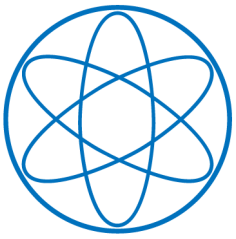

Quantum computation with commuting operations

Master Thesis
Xiaotong Ni

October 19, 2012

Garching

First reviewer:
Prof. Dr. J. Ignacio Cirac

Second reviewer:
Prof. Dr. Alejandro Ibarra

Introduction

Quantum physics has changed our world dramatically, from Laser pointers in our hand to our understanding of black holes. However, it is only lately that people start to ask whether quantum mechanics can be employed to make better computers or other information processing instruments. Around 1980, Richard Feynman and other scientists [1, 2] started to consider the possibility of using quantum computers to simulate interacting quantum systems, since in general it takes exponentially many coefficients to describe a many body system, which makes simulating quantum systems on a classical computer an infeasible task. In 1994, Peter Shor shows that quantum computers can factor integers much faster then any known classical algorithm [3], which also breaks a widely used cryptography protocol on the Internet. After that, quantum computing starts to attract people's interest, and physicists have spent huge efforts to build a quantum computer. Many different experimental implementations have been proposed, by using ion traps [4], superconducting circuits [5], etc. However, so far, only very small quantum computers have been built, which can achieve tasks like factoring 15 into 3×5 (see, for example [6]).

Besides the practical aspect, there is another reason that makes quantum computing interesting. Before quantum computation was known, evidences were pointing to the direction that there is no physical machine able to do computation substantially faster than an electronic computer (more exactly, a Turing machine). This is the so called "extended Church-Turing thesis" (also called "strong Church-Turing" thesis, see [7]). Note that the conjecture is rather about physical laws, and has very similar spirit as thermodynamic laws. Thus if we prove quantum computers can outperform classical ones, then we have to abandon our long-time belief; otherwise we will have more faith in it. There are also some interesting results on deriving physical laws from the computational power aspect. For example, it was shown in [8] that certain kinds of non-linearity in quantum mechanics would drastically increase its computational power, thus considered to be unlikely.

However, it turns out that it is very difficult to characterize the power of quantum computers. For example, while the quantum algorithm for factoring performs exponentially faster than the best *available* classical algorithm, it is extremely hard to prove this for every *possible* classical algorithm. Indeed, if we can prove there is a (decision) problem which can be efficiently solved by quantum computers but not by classical ones, we can solve some long-time open problems in complexity theory (see chapter 2). On the other hand, people also try to find an efficient classical factoring algorithm for decades, yet no one succeeded. Therefore quantum computers are likely to be more powerful than classical ones. Despite the difficulties to obtain rigorous proofs, the widely accepted conjecture is that quantum computers have the edge over classical ones, while at the same time they are not powerful enough to solve a large family of important problems (the family of **NP**-Complete problems).

At the same time, it should also be kept in mind that quantum computers are useful to

speed-up certain tasks, though not very drastically. For example, it is known that quantum computers can do brute-force search faster then classical computers, which is the so called Grover algorithm [9]. The speed-up is only of a factor $\sqrt{N}$, where N is the number of items to be searched. This does not qualify as a substantial speed-up in computational complexity theory. But if some day scalable quantum computers become easy to build, then these kind of algorithms will certainly become handy.

In this thesis, instead of studying the power of quantum computing directly, we will focus on restricted classes of quantum circuits and analyse their power (quantum circuits are a standard model of quantum computation). In most situations, a restricted class of quantum circuits cannot approximate every unitary operator, which is in contrast to normal universal quantum circuits. Studying restricted quantum circuits helps us understanding the power of quantum computation in the following sense: if we want to have an upper bound of the computational power of quantum computers, we must be able to obtain the upper bound for all restricted class; On the other hand, if some restricted quantum circuits are already able to achieve certain tasks that are infeasible on classical computer, then we have more evidence that quantum computers are more powerful.

A number of previous works have investigated restricted quantum circuits. The results can be broken up in two categories: efficient classical simulations and hardness results. First, for several restricted but nontrivial classes of quantum circuits, it has been found that efficient classical simulations are possible. For instance, if in each step of a quantum circuit the entanglement (quantified by the p-blockedness [10] or by the Schmidt rank [11]) is bounded, the circuit can be simulated efficiently classically. Such results demonstrate that certain types of entanglement must be generated in sufficiently large amounts if a quantum algorithm is to yield an exponential speed-up. Certain other circuit classes can be simulated classically using entirely different arguments not based on entanglement considerations [12, 13, 14, 15, 16, 17], e.g. by using the Pauli stabilizer formalism [12] or the framework of matchgate tensors [13, 14, 15] .

Conversely, it has been shown that some restricted quantum computation schemes can perform tasks that appear to be hard classically [18, 19, 20, 21, 22]. For example, in [18] the hardness of simulating linear optical quantum computation was discussed. In [21, 22] it was shown that constant depth quantum circuits are hard to simulate exactly or with a high accuracy. Besides theoretical importance, these results also lower the threshold to demonstrate nontrivial quantum computation in experiments.

In this thesis we focus particularly on commuting quantum circuits. Commuting quantum circuits consist of pairwise commuting operations. In other words, we want to know if we have a device capable of applying pairwise commuting operations on a product state, whether we can use the device to achieve computational tasks which are not easy on a classical computer. Several features make such circuits interesting. For example, commuting circuits exhibit genuine quantum effects, e.g. they can generate entangled pure states, which is a necessary condition to achieve quantum computation. Further, since commuting operations can be performed simultaneously, there is no time order in the computation, which is drastically different from other computational models. Consider that on a classical computer, it will even be very hard to compute $(1 + 2) \times 3$ when we do not have time order in the operations, since we need to do addition before the multiplication. Moreover, all gates in the circuit can be diagonalized simultaneously. The latter property might at first sight suggest an intrinsic simplicity of this circuit class; however it is important to note that the diagonalizing unitary can by a complex entangling operator. The main message of this thesis is there exist evidence that, in spite of the simplicity, commuting circuits indeed have nontrivial power beyond classical computation.

We will use the word "qubit" to denote a 2-level quantum system, and "qudit" for a d-level one. Our main results can be summarized as follows

- *2-local circuits are easy.* Consider n qudits which are initialized in a product state. If m commuting operations are applied to them, where each operation only act on two qudits, then we can simulate any measurement outcome on a single qudit efficiently on a classical computer. More precisely, the time we need to do the simulation is a polynomial of n and m.

- *3-local circuits are hard.* In the above setting, if each operation can act on three qudits instead of two, then we are unlikely to be able to simulate the measurement outcome efficiently to a certain accuracy. In fact, to simulate the measurement outcome for qubits is already hard. This result is based on a widely accepted conjecture in computer science.

- *Commuting Pauli circuits.* Now instead of operations acting on two or three qudits, we consider the commuting operations of the form $e^{i\theta P}$ on qubits, where P is a Pauli operator (i.e., P can be written as a tensor product of Pauli matrices). We show that in this case, we can simulate the measurement outcome on a single qubit efficiently. Note that here the commuting operation $e^{i\theta P}$ can act on any number of qubits.

- *Mapping non-commuting circuits to commuting circuits.* Certain non-commuting quantum processes can be efficiently simulated by purely commuting quantum circuits. This provides further evidence of the nontrivial computational power of commuting circuits.

It is noteworthy that several distinct techniques were used to prove the above results, including the Pauli stabilizer formalism used in quantum error correction, tensor network methods used in condensed matter physics, and recently introduced sampling methods for simulating quantum circuits.

The structure of the thesis is the following:

- In chapter 1 we will introduce the circuit model of quantum computation. It is the most widely used model in quantum computation, and it will give us a way to characterize the time and space resources needed for certain quantum computations. We also introduce the concept of universality. A universal gate set is a set of basic unitary operations that allow to approximate every complex unitary operator. Two alternative models of quantum computing are also mentioned in this chapter, namely measurement based quantum computation and adiabatic quantum computation.

