifications does this nonlinearity phenomenon have on quantum circuits?

Bacon, van Dam, and Russell [BvDR08b] introduced and analyzed algebraic quantum circuits that are defined over all finite integer rings $\mathbb{Z}_m$ and finite fields $\mathbb{F}_q$. This class of quantum circuits uses algebraic operations of addition and multiplication, as well as the quantum Fourier transform. They showed that the acceptance amplitudes $\langle \vec{b}|\, C\, |\vec{a}\rangle$ of an algebraic quantum circuit can be expressed as an exponential sum over the computational paths between the input $\vec{a}$ and output $\vec{b}$ of the circuit. They also showed that every algebraic quantum circuit has a unique multivariate polynomial f associated with it that captures the "action" of the circuit, where in general f will be a cubic polynomial but for linear circuits (consisting of addition operations) f is only quadratic. With the polynomial f, they proved that the norm of the output amplitude of a linear arithmetic quantum circuits is determined by the dimension of the set of singular points on f. Hence in our problem, it is natural to ask:

What information or quantum property is significantly affected by singular points? Can this (or some similar quantity) be extended to general quantum circuits?

Bibliography

[AB06] S. Anders and H. Briegel. fast simulation of stabilizer circuits using a graph state representation. *Phys. Rev. A*, 73(022334), 2006.

[AC17] Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. In *32nd Computational Complexity Conference (CCC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[ADH97] Leonard M Adleman, Jonathan DeMarrais, and Ming-Deh A Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[AG04] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70(052328), 2004.

[Alb38] A. Albert. Symmetric and alternate matrices in an arbitrary field, I. *Trans. Amer. Math. Soc.*, 43:386–436, 1938.

[Alb83] David Z Albert. On quantum-mechanical automata. *Physics Letters A*, 98(5-6):249–252, 1983.

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM)*, 51(4):595–605, 2004.

[Bac17] M. Backens. A new holant dichotomy inspired by quantum computation. In *Proc. 44th Annual International Conference on Automata, Languages, and Programming*, Leibniz International Proceedings in Informatics (LIPIcs), pages 16:1–16:14, 2017.

[Bac18] M. Backens. A complete dichotomy for complex-valued Holant^c. In *Proc. 45th Annual International Conference on Automata, Languages, and Programming*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, 2018.

[BBC$^+$93] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993.

[BBC$^+$95] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P.W. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52(5):3457–3467, 1995.

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.

[BBC$^+$19] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. In *Proceedings of QIP'19, also https://arxiv.org/abs/1808.00128*, 2019.

[BCDP96] D. Beckman, A.N. Chari, S. Devabhaktuni, and J. Preskill. Efficient networks for quantum factoring. *Phys. Rev. A*, 54:1034–1063, 1996.

[BDEJ95] A. Barenco, D. Deutsch, A. Ekert, and R. Jozsa. Conditional quantum dynamics and logic gates. *Physical Review Letters*, 74(20):4083–4086, 1995.

[Bea03] S. Beauregard. Circuit for shor's algorithm using 2n+3 qubits. *Quantum Information and Computation*, 3:175, 2003.

[BFLR00] E. Bayer-Fluckiger, D. Lewis, and A. Ranicki. *Quadratic Forms and Their Applications: Proceedings of the Conference on Quadratic Forms and Their Applications, July 5-9, 1999, University College Dublin.* Contemporary mathematics - American Mathematical Society. American Mathematical Society, 2000.

[BG16] S. Bravyi and D. Gosset. Improved classical simulation of quantum circuits dominated by Clifford gates. *Physical Review Letters*, 116, 2016.

[BIS$^+$16] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. https://arxiv.org/pdf/1608.00263.pdf, 2016.

[BJS10] M. Bremner, R. Jozsa, and D. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. http://arxiv.org/abs/1005.1407, May 2010.

[Bk18] K. Bu and D. koh. Classical simulation of quantum circuits by half Gauss sums. https://arxiv.org/abs/1812.00224, December 2018.

[BKT18] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 297–310. ACM, 2018.

[Bro72] Edgar H Brown. Generalizations of the Kervaire invariant. *Annals of Mathematics*, 95:368–383, 1972.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[BvDR08a] D. Bacon, W. van Dam, and A. Russell. Analyzing algebraic quantum circuits using exponential sums. http://www.cs.ucsb.edu/ vandam/LeastAction.pdf, November 2008.

[BvDR08b] D. Bacon, W. van Dam, and A. Russell. Analyzing algebraic quantum circuits using exponential sums. http://www.cs.ucsb.edu/ vandam/LeastAction.pdf, November 2008.

[BW92] Charles H. Bennett and Stephen J. Wiesner. Communication via one- and two-particle operators on einstein-podolsky-rosen states. *Phys. Rev. Lett.*, 69:2881–2884, Nov 1992.

[BW03] B. Butscher and H. Weimer. Simulation eines Quantencomputers. http://www.libquantum.de/files/libquantum.pdf, 2003.

[CC12] Jin-Yi Cai and Xi Chen. Complexity of counting csp with complex weights. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, pages 909–920. ACM, 2012.

[CCLL10] J.-Y. Cai, X. Chen, R. Lipton, and P. Lu. On tractable exponential sums. In *Proceedings of the 2010 Frontiers in Algorithms Workshop*, volume 6213 of *Lect. Notes in Comp. Sci.*, pages 48–59. Springer Verlag, 2010.

[CGW16] Jin-Yi Cai, Heng Guo, and Tyson Williams. The complexity of counting edge colorings and a dichotomy for some higher domain holant problems. *Research in the Mathematical Sciences*, 3(1):18, 2016.

[CGW17] Jin-Yi Cai, Heng Guo, and Tyson Williams. Clifford gates in the holant framework. *arXiv preprint arXiv:1705.00942*, 2017.

[CGW18] Jin-Yi Cai, Heng Guo, and Tyson Williams. Clifford gates in the holant frameqwork. *Theor. Comp. Sci.*, 75:163–171, 2018.

[Che97] Joseph Cheriyan. Randomized $\tilde{o}(M(|V|))$ algorithms for problems in matching theory. *SIAM Journal on Computing*, 26(6):1635–1655, 1997.

[CHSH69] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, Oct 1969.

[Cir80] B. S. Cirel'son. Quantum generalizations of bell's inequality. *Letters in Mathematical Physics*, 4:93–100, 1980.

[CKL13] H.Y. Cheung, T.C. Kwok, and L.C. Lau. Fast matrix rank algorithms and applications. *J. Assn. Comp. Mach.*, 60:1–25, 2013.

[CLL11] Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 190–199. IEEE, 2011.

[CLO13] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra.* Springer Science & Business Media, 2013.

[CLX14] J.-Y. Cai, P. Lu, and M. Xia. The complexity of complex weighted Boolean #CSP. *J. Comp. Sys. Sci.*, 80:217–236, 2014.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle, and the universal quantum computer. *Proceedings of the Royal Society*, A400:97–117, 1985.

[Deu89] D. Deutsch. Quantum computational networks. In *Proceedings of the Royal Society of London*, volume 425(1868) of *Series A*, pages 73–90, 1989.

[DHH$^+$04] C. Dawson, H. Haselgrove, A. Hines, D. Mortimer, M. Nielsen, and T. Osborne. Quantum computing and polynomial equations over the finite field Z_2. *Quantum Information and Computation*, 5:102–112, 2004.

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[DM03] J. Dehaene and B.L.R. De Moor. The Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, 68:042318, 2003.

[DP18] J.-G. Dumas and C. Pernet. Symmetric indefinite triangular factorization revealing the rank profile matrix. In *Proc. 43rd International Symposium on Symbolic and Algebraic Computation*, pages 151–158, 2018. Also https://arxiv.org/abs/1802.10453.