- In chapter 2 we give a brief tutorial of complexity theory. It allows us to compare the power between classical computers and quantum computers formally. In particular, it let us understand the difficulties and importance of studying the power of quantum computing, since to settle it we need to solve some long time open problems in complexity theory.

- In chapter 3 we give a summary of many important classical simulation results. It is shown that many different kinds of quantum circuits can be simulated on a classical computer by using very different techniques, including stabilizer formalism, tensor contraction method, sampling method, etc. It is worth noting that these techniques are also very useful in other subjects of quantum physics.

- Finally in chapter 4 we will study the power of commuting quantum circuits, which is our contribution:

– X. Ni and M. Nest, Commuting quantum circuits: efficient classical simulations versus hardness results, accepted by *Quantum Information & Computation*, preprint arXiv:1204.4570, 2012.

And the following talk was given:

– X.Ni, Commuting quantum circuits: efficient classical simulations versus hardness results, *9th Central European Quantum Information Processing Workshop*, Bratislava, June 7th, 2012

Contents

Chapter 1

Circuit Model of Quantum Computation

Nowadays **quantum computation** has become a well-known concept. Intuitively, when we have control over n two-level quantum systems, we can apply operations over 2^n coefficients, which is far beyond the capability of classical computation. Many different experimental implementation have been proposed. So to study the power of quantum computation, some concise theoretical models is needed to save us from going into detail of every experiment. The model should also represent the full potential of quantum computation, i.e. no experimental setting could solve problems substantially more faster than the model.

In this chapter we focus on the circuit model, which is the most widely used model to represent quantum computation. Before we start, we first introduce some symbols which are mainly used to describe how an algorithm performs asymptotically. We will use $\text{poly}(n)$ to represent an arbitrary polynomial of n. When we say $f(n) = O(g(n))$, we mean that $|f(n)| < cg(n)$ for some constant c when n is sufficiently large.

1.1 Definition of Circuit Model

Let us begin with an example of quantum circuit (see Fig. 1.1). In a quantum circuit, unless explicitly stated, each wire will represent a qubit (i.e., a two-level quantum system). We will always use $|0\rangle$ and $|1\rangle$ to represent the basis of a qubit, which is called as the computational basis. So the state of a single qubit can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{1.1}$$

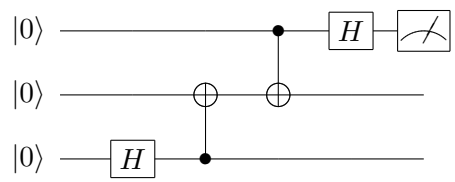

Figure 1.1: A quantum circuit

Similarly, if we have n qubits, the state can be written as

$$|\psi\rangle = \sum_{x_1 x_2 \cdots x_n} c_{x_1 \cdots x_n} |x_1 \cdots x_n\rangle, \tag{1.2}$$

where the sum is over all binary strings $x_1 \cdots x_n$, and $c_{x_1 \cdots x_n}$ is a set of exponentially many coefficients describing the state. The basis formed by all $|x_1 \cdots x_n\rangle$ is the computational basis for n qubits. We will also use the word qudit to describe a d-level quantum system.

Unitary operators are represented by gates. In circuit 1.1, for instance, there are Hadamard gates (H) and control-not gates (CNOT), where the later ones are represented by the lines across two wires. Here are some common gates used in quantum circuits:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \; Y = \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix} \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \; S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \tag{1.3}$$

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{CPHASE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{1.4}$$

where X, Y, Z are Pauli gates, and CZ is called control-phase gate. CNOT is a very widely used gate in quantum circuits. A CNOT gate acting on qubit 1 and 2, controlled by qubit 1 does the following

$$|x_1\rangle|x_2\rangle \to |x_1\rangle|x_1 \oplus x_2\rangle, \tag{1.5}$$

where $x_1 \oplus x_2$ is the addition over $\mathbf{Z}_2$. When represented in circuit 1.1, the qubit on the dot side is the control qubit. We say a gate is a k-local when it only act on k qubits. We will also use Toffoli gate in this thesis, which is a 3-qubit gate that does the following

$$|x_1\rangle|x_2\rangle|x_3\rangle \to |x_1\rangle|x_2\rangle|x_3 \oplus x_1 x_2\rangle, \tag{1.6}$$

where $x_1 x_2 x_3$ is any binary string.

In the most cases a state chosen from the computational basis will be taken as input. In the end of a quantum circuit, a measurement will be done to obtain a classical outcome from a quantum state. Usually it will be a one qubit measurement on the first qubit.

Although we already have a way to describe quantum circuits, some details are still missing for it to be a computation model. For example, a computation problem can normally have bit strings of any length as input, while a specific quantum circuit C_n only allows bit string of length n as input, where n is the number of input qubits. So to implement quantum computation, we need a family of quantum circuits to deal with input of different length. We also need to specify how we generate the description of the quantum circuits with different number of input qubits. For instance, if we have an unlimited computational power to generate the description of the quantum circuits based on the input, the circuits would also have unlimited power. This is simply because we can use the unlimited computational power to first compute the solution to the input, and then generate a possibly very short circuit which outputs the known solution. Moreover, when we build a quantum computer in the real world, we do not have access to any unlimited computational power. So to avoid this issue, we imply the uniformity condition: the quantum circuits with n input qubits can be generated by a classical computer in $poly(n)$ time. Below is the formal definition of uniform family of quantum circuits:

Definition 1.1.1. *A uniform family of quantum circuits is a set of $\{C_n\}$, where C_n is a quantum circuit with n input qubits. Each of the input qubit can be $|0\rangle$ or $|1\rangle$. The description of C_n can be computed by a classical computer in poly(n) time. Possible operations are*

- *gates that act on at most 2 qubits, chosen from a finite set $\{G\}$.*
- *creating some auxiliary qubits, initialized in $|0\rangle$*

In the end of a circuit, a one qubit measurement will be done on the first qubit. We will use the terminology "size of a circuit" to denote how many gates are used in the circuit. Note that the size of a uniform circuit family is always of poly(n).

Although this is the most often used definition of quantum circuits, we will often use variants of it during this thesis. For example, we might run the same quantum circuits multiple times, and then use a classical computer to do some post-processing on the outcomes. We will also study some 3-local circuits instead of 2-local ones. However, if we have a universal set of quantum gates in the definition 1.1.1, these variants can be shown to have the same power. We will introduce the concept of universality first.

1.2 Universal sets of quantum gates

In this section, we will discuss why in the above definition we can restrict ourself to choose from a finite set of gates acting on two qubits. For this, we need the concept of universality.

Definition 1.2.1. *A set of gates is called universal if any unitary operator can be approximated by a circuit of these gates to an arbitrary accuracy. More precisely, for a unitary operator U (on arbitrary number of qubits) and $\epsilon > 0$, there exist a circuit C of these gates, such that $\|U - C\| < \epsilon$. Here $\|\cdot\|$ is the spectral norm, which is the largest singular value of the operator.*

In the above definition, we also use C to represent the unitary operator implemented by circuit C. Note that we did not address the efficiency of approximating a gate. Indeed, it has been proved that there exist a family of gates $\{U_n\}$, which acting on n qubits respectively, so that U_n requires an exponential number of gates to approximate [7].

To show that this is a good definition of universality, we also need the fact that the final state of the circuit $C|\psi\rangle$ is close to the desired state $U|\psi\rangle$. Indeed, by the property of spectral norm, we have

$$\|A\| = \sup_{|\alpha\rangle} \|A|\alpha\rangle\|. \tag{1.7}$$

Particularly, this implies

$$\|C - U\| = \sup_{|\alpha\rangle} \|(C - U)|\alpha\rangle\|, \tag{1.8}$$

which means for any $|\psi\rangle$, if C and U are ϵ-close, $C|\psi\rangle$ and $U|\psi\rangle$ will be ϵ-close as well. Moreover, we have the following theorem

Theorem 1.2.2. *Let $|\eta_1\rangle = C|\psi\rangle$, $|\eta_2\rangle = U|\psi\rangle$ and $|\Delta\rangle \equiv |\eta_1\rangle - |\eta_2\rangle$. We have for any projector P,*

$$|\langle\eta_1|P|\eta_1\rangle - \langle\eta_2|P|\eta_2\rangle| \leq 2\|U - C\|. \tag{1.9}$$

Since $\langle\eta|P|\eta\rangle$ corresponds to the probability of getting certain outcome during a measurement, we know that when C and U are ϵ-close, the probability distribution of measurement outcomes will also be close. This theorem can be proved easily:

Proof.

$$|\langle\eta_1|P|\eta_1\rangle - \langle\eta_2|P|\eta_2\rangle| \leq |\langle\eta_1|P|\Delta\rangle + \langle\Delta|P|\eta_2\rangle| \tag{1.10}$$
$$\leq |\langle\eta_1|P|\Delta\rangle| + |\langle\Delta|P|\eta_2\rangle| \tag{1.11}$$
$$\leq 2\||\Delta\rangle\| \tag{1.12}$$
$$\leq 2\|U - C\| \tag{1.13}$$

□

The concept of universality also appears in classical computation, where people have shown that with the logical operation AND and NOT we can do all classical computation. This is basically the reason that we can use the same chips in our computer to run different kinds of softwares. Also, Toffoli gate (1.6) alone is universal for classical computation.

Surprisingly, while the Hilbert space is much larger then classical world, it is fairly easy to find a universal set of quantum gates. we have the following theorem:

Example 1.2.3. *H,S,CNOT form a universal set of gates.*

The proof of this can be seen in [7]. The outline of the proof is showing H and S is capable to approximate any single qubit gate, and together with CNOT any unitary operator can be approximated. However, CNOT is far from the only 2-local gate which makes quantum computation work. In [23], a necessary condition for a set of gates to be universal is presented, as following:

Theorem 1.2.4. *Define 2-local gate P to be $P|x\rangle|y\rangle = |y\rangle|x\rangle$. Then the set $SU(2)$, containing all single qubit gates, plus any single 2-qubit gate that cannot be written as $U_1 \otimes U_2$ and $U_1 \otimes U_2 P$, is a universal set of gates.*

In the above theorem, U_1 and U_2 are arbitrary single qubit gates. A direct consequence of this is all single qubit gates, together with any 2 qubit gate which can generate entangled states from product states, are a universal set of gates. Here a two qubit entangled state is a pure state which cannot be written as a product state $|\alpha_1\rangle \otimes |\alpha_2\rangle$, and it is clear that a gate able to generate entangled state cannot be of the form $U_1 \otimes U_2$ (or $U_1 \otimes U_2 P$). So to implement quantum computation in experiments, we only need to have some non-trivial interaction between two qubits as well as the full control on each single qubit. This is not a very high requirement. The reason people still have not built a large quantum computer is due to the errors during the computation process.

Moreover, Solovay-Kitaev algorithm [24] guarantees under some condition, any k-local gates can be approximated with a high accuracy efficiently by 2-local gates. Here "efficiently" means the gates needed to approximate a k-local gate scales $O(log^4\epsilon)$ with the accuracy ϵ, and k is treated as a constant here.

Theorem 1.2.5. *Let G be a universal set of 2-local gates, with the condition that $U \in G \Leftrightarrow U^\dagger \in G$. Let k be any constant, then there is an efficient algorithm that for any k-local gate V and $\epsilon > 0$, a sequence $U_j \in G$ with $1 \leq j < q = O(log^4\epsilon)$ can be found, such that*

$$\| \prod_{1 \leq j \leq q} U_j - V\| < \epsilon. \tag{1.14}$$

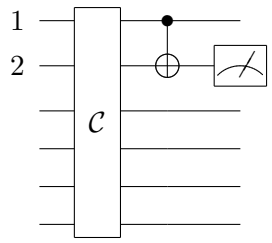

Figure 1.2: The structure of circuit C'

Again, the norm here is the spectral norm. Together with the discussion above, we know that $O(2^{\text{poly}(k)}, \log \epsilon)$ gates are needed to approximate a k-local gate with error ϵ. As an application, we can show that with a universal set of quantum gates, we can achieve classical computation efficiently. To prove this, we need the fact that Toffoli gate (1.6) is universal for classical computation. In fact, for any boolean function family $f : \{0,1\}^n \to \{0,1]\}$ that can be computed in poly(n) time, there is a poly(n) size circuit with only Toffoli gates that maps input (x, y) to $(x, y + f(x))$ [7]. Since Toffoli gate is also a unitary quantum gate, each Toffoli gate can be replaced by poly(n) gates from the universal set with a exponentially small error in n, we conclude that quantum circuits can approximate any classical computation with at most a poly(n) overhead.

1.3 Variants of quantum computation

There are many variants of the standard quantum circuit definition. For example, we can use qutrits (3-level systems) instead of qubits, or use 3-qubit gates instead of 2-qubit gates. However, it is not hard to show that all these variants can be simulated by the standard circuits with at most a polynomial overhead, and a polynomial difference is usually not considered to be substantial (see chapter 2). Also, we can do multi-qubit measurement in the end of circuit with some post processing on classical computer. This issue is again not essential here for two reasons. First in complexity theory, decision problems, which solution is only 0 or 1, are of the main interest (see chapter 2 for more discussion). Moreover, the classical post processing can also be done in the quantum circuit before measurement. Here we will use a simple example to illustrate how this can be done in principle.

Example 1.3.1. *Let C be a quantum circuit. We do a measurement on the first two qubits in the computational basis, and let the result be x_1 and x_2. Then we output the binary sum $x_1 \oplus x_2$. This procedure can be turned into another circuit C' with a single qubit measurement.*

We can construct C' from C as figure 1.2. Assume that after circuit C, the final state is

$$\begin{aligned} |\psi\rangle =& \alpha_{00}|0\rangle_1|0\rangle_2|\psi_{00}\rangle + \alpha_{01}|0\rangle_1|1\rangle_2|\psi_{01}\rangle && (1.15)\\ &+ \alpha_{10}|1\rangle_1|0\rangle_2|\psi_{10}\rangle + \alpha_{11}|1\rangle_1|1\rangle_2|\psi_{11}\rangle && (1.16)\\ =& \sum_{x_1,x_2} \alpha_{x_1x_2}|x_1\rangle_1|x_2\rangle_2|\psi_{x_1x_2}\rangle && (1.17) \end{aligned}$$

Here the first two qubit 1 and 2 are measured in the original circuit C, and the probability to get x_1x_2 is $|\alpha_{x_1x_2}|^2$. For circuit C', instead of doing the measurement on qubit 1 and 2 immediately,

we apply a CNOT gate on these two qubits, controlled on qubit 1. The state becomes

$$\sum_{x_1,x_2} \alpha_{x_1x_2}|x_1\rangle_1|x_1 \oplus x_2\rangle_2|\psi_{x_1x_2}\rangle. \tag{1.18}$$

Now if we measure the qubit 2, we get the outcome 0 with probability

$$p = |\alpha_{00}|^2 + |\alpha_{11}|^2, \tag{1.19}$$

which is exactly the same probability of getting 0 when running circuit C and classical post-processing. This procedure can be easily generalized to the case with general classical post-processing, since we can always replace the classical computation part with an approximate quantum circuit before the measurement, and then use a similar argument as above.

Circuit model is far from the only way of doing quantum computation. Two of the alternatives are measurement based quantum computation [25] and adiabatic quantum computation [26]. In the following subsections we will give a very brief introduction to these two models, and some basic ideas of how to show they are equivalent to the circuit model. Interested readers are suggested to read the references, since details are missing in our treatment. The reader may also skip these two subsection during the first reading of this thesis, since they are somewhat disconnected from the main text.