[EK90] A. Ehrenfeucht and M. Karpinski. The computational complexity of (XOR, AND)-counting problems. Technical Report TR-90-032, Mathematical Sciences Research Institute, University of California at Berkeley, 1990.

[Fey82] R. Feynmann. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[Fey86] R. Feynmann. Quantum mechanical computers. *Foundation of Physics*, 16:507–531, 1986.

[Gal14] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, 2014.

[Gat14] Andreas Gathmann. Notes on algebraic geometry. http://www.mathematik.uni-kl.de/ gathmann/class/alggeom-2014/alggeom-2014.pdf, 2014.

[Gau66] Carl Friedrich Gauss. Disquisitiones arithmeticae. *New Haven and London, Yale University Press*, 1966.

[GHZ89] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bells theorem. In *Bells theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

[GKR02] Markus Grassl, Andreas Klappenecker, and Martin Rotteler. Graphs, quadratic forms, and quantum codes. In *Proceedings IEEE International Symposium on Information Theory,*, page 45. IEEE, 2002.

[Got98] Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

[GR19] Chaowen Guan and Kenneth W Regan. Stabilizer circuits, quadratic forms, and computing matrix rank. *arXiv preprint arXiv:1904.00101*, 2019.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[GS06] V. Gerdt and V. Severyanov. A software package to construct polynomial sets over Z_2 for determining the output of quantum computations. *Nuclear Instruments and Methods in Physics Research A*, 59:260–264, 2006.

[Hah08] Alexander J Hahn. Quadratic forms over $\mathbb{Z}$ from diophantus to the 290 theorem. *Advances in applied Clifford algebras*, 18(3-4):665–676, 2008.

[Har09] Nicholas JA Harvey. Algebraic algorithms for matching and matroid problems. *SIAM Journal on Computing*, 39(2):679–702, 2009.

[Har13] Joe Harris. *Algebraic geometry: a first course*, volume 133. Springer Science & Business Media, 2013.

[HDE$^+$06] Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, M Nest, and H-J Briegel. Entanglement in graph states and its applications. *arXiv preprint quant-ph/0602096*, 2006.

[Hea64] Thomas Little Heath. *Diophantus of Alexandria.* Dover Publications Inc. New York, 1964.

[HEB04] Marc Hein, Jens Eisert, and Hans J Briegel. Multiparty entanglement in graph states. *Physical Review A*, 69(6):062311, 2004.

[HRS17] T. Häner, M. Roetteler, and K. Svore. Factoring using 2n+2 qubits with Toffoli based modular multiplication. *Quantum Information and Computation*, 17, 2017.

[HS17] T. Häner and D. Steiger. 0.5 petabyte simulation of a 45-qubit quantum circuit. arXiv:1704.01127v1, April 2017.

[HSST16] T. Häner, D. Steiger, M. Smelyanskiy, and M. Troyer. High performance emulation of quantum circuits. In *Proceedings of the International Conference* *for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, Utah, Nov. 2016*. IEEE press, 2016. Article 74 in e-volume.

[IW96] Russell Impagliazzo and Avi Wigderson. P=BPP unless e has sub-exponential circuits: Derandomizing the xor lemma (preliminary version). *In Proceedings of the 29th STOC*, pages 220–229, 1996.

[JvdN14] R. Jozsa and M. van den Nest. Classical simulation complexity of extended Clifford circuits. *Quantum Information and Computation*, 14:633–648, 2014.

[Kar19] Marek Karpinski. Personal communication, 2019.

[KPS17] D.E. Koh, M.D. Penney, and R.W. Spekkens. Computing quopit Clifford circuit amplitudes by the sum-over-paths technique. http-s://arxiv.org/pdf/1702.03316, 2017.