1.3.1 Measurement based quantum computation (MBQC)

In the quantum circuit model, we apply a sequence of gates on some product input state, and which gates we apply depend on the specific problem. The scheme of MBQC is very different. In MBQC, instead of having a product state as input, we have an entangled cluster state; and instead of having different circuits for different problems, we have a different measurement patterns of single qubit measurements. A measurement pattern includes the time order of measuring individual qubits, and how the measurement basis is chosen, which may depends on previous outcomes.

More precisely, the qubits in MBQC are usually arranged as a square lattice. Except for input qubits, all others are initialized to state $|0\rangle$. To obtain the cluster state, we first apply Hadamard gates on every qubit (except input), and then apply control-phase gates on each pair of adjacent qubits. To see how MBQC can achieve universal quantum computation, first we look at a simple example. Consider the state $(\alpha|0\rangle + \beta|1\rangle)|0\rangle$. After applying Hadamard gate on the second qubit and a control-phase gate, we get

$$|\psi\rangle = \alpha|0\rangle|+\rangle + \beta|1\rangle|-\rangle, \tag{1.20}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then by measuring the first qubit in the basis $\{\frac{1}{\sqrt{2}}(|0\rangle \pm e^{i\phi}|1\rangle)$, the second qubit will be projected to

$$X^m HU_z(\phi)(\alpha|0\rangle + \beta|1\rangle), \tag{1.21}$$

where $m = 1$ if the first qubit is projected into $\{\frac{1}{\sqrt{2}}(|0\rangle - e^{i\phi}|1\rangle)$, and $m = 0$ otherwise. Assuming that m is always equal to 0, then we have a way to implement single qubit gate $HU_z(\phi)$. This procedure can be concatenated to achieve any one qubit gate by using different ϕ each time.

To address the X^m error, we need to do adaptive measurements, which are measurements with basis depend on the previous measurement outcomes. Again use the example of single

qubit gate above. Assuming we want to obtain the state $HU_z(\phi_2)HU_z(\phi_1)|0\rangle$, and we get an X error when we do the first measurement. In this case we end up with $XHU_z(\phi_1)|0\rangle \equiv X|\psi\rangle$ after the first measurement. By noticing that $XZ = -ZX$ and $HX = ZH$, we can set the second gate to be $HU_z(-\phi_2)$ instead of $HU_z(\phi_2)$. So now assuming we do not have an X error when we do the second measurement, we will end up with

$$HU_z(-\phi_2)X|\psi\rangle = ZHU_z(\phi_2)|\psi\rangle. \tag{1.22}$$

Therefore we can obtain the desired states with some possible Pauli errors (i.e., errors in the form of X, Y, Z and some constants). It is not hard to see by the same technique we can implement all single qubit gates with Pauli errors. Note that if the final measurement of the computation is done in the Z basis, then we can recover the desired output distribution even when we have some known Pauli errors. For example, if the outcome is 1 and we know there is an X or Y error on the qubit, then we can record 0 as the actual outcome.

Note that the above implementation of single qubit gates also give us a way to transfer certain states from one place to another on the square lattice. By theorem 1.2.4, we only need control-phase gates to achieve universal computation. However, notice that to make cluster state, we have applied control-phase gates on all adjacent qubits. It can be shown that we can achieve control-phase operation in MBQC by using these control-phase gates. The basic idea is to use the fact that control-phase gates commute with measurements on other qubits. We can change the order of operations as following:

1. We do measurements on other qubits (as above), transfer two state $|\psi\rangle, |\varphi\rangle$ to two adjacent qubits.

2. Apply the control-phase gate on these two qubits.

3. Measure these two qubits in suitable basis, transfer the two states to other sites of the lattice.

Note that Pauli operators can propagate through CZ gates in a similar way they propagate through H gates. In fact, CZ and H both belongs to the so called Clifford operations, which we will describe in more detail later. By all of these ingredients, we can achieve universal quantum computation with MBQC.

1.3.2 Adiabatic quantum computation

Compared to quantum circuits or MBQC, adiabatic quantum computation [26] has a very different setting. The idea is trying to reach the ground state of a local Hamiltonian which encodes the solution of certain problem. While it is very easy to encode hard computational problems into ground states (see [26] for example), there is no universal way to cool the system to the ground state for different Hamiltonians, and this should not be considered as a simple task.

The adiabatic theorem give us a consistent way to achieve this. It states that for a system starting in the ground state of its Hamiltonian H_0, if the Hamiltonian changes slowly enough and the gap between ground state and other eigenstate is larger then some constant, then the system will end in the ground state of the final Hamiltonian H_1. So we can choose the initial Hamiltonian to have an easy to prepare ground state, e.g. $|00\cdots0\rangle$, and slowly evolve it to the desired Hamiltonian. This is often done in a linear way, by setting $H(t) = (1-t)H_0 + tH_1$ for

$0 \leq t \leq 1$. We know that this procedure is efficient if it can be shown that the gap does not get too small. More precisely, one version of the adiabatic theorem states [27]

Theorem 1.3.2. *Let $H(s), 0 \leq s \leq 1$, be a time dependent Hamiltonian. Let $\psi(s)$ be one of its eigenstates, and let $\gamma(s)$ be the corresponding eigenvalue. Assume that for any $s \in [0,1]$, all other eigenvalues of $H(s)$ are either smaller than $\gamma(s) - \lambda$ or larger than $\gamma(s) + \lambda$. Consider the adiabatic evolution given by H and ψ applied for time T. Then, the following condition is enough to guarantee that the final state is at distance at most δ from $\psi(1)$:*

$$T \geq \frac{10^5}{\delta^2} max \left\{ \frac{\|H'\|^3}{\lambda^4}, \frac{\|H'\| \cdot \|H''\|}{\lambda} \right\}. \tag{1.23}$$

While this looks like a total different model than the circuit model, it can be proved that if the Hamiltonians during the adiabatic process is local, then the computational power of adiabatic computation and circuit model are equal [28]. This furthermore confirms that circuit model is a suitable model to study the power of quantum computation.

The proof in [28] is quite complex. Here we will give a brief review of how to transform a (polynomial size) quantum circuit into an adiabatic quantum computation.

The first difficulty we met is that to do adiabatic quantum computation, we need to know the final state $|\psi\rangle$ before we can assign a Hamiltonian for it. However, the state $|\psi\rangle$ is computed by the quantum circuit, which means it is very hard to have a description for $|\psi\rangle$ on the classical computer. This difficulty is solved by realizing we do not need the final state of adiabatic computation to be the exactly same final state of the quantum circuit: they only need to have some reasonable overlap. More precisely, assume the 2-local gates in the circuit to be $U_1, \cdots U_L$, and the states after lth gate to be $|\alpha(l)\rangle$. Then we set the final state of the adiabatic quantum computation to be

$$|\eta\rangle = \frac{1}{\sqrt{L}} \sum_{0 \leq l \leq L} |\alpha(l)\rangle \otimes |1^l 0^{L-l}\rangle^c, \tag{1.24}$$

where $|1^l 0^{L-l}\rangle^c$ has first l qubits to be $|1\rangle$ and the other $L-l$ qubits to be $|0\rangle$ (the c here stands for clock). We will call qubits in $|1^l 0^{L-l}\rangle^c$ clock qubits, since they serve as the label of which step $|\alpha(l)\rangle$ is in the quantum circuit. Assuming we can reach state (1.24), then we can first measure the clock qubits. If they are in the state $|1^L\rangle$, then other qubits will end up being $|\alpha(L)\rangle$, which is the final state of the quantum circuit. This procedure succeed with a probability $\frac{1}{L+1}$. Therefore we only need to repeat this procedure poly(L) times to have a very high success probability, which is considered to be efficient (see the next section for more discussion on efficiency).

The initial and final Hamiltonian H_0 and H_1 are designed in such a way that they give penalty to states do not have the desired form. For the initial Hamiltonian H_0, we can set it to be

$$H_0 = H_{\text{clockinit}} + H_{\text{input}} + H_{\text{clock}}. \tag{1.25}$$

Noting that we set H_0 in this way not only to ensure the ground state to be in state $|0^n\rangle \otimes |0^L\rangle^c$, but also for later use. H_{input} is used to ensure at step 0, $|\alpha(0)\rangle = |0^n\rangle$. This is achieved by setting

$$H_{\text{input}} = \sum_{1 \leq j \leq n} |1\rangle\langle 1|_j \otimes |0\rangle\langle 0|_1^c, \tag{1.26}$$

where $|1\rangle\langle 1|_j$ act on the jth qubit of $|\alpha\rangle$ and $|0\rangle\langle 0|_1^c$ acts on the first clock qubit. H_{clock} is used to ensure the clock qubits are in the form $|1^l 0^{L-l}\rangle^c$:

$$H_{\text{clock}} = \sum_{1\leq l\leq L} |01\rangle\langle 01|_{l,l+1}^c. \tag{1.27}$$

So if any 0 and 1 in the clock qubits are in the reverse order 01, the state will receive an energy penalty. $H_{\text{clockinit}}$ is to make sure the clock qubits starting with $|0^L\rangle$, which simply is

$$H_{\text{clockinit}} = |1\rangle\langle 1|_1^c. \tag{1.28}$$

Similarly, we break down H_1 into three parts:

$$H_1 = \frac{1}{2}\sum_{1\leq l\leq L} H_l + H_{\text{input}} + H_{\text{clock}}, \tag{1.29}$$

where $H_{\text{input}}, H_{\text{clock}}$ is the same as in H_0. The Hamiltonian H_l is used to check whether $U_{l+1}|\alpha(l)\rangle = |\alpha(l+1)\rangle$:

$$H_l = I \otimes |100\rangle\langle 100|_{l-1.l,l+1}^c - U_l \otimes |110\rangle\langle 100|_{l-1.l,l+1}^c \tag{1.30}$$

$$- U_l^\dagger \otimes |100\rangle\langle 110|_{l-1.l,l+1}^c + I \otimes |110\rangle\langle 110|_{l-1.l,l+1}^c. \tag{1.31}$$

It can be verified that $|\eta\rangle$ in equation (1.24) is a ground state of H_1. It remains to be check that $|\eta\rangle$ is the only ground state and during the adiabatic process $H(t) = (1-t)H_0 + tH_1$ the gap do not get too small. These two parts are very technical. So we refer the readers to the original paper if interested.

Chapter 2

Introduction to Computational Complexity

The main reason of building a quantum computer is the hope that quantum computers can outperform classical ones. Indeed Shor's algorithm gives us evidence to support this conjecture. In this section we introduce the framework of complexity theory, which will give us a rigorous definition of what "outperform" means.

Computational complexity theory divides problems into many subclasses. From this perspective, it provides a coarse-graining of computational problems. This is much needed when we study some complex problems. For instance, while it is possible to calculate how many steps are necessary to add two numbers, it is neither possible nor necessary to know exactly how many steps are needed for Google to perform a single search. Indeed, it is already very hard to do this for some simple tasks. As an example, calculating the minimum steps required to do matrix multiplication has been an open problem for decades. In complexity theory, when dealing with some unfamiliar problems, we first try to sort them into those well studied subclass, which in turn let us know a lot of important properties of the problems.

In complexity theory, decision problems (i.e., output consists only a single bit 0 or 1) is of the most interest. When the output is 1, we say that the input is accepted or it has a positive answer. Many problems can be reduced to the form of decision problems.

The class of decision problems that can be solved efficiently by classical computer is considered to be class **P**. It consists of problems that have algorithms running in polynomial steps (of input size). Here "one step" means an elementary operation, e.g. a Toffoli gate. One good way to understand this is using the analogue to the desktop computers: one step corresponds to a basic operation in computer language like C. A rigorous definition of **P** would involve the use of Turing machine (see [7] for the detailed discussion). The reason polynomial time is regarded as efficiency is polynomial grows relatively slow when input size n gets bigger, especially when compared to the growth speed of exponential. Another empirical reason is that most useful polynomial time algorithms run in a time of small exponent of n. That is, it is very hard to find a natural problem in **P** that takes $O(n^{20})$ time to solve.

On the other hand, a lot of realistic problems that are not known to have a polynomial time algorithm are in the class **NP**, which are the decision problems that can be verified efficiently. More precisely, when a decision problem in **NP** has positive answer, there is a polynomial length string that allows to verify that the answer is indeed positive by a polynomial time algorithm. Informally, the relation between **P** and **NP** is like the relation of proving theorem and checking

whether the proof is valid. As an example, while it is extremely hard to find the proof for Fermat's last theorem, many mathematicians can check the proof in a reasonable time after it is found. Another example of a problem in **NP** is the factoring problem, which has a very close relation to quantum computation.

Example 2.0.3. *The factoring problem is given a number $N = x_1x_2\cdots x_n$, we find its unique decomposition*

$$N = p_1^{j_1}p_2^{j_2}\cdots p_l^{j_l}. \tag{2.1}$$

Here $x_1x_2\cdots x_n$ is the binary representation of N, and $p_1,\cdots,p_l$ is a set of prime number. It can be transformed into a decision problem $\mathcal{F}$ in the following way. The decision problem $\mathcal{F}$ is that given a number N and an integer k, the output is the kth last digit of N's smallest prime divisor p. For example, if $k = 2$ and $p = y_1y_2\cdots y_m$, then the output is y_{m-1}. If here $k > m$, the output is simply 0. It can be seen that we only need solve problem $\mathcal{F}$ polynomial times to get the decomposition (2.1), by getting the prime divisor p of N and then solving $\mathcal{F}$ with the number N/p. This means if we can solve the decision problem $\mathcal{F}$ efficiently, we can also do factoring efficiently.

*It is worth noting that when we say we solve $\mathcal{F}$ efficiently, we mean solve $\mathcal{F}$ in poly(n) time rather then poly(N) time, since the input size of $N = x_1x_2\cdots x_n$ is n. Indeed, it is very easy to find an algorithm in poly(N) time, while people do not know whether there is an algorithm in poly(n) time. However, it can be seen that the factoring problem $\mathcal{F}$ in **NP**, since the smallest divisor p of the number N can serve as a certificate. (This also relies on the fact that checking whether a number is prime is in **P**)*

It is easy to notice that **P**⊆**NP**, since for any problem in **P**, we can "verify" it has a positive answer by solving the problem directly. Thus it is not guaranteed that a problem is hard when it is in **NP**. To capture the hardness of **NP**, people introduce the class **NP-Complete** to be the class of hardest problems in **NP**, in the sense that if you can solve one of them efficiently, you can solve all problems in **NP** efficiently. It is guaranteed in [29] that there exist problems in **NP-Complete**. Here is an example of **NP-Complete** problem [30]

Example 2.0.4. ***3SAT*** *is a problem that asks whether certain boolean equation can have value 1 (also termed as whether the boolean function can be satisfied). For a set of boolean variables $x_1,\cdots,x_n$, an instance of **3SAT** problems can be written as*

$$(\neg^{u_{11}}x_{w_{11}} \vee \neg^{u_{12}}x_{w_{12}} \vee \neg^{u_{13}}x_{w_{13}}) \wedge \cdots \wedge (\neg^{u_{m1}}x_{w_{m1}} \vee \neg^{u_{m2}}x_{w_{m2}} \vee \neg^{u_{m3}}x_{w_{m3}}), \tag{2.2}$$

*where $u_{jk} = 0, 1$, $1 \le w_{jk} \le n$, $\neg^0 x = x$ and $\neg^1 x = \neg x$ is the negation. $\vee$ and $\wedge$ is the **OR** and **AND** operation respectively. Then the question is whether there are $x_1,\cdots,x_n$ such that value of (2.2) is 1.*

It is easy to see that when there exist an assignment of $\{x_j\}$ that satisfies the boolean formula, there is a string that let us verify this fact, namely the values of $\{x_j\}$. To prove it is complete for **NP** is much harder, see [30].