[LCJ13] Chia-Chun Lin, Amlan Chakrabarti, and Niraj K Jha. Ftqls: Fault-tolerant quantum logic synthesis. *IEEE Transactions on very large scale integration (VLSI) systems*, 22(6):1350–1363, 2013.

[Lin14] Chia-Chun Lin. *Reversible and Quantum Circuit Synthesis*. PhD thesis, Princeton University, 2014.

[Lov06] László Lovász. The rank of connection matrices and the dimension of graph algebras. *European Journal of Combinatorics*, 27(6):962–970, 2006.

[Lov07] László Lovász. Connection matrices. *OXFORD LECTURE SERIES IN MATHEMATICS AND ITS APPLICATIONS*, 34:179, 2007.

[LYGG08] Shiang Yong Looi, Li Yu, Vlad Gheorghiu, and Robert B Griffiths. Quantum-error-correcting codes using qudit graph states. *Physical Review A*, 78(4):042303, 2008.

[MFIB18] I. Markov, A. Fatima, S. Isakov, and S. Boixo. Quantum supremacy is both closer and farther than it appears. https://arxiv.org/pdf/1807.10749, 2018.

[Mon85] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.

[Mon17] A. Montanaro. Quantum circuits and low-degree polynomials over F_2. *Journal of Physics A*, 50, 2017.

[MS01] Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory i: An approach to the p vs. np and related problems. *SIAM Journal on Computing*, 31(2):496–526, 2001.

[MS04] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255. IEEE, 2004.

[MS07] Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory: Introduction. *CoRR*, abs/0709.0746, 2007.

[MS08] Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory ii: Towards explicit obstructions for embeddings among class varieties. *SIAM Journal on Computing*, 38(3):1175–1206, 2008.

[MS12] I. Markov and M. Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Information and Computation*, 12:361–394, 2012.

[MS13] I. Markov and M. Saeedi. Faster quantum number factoring via circuit synthesis. *Phys. Rev. A*, 87:012310 1–5, 2013.

[NC00] M.A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[Nob06] S. Noble. Evaluating the rank generating function of a graphic 2-polymatroid. *Combinatorics, Probability and Computing*, 15:449–461, 2006.

[O13] Onorato Timothy OMeara. *Introduction to quadratic forms*, volume 117. Springer, 2013.

[OD98] Kevin M Obenland and Alvin M Despain. A parallel quantum computer simulator. *arXiv preprint quant-ph/9804039*, 1998.

[Öme14] Bernhard Ömer. `http://tph.tuwien.ac.at/~oemer/qcl.html`. 2014.

[OW93] J. Oxley and G. Whittle. A characterization of Tutte invariants of 2-polymatroids. *J. Comb. Thy. Ser. B*, 59:210–244, 1993.

[PG12] Archimedes Pavlidis and Dimitris Gizopoulos. Fast quantum modular exponentiation architecture for shor's factorization algorithm. *arXiv preprint arXiv:1207.0511*, 2012.

[PG14] A. Pavlidis and D. Gizopoulos. Fast quantum modular exponentiation architecture for shor's factoring algorithm. *Quantum Information and Computation*, 14:694–682, 2014.

[PV14] Martin B Plenio and Shashank S Virmani. An introduction to entanglement theory. In *Quantum Information and Coherence*, pages 173–209. Springer, 2014.

[RB01] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22):5188, 2001.

[RBB03] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Phys. Rev. A*, 68:022312, Aug 2003.

[RC09] K. Regan and A. Chakrabarti. Quantum circuits, polynomials, and entanglement measures, 2009. Draft.

[RCG18] Kenneth Regan, Amlan Chakrabarti, and Chaowen Guan. Algebraic and logical emulations of quantum circuits. In *Transactions on Computational Science XXXI*, pages 41–76. Springer, 2018.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.

[San07] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 118–126. Society for Industrial and Applied Mathematics, 2007.

[Sar14] P. Sarvepalli. Quantum codes and symplectic matroids. In *Proceedings of the 2014 IEEE International Symposium on Information Theory; also https://arxiv.org/abs/1104.1171*, 2014.