At first sight, the fact **3SAT** is **NP-Complete** is not very interesting. However, for example, if we can solve **3SAT** efficiently, we can also solve the factoring problem efficiently, even though they do not seem to have any connection at all! Up to now, hundreds of **NP-Complete** problems have been found, and none of them is known to be solved by a polynomial time algorithm. This leads people to believe **P**≠**NP**, while in the same time people failed to prove it

for decades. This open problem carries huge importance, for many practical problems belongs to **NP-Complete**.

Probabilistic algorithms, on the other hand, output right answer with some probability. To be a useful algorithm, however, the success probability cannot be arbitrary close to $\frac{1}{2}$. In particular, **BPP** is the probabilistic version of **P**. Compared to **P**, there is an additional source of fair coins, and the requirement that the algorithm output the right answer with probability $\frac{2}{3}$. More precisely, a family of function $f_n(x) : \{0,1\}^n \to \{0,1\}$ is in **BPP** when there exist a family of function $g_n(x,y)$ in **P**, such that when y is set to be a random binary string, $P(f_n(x) = g_n(x,y)) \leq 2/3$. Here y is of length poly(n). To see how randomness may help to do computation, we give an example here, albeit not for a decision problem.

Example 2.0.5. *Consider the task of evaluating the integral*

$$\int_0^1 f(x)\mathrm{d}x \tag{2.3}$$

when given a function f. A standard way of doing this is choosing a set of points $x_j = j/n,\ 1 \leq j \leq n$ and computing the sum

$$\frac{1}{n}\sum_{j\leq n} f(x_j). \tag{2.4}$$

However, this method will fail for functions like $f(x) = \sin^2 n\pi x$, which has a periodic structure. Indeed, for any choice of x_j, there will always be some family of functions having a poor result when we use equation (2.4) to evaluate their integral. One way around this is to choose x_j randomly. We can still get unlucky by choosing bad x_j, but there will not be systematic error in this case.

Quantum computers clearly have a probabilistic nature. So the class **BQP**, which consists the problems that can be solved by quantum computers efficiently, are defined in a similar way as **BPP**:

Definition 2.0.6. *The class **BQP** consists of decision problems that can be solved by a uniform family of polynomial size 2-local quantum circuits (each with a single qubit measurement). Here "solve" means the quantum circuits output the correct solution with probability larger then 2/3.*

Now we can discuss some evidence that indicate quantum computers might outperform classical ones, in a formal way. First it can be shown that $\mathbf{P} \subseteq \mathbf{BQP}$. This is because Toffoli gate, which is universal for classical computation, can be implemented in quantum circuits efficiently. Conversely, several evidence suggest that the inclusion is strict.

Example 2.0.7. ***Shor's algorithm*** *[3] is an efficient quantum algorithm for factoring natural numbers, while no such classical algorithm is known. Apart from indicating **P** might not equal to **BQP**, it also break the widely used **RSA** cryptography scheme.*

Most evidence is given in the form of an oracle problem (or black-box problem). Simon's algorithm is one of them.

Example 2.0.8. ***Simon's algorithm*** *is a quantum algorithm which solves a black-box problem exponentially faster than any classical algorithm. The problem is the following: given a function $f : \{0,1\}^n \to \{0,1\}^n$, which is implemented by a black box. It is promised that for a string $s \in \{0,1\}^n$, $f(x) = f(y)$ if and only if $x = y$ or $x \oplus y = s$, for all x and y. The objective*

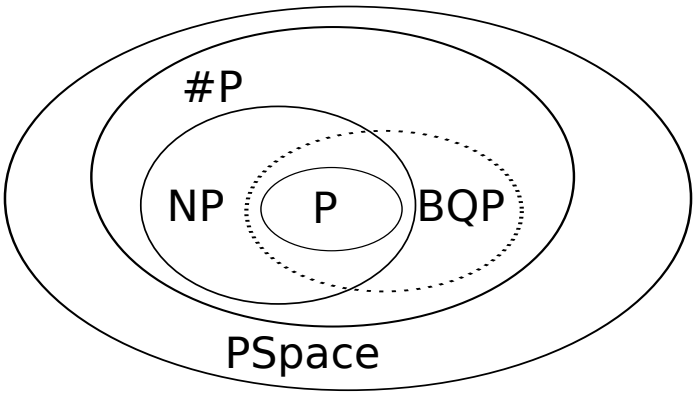

Figure 2.1: The hierarchy of complexity class. The relation between **NP** and **BQP** is still unknown. Also #**P** is not a class of decision problems, so the inclusion here only represent the level of hardness.

is to find s. Simon's algorithm only need to access to the black box polynomial times, while it can be proven that classical algorithms need to access the box exponential times, thus provide an exponential separation between classical and quantum algorithms in this case.

The algorithm is rather simple, nevertheless it is the prototype of many quantum algorithms. Therefore we will give the outline here. First we shall have a more precise definition of the black box. In the classical case, the box will take binary string (x, y) as input and output $(x, y \oplus f(x))$. Correspondingly, in the quantum case, the box is a unitary transformation that maps $|x, y\rangle$ to $|x, y \oplus f(x)\rangle$. The routine of the algorithm is the following:

- Initiate $2n$ qubits to state $|0\rangle|0\rangle$, each $|0\rangle$ is n qubits in the 0 state.
- Apply Hadamard gates to first n qubits. The state ends up with $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|0\rangle$, where the sum of x is taken over all binary string of length n.
- Apply the black box on the state. By the definition of the box, the state is transformed to $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle$
- Act Hadamard gates on first n qubits again. The state becomes $\frac{1}{2^n} \sum_{x,y} (-1)^{x \cdot y} |y\rangle|f(x)\rangle$, where x and y are both summed over all binary strings.
- Measure the first n qubits.

Now assume $s \neq 0$, the coefficient of $|y\rangle|f(x)\rangle$ in the final state is $(-1)^{x \cdot y} + (-1)^{(x+s) \cdot y}$. Thereby the y we measured in the end distribute uniformly in the subspace of $y \cdot s = 0$. Assume that we have run the circuit $k + 1$ times and obtain $y_1, \cdots, y_{k+1}$. If $A = span\{y_1, \cdots y_k\}$ is not the whole $n - 1$ dimension subspace orthogonal to s in $\mathbb{Z}_2^n$, the probability of y_{k+1} not belonging to A is at least $1/2$. Thus, by running the circuit $k =$poly(n) times, we will have the subspace A to be the whole subspace orthogonal to s with a high probability, and from A we can obtain s.

Though Simon's algorithm shows for certain computational tasks quantum algorithms can outperform their classical counterpart, it does not help much in proving $\mathbf{P} \neq \mathbf{BQP}$. The reason is that the black box in the Simon's algorithm hides some of the structure of the problem. For

example, consider the problem of checking whether a number is prime. If instead of being given the number N, we are only given an upper bound M of N and a black box which tells you whether a number k is a factor of N. It is easy to see, in this case, we have to use the black box at least for every $p^2 < M$ to make sure N is a prime. In contrast to the normal efficient primality test algorithm, this procedure is not efficient.

In fact, proving the statement $\mathbf{P} \neq \mathbf{BQP}$ would solve some long-standing open problem about complexity theory. To see this, we need first introduce the complexity class **PSPACE**. **PSPACE** is the class of decision problem that can be solved with a polynomial large space. For example, the space a computer program uses is the part of memory and hard driver it reads and writes during the whole process. Since **PSPACE** does not have any restriction of time, it can contain extremely hard problems. Indeed, it can be readily seen that $\mathbf{NP} \subseteq \mathbf{PSPACE}$, by the fact that we can check all possible solution in a **3SAT** problem in polynomial space. However, even though most people believe $\mathbf{P} \neq \mathbf{PSPACE}$, for about 30 years no one succeed in proving it. Now we will show that $\mathbf{P} \neq \mathbf{BQP}$ would imply $\mathbf{P} \neq \mathbf{PSPACE}$, by proving the following theorem:

Theorem 2.0.9. ***BQP*** $\subseteq$ ***PSPACE***

Proof. Assume the quantum circuit C has the form $C = U_1 U_2 \cdots U_m$, where U_j are unitary operators on 2 qubits. The state is initialized in some computational basis $|x\rangle$. In the end of circuit we measure the first qubit in the computational basis. Thus, the expectation value of the measurement is

$$\langle x|U_m^\dagger \cdots U_1^\dagger Z U_1 \cdots U_m|x\rangle \tag{2.5}$$

We can insert $I = \sum_{x_j=1}^{2^n} |x_j\rangle\langle x_j|$ between every two operators in the equation, results in

$$\sum_{x_1 \cdots x_{2m+2}} \langle x|U_m^\dagger|x_1\rangle\langle x_1| \cdots \langle x_{m+1}|Z|x_{m+2}\rangle\langle x_{m+2}|U_1|x_{m+3}\rangle \cdots \langle x_{2m+2}|U_m|x\rangle \tag{2.6}$$

It is easy to see that each term in the summation can be calculated using polynomially large space, and the summation can be done term by term. Thus the calculation of expectation value can be done in a polynomially large space. Whether x is accepted can be decided based on the expectation value. □

Another class, called $\#\mathbf{P}$, will serve to prove some hardness results in the thesis. It contains counting problems that associated with problems in **NP**. For example, "How many solutions does a 3SAT instance have?" belongs to $\#\mathbf{P}$. In fact, if $f_n(x) : \{0,1\}^n \to \{0,1\}$ are a set of function that can be computed in poly(n) time, then finding the value of

$$\sum_{x \in \{0,1\}^n} f(x) \tag{2.7}$$

is a **#P-Complete** problem. Unlike other complexity class we mentioned above, $\#\mathbf{P}$ is not a class of decision problems, i.e. its solution is a natural number. While this means we cannot write any inclusion like "$\mathbf{NP} \subset \#\mathbf{P}$", it is true that if we can solve a problem in **#P-Complete**, then we can solve all **NP** problems. To see this fact, we can encode the number of solutions to a 3SAT problem in equation (2.7). So if we can calculate the sum efficiently, we can also tell whether there exist a solution to that 3SAT problem, which is a problem in **NP-Complete**.

Chapter 3

Classical simulation of quantum circuits

In chapter 2 we have seen that it is very hard to make any assertion on the power of **BQP**, yet the question is of huge importance. In this chapter, we try to understand what empowers quantum computation, by investigating some subclass of quantum circuits. We will see some of them can be simulated by a classical computer efficiently, though they can generate very complex states. We will first discuss what classical simulation of quantum circuits means. Then we will see several examples of quantum circuits which can be simulated efficiently by classical means.

3.1 Definition of classical simulation

To find a definition for classical simulation is not an trivial task. Indeed, "classical simulation" often has different meanings in different articles. Here are two types of classical simulation we will encounter frequently in this thesis:

- **Strong simulation**: We say that a family of circuits $\mathcal{C}_n$ can be efficiently simulated classically in the strong sense if there exists a classical algorithm with runtime $\text{poly}(n, \log \frac{1}{\epsilon})$ which outputs a number E such that

$$|E - \langle Z_1 \rangle| \leq \epsilon. \tag{3.1}$$

  Here Z_1 is the Z operator on the first qubit, and $\langle Z_1 \rangle$ is the expectation value of Z_1 when measured in the final state. Z_1 can also replaced by some other operators when necessary. Thus a strong simulation algorithm achieves an exponential accuracy $\epsilon = 2^{-\text{poly}(n)}$ in $\text{poly}(n)$ time.

- **Weak simulation**: We say that $\mathcal{C}_n$ can be efficiently simulated classically in the weak sense if there exists a classical algorithm with runtime $\text{poly}(n, \frac{1}{\epsilon})$ which outputs a number E satisfying (3.1). Thus a weak simulation algorithm achieves *polynomial* accuracy $\epsilon = 1/\text{poly}(n)$ in polynomial time. We will often allow weak simulations to fail with an exponentially small probability. In this sense, we say that $\mathcal{C}_n$ can be efficiently simulated classically in the weak sense if there exists a probabilistic classical algorithm with runtime $\text{poly}(n, \frac{1}{\epsilon}, \log \frac{1}{1-p})$ which outputs a number E satisfying (3.1) with probability p. Thus for

polynomial accuracies and for success probabilities which are exponentially (in n) close to 1, the classical simulation runs in poly(n) time.

Note that in this thesis, when we say some circuit can be simulated, we always means it can be simulated efficiently on a classical computer.

As suggested by their names, strong simulation is much harder to achieve then weak simulation. To this aspect, we first prove the following theorem.

Theorem 3.1.1. *If we can simulate quantum circuits with $H, S, CNOT$ strongly, we can solve all $\#\boldsymbol{P}$ problems efficiently.*

Proof. We will show that we can encode **#P-Complete** problem (2.7) into quantum circuits. Note that $\{H, S, S^7, CNOT\}$ satisfies condition of the Solovay-Kitaev algorithm, since $S^8 = 1$. So by results in the end of section 1.2, we have for any $f_n : \{0,1\}^n \to \{0,1\}$ that can be computed efficiently on a classical computer, there exists a poly$(n, log^4\epsilon_1)$ size quantum circuit with $\{H, S, S^7, CNOT\}$ that realizes the following unitary transformation with error ϵ_1:

$$|x\rangle|y\rangle \to |x\rangle|y + f_n(x)\rangle, \tag{3.2}$$

where x is an n-bit string and y is a single bit. We can apply this quantum circuit to the initial state $\sum_{x\in\{0,1\}^n} |x\rangle|0\rangle$, resulting in

$$\sum_{x\in\{0,1\}^n} |x\rangle|f_n(x)\rangle, \tag{3.3}$$

where the sum is over all binary strings. For a measurement on the last qubit with Z operator, let the probability of getting state $|1\rangle$ to be p. By theorem 1.2.2, we know that the probability p of the approximated quantum circuit satisfies $|p - \frac{1}{2^n}\#(f_n(x) = 1)| < 2\epsilon_1$, where $\#(f_n(x) = 1)$ is the number of x such that $f_n(x) = 1$. By definition, if we can achieve strong simulation, we can also calculate p with error $\epsilon_2 = 2^{-n-3}$ in polynomial time. By choosing $\epsilon_1 = 2^{-n-4}$, we can then calculate $\frac{1}{2^n}\#(f_n(x) = 1)$ with error $\epsilon_2 = 2^{-n-2}$. With this error, we can determine $\#(f_n(x) = 1)$ exactly, since it is an integer. □

This theorem indicates that for a circuit that can be simulated strongly, we can compute the expectation value faster on a classical computer then using the circuit itself. Intuitively, strong simulation is more powerful then quantum circuit itself because sampling can not guarantee a exponential accuracy. Here is an example:

Example 3.1.2. *Let X be the random variable associated with a measurement, which satisfies $p(X = 1) = e^{-n}$ and $p(X = 0) = 1 - e^{-n}$. Thus $E(X) = e^{-n}$. We can sample from X to estimate $E(X)$ with the formula*

$$\overline{X} = \frac{1}{k}\sum_{1\le j\le k} X_j \tag{3.4}$$

If we want the error to be at most $\frac{1}{2}e^{-n}$, we have to sample exponentially many times in n to make the denominator in above equation large enough. Therefore we need to run the circuit exponentially many times. However, if the circuit can be simulated strongly, then we can calculate $E(X)$ with exponentially small error in poly(n) time.

On the other hand, weak simulation of a quantum circuit yields the same accuracies as running the circuit poly(n) times. To prove this, we need the Chernoff-Hoeffding bound, which is a tool to bound how accurately the expectation value of a random variable may be approximated using of "sample averages". Let $X_1, \ldots X_K$ be i.i.d. real-valued random variables with $E := \mathbb{E}X_i$ and $X_i \in [-1, 1]$ for every $i = 1, \ldots, K$. Then the Chernoff-Hoeffding bound asserts that

$$\text{Prob}\left\{\left|\frac{1}{K}\sum_{i=1}^{K} X_i - E\right| \leq \epsilon\right\} \geq 1 - 2e^{-\frac{K\epsilon^2}{4}}. \tag{3.5}$$

For complex-valued X_i a similar bound can be obtained for $|X_i| \leq 1$. Now consider an n-qubit quantum circuit family $\mathcal{C}_n$ followed by measurement of Z_1. Suppose that the circuit runs K times, yielding an outcome $z_i \in \{1, -1\}$ in each run. Using equation (3.5) one shows that the number $E := [\sum z_i]/K$, where the sum is over all $i = 1 \cdots K$, satisfies $|E - \langle Z_1\rangle| \leq \epsilon$ with probability $p \geq 1 - 2e^{-K\epsilon^2/4}$. Consequently, for any $\epsilon = 1/\text{poly}(n)$ there exists a suitable $K = \text{poly}(n)$ such that $|E - \langle Z_1\rangle| \leq \epsilon$ holds with probability p exponentially close to 1. In other words, the above procedure allows to achieve a polynomial approximation of $\langle Z_1\rangle$ in polynomial time with exponentially small probability of failure. This performance of the quantum computation corresponds precisely to the performance required of weak classical simulations.

In summary, the strong simulation seems to have an overly strong requirement on accuracy, especially compared to the weak simulation. However, for most classes of circuits that we know how to simulate classically, we can simulate them strongly, as we will see in the following sections. Additionally, it is very difficult to prove certain class of circuits is hard to simulate weakly, because it is very close to proving **P≠BQP**. For this purpose strong simulation is often a nice starting point.

3.2 Examples of classical simulation

3.2.1 Quantum computation with "little entanglement" can be simulated strongly

In this subsection, we will follow the paper by Jozsa and Linden [10] and show that if during each step of a quantum circuit, the entanglement is always "limited", then we can simulate the circuit strongly. One thing should be kept in mind is that entanglement measures are far from unique. Here when we say the entanglement is limited, we mean that at each step of the quantum circuit the state is p-blocked, which is defined as follows:

Definition 3.2.1. *Let $|\psi\rangle$ be a state of m qubits where the qubits are labelled by $B = \{1, 2, \ldots, m\}$. $|\psi\rangle$ is p-blocked if B can be partitioned in to subsets of size at most p:*

$$B = B_1 \cup B_2 \cup \ldots \cup B_K \quad |B_j| \leq p \tag{3.6}$$

and $|\psi\rangle$ is a product state for this partition:

$$|\psi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \cdots |\psi\rangle_K \tag{3.7}$$

where $|\psi\rangle_j$ is a state of qubits in B_j only.

Needless to say, a p-blocked states has some constraint on the entanglement it can have (e.g., a 1-blocked state is simply a product state). Now we can formulate the theorem as following:

Theorem 3.2.2. *For a family of polynomial size quantum circuits with 2-local gates and product states as input, if we have the promise that at each step of a circuit, the state is p-blocked, then the final state can be calculated efficiently. Here p is treated as a constant.*

We note that in this theorem, it is not required that the gates are chosen from a finite set, as long as they can be described efficiently.

Proof. The basic idea of this proof is to update the state each time a gate is applied. This is possible because in contrast to a general quantum state, a p-blocked state can be stored on a classical computer efficiently. The update procedure can be done as following. Suppose before gate U, the state can be written as

$$|\psi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \cdots |\psi\rangle_K. \tag{3.8}$$

Since U is a 2-local gate, it can act on at most 2 blocks of qubits. If U only acts on one block $|\psi\rangle_j$, then we only need to update $|\psi\rangle_j$ to $U|\psi\rangle_j$, which can be done efficiently. Now assume U acts on 2 blocks, say $|\psi\rangle_j$ and $|\psi\rangle_l$. We can first calculate $|\alpha\rangle = U|\psi\rangle_j|\psi\rangle_l$. By the promise, we know either $|\alpha\rangle$ contains not more then p qubits, or $|\alpha\rangle$ can be written as a tensor product of two states, which both contain not more then p qubits. Since p is a constant, we can find the tensor product by brute-force search without worrying about efficiency: we can go through every way of dividing $|\alpha\rangle$ into two blocks that each has not more then p qubits, and check whether $|\alpha\rangle$ is a tensor product with respect to the partition. So we have shown that each update can be done efficiently, and thus we are able to calculate the final state. □

An immediate corollary is we can do strong simulation with circuits of this kind. We note that the theorem can be generalized to the case where the state is only approximately p-blocked (see [10, 11]). This means the gates do not need to be described exactly in order for the circuits to be simulated.

3.2.2 "Little entanglement" is enough for quantum computation

Although we have just proved that quantum speed-up cannot achieved by p-blocked states, we shall still be cautious to make any assertion about the role of entanglement in quantum computation. A recent paper [31] has shown that universal quantum computation can be done with very little entanglement, judged by several entanglement measures. Here we will prove the result for the Von Neumann entropy.

Definition 3.2.3. *Suppose $|\psi\rangle$ is a n-qubit pure state. For a bipartition of n qubits into $\{A, B\}$, the corresponding Von Neumann entropy is*

$$S(\rho_A) = tr(\rho_A \log_2 \rho_A), \tag{3.9}$$

where $\rho_A = tr_B(|\psi\rangle\langle\psi|)$. Von Neumann entropy is a measure of entanglement for pure state in the sense that

- *it equals to zero only when the system is tensor product of subsystem A and B,*
- *it is invariant under local unitary operators, that is, operators of system A or operators of system B.*

Some preliminaries are needed before the proof.

Definition 3.2.4. *The **trace distance** between quantum states ρ and σ is*

$$D(\rho,\sigma) \equiv \frac{1}{2} tr|\rho-\sigma|, \tag{3.10}$$

where $|A| \equiv \sqrt{A^\dagger A}$.

Trace distance has a very nice property that any physical operation cannot increase the distance between two states. Especially, we have

$$D(tr_A\rho, tr_A\sigma) \leq D(\rho,\sigma), \tag{3.11}$$

which says partial trace cannot increase distance between two states. The following theorem [32] about continuity of Von Neumann entropy will also be used in our proof:

Theorem 3.2.5. *Let $D = D(\rho,\sigma)$, where ρ and σ are density matrix of dimension d. Then as long as $D \leq 1/(2e)$, one has*

$$|S(\rho) - S(\sigma)| \leq 2D\log_2(d) - 2D\log_2(2D). \tag{3.12}$$

Now we can state and prove the main theorem:

Theorem 3.2.6. *Consider a set of numbers $\{\delta_n\}$ that decrease polynomially, say $\delta_n = 1/poly(n)$. Then it is always possible to efficiently solve every problem in **BQP** even when, throughout the entire computation, the Von Neumann entropy of every n-qubit state is at most $O(\delta_n)$ for every bipartition.*

Proof. The key idea of this proof is to show we can use a small neighbourhood of state $|0\rangle^{\otimes n}$ to simulate normal quantum circuits. More precisely, for a quantum circuits C with $n-1$ qubits, we can start with

$$|0\rangle^{\otimes(n-1)}(\sqrt{1-\epsilon}|0\rangle + \sqrt{\epsilon}|1\rangle), \tag{3.13}$$

where $\epsilon > 0$ is some small constant. Then we can apply every gate in C on the first $n-1$ qubits controlled on the last qubit being in state $|1\rangle$. Hence the resulting circuit consists