[SBB$^+$04] Tian Sang, Fahiem Bacchus, Paul Beame, Henry Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *Seventh International Conference on Theory and Applications of Satisfiability Testing, Vancouver*, 2004.

[SBH06] Spec.org, B. Butscher, and Weimer H. 462.libquantum spec cpu2006 benchmark description. https://www.spec.org/cpu2006/Docs/462.libquantum.html, 2006.

[SBK05a] Tian Sang, Paul Beame, and Henry Kautz. Heuristics for fast exact model counting. In *Eighth International Conference on Theory and Applications of Satisfiability Testing, Edinburgh, Scotland*, 2005.

[SBK05b] Tian Sang, Paul Beame, and Henry Kautz. Performing Bayesian inference by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, PA*, 2005.

[Sch01] Dirk Schlingemann. Stabilizer codes can be realized as graph codes. *arXiv preprint quant-ph/0111080*, 2001.

[Sch09] Kai-Uwe Schmidt. Z_4-valued quadratic forms and quaternary sequence families. *IEEE Transactions on Information Theory*, 55:5803–5810, 2009.

[Shi03] Y. Shi. Both Toffoli and Controlled-NOT need little help to do universal quantum computation. *Quantum Information and Computation*, 3:84–92, 2003. arXiv:quant-ph/0205115.

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, pages 124–134, 1994.

[Sim97] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[SRWDM17] Mathias Soeken, Martin Roetteler, Nathan Wiebe, and Giovanni De Micheli. Logic synthesis for quantum computing. *arXiv preprint arXiv:1706.02721*, 2017.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing* Arch. Elektron. Rechnen, 7:281–292, 1971.

[Sto10] A. Stothers. *On the Complexity of Matrix Multiplication.* PhD thesis, University of Edinburgh, 2010.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[Str73] V. Strassen. Berechnung und Programm II. *Acta Informatica*, 2:64–79, 1973.

[SW01] D. Schlingemann and R. F. Werner. Quantum error-correcting codes associated with graphs. *Phys. Rev. A*, 65:012308, Dec 2001.

[TBI97] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[Thu06] M. Thurley. sharpsat – counting models with advanced component caching and implicit bcp. In *Theory and Applications of Satisfiability Testing - SAT 2006: 9th International Conference, Seattle, WA, USA, August 12-15, 2006. Proceedings*, volume 4121 of *Lect. Notes in Comp. Sci.*, pages 424–429. Springer Verlag, 2006.

[vdN09] M. van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond, 2009. arXiv:0811.0898.

[VdNDDM04] Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. Graphical description of the action of local clifford transformations on graph states. *Phys. Rev. A*, 69:022316, Feb 2004.

[VMH03] George F Viamontes, Igor L Markov, and John P Hayes. Improving gate-level simulation of quantum circuits. *Quantum Information Processing*, 2(5):347–380, 2003.

[VRMH03] George F Viamontes, Manoj Rajagopalan, Igor L Markov, and John P Hayes. Gate-level simulation of quantum circuits. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, pages 295–301. ACM, 2003.

[VZGG13] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

[WB11] H. Weimer and B. Butscher. libquantum 1.1.1: the C library for quantum computing and quantum simulation. http://www.libquantum.de/, 2003–2013 (v. 1.1.1).

[WH] D. Wybiral and J. Hwang.

[Wil12] V.V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. 44th Annual ACM Symposium on the Theory of Computing*, pages 887–898, 2012.

[WML$^+$10] H. Weimer, M. Müller, I. Lesanovsky, P. Zoller, and H.P. Büchler. A Rydberg quantum simulator. *Nature Physics*, 6:382–388, 2010.

[Woo98] Alan R. Woods. Unsatisfiable systems of equations, over a finite field. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, pages 202–, Washington, DC, USA, 1998. IEEE Computer Society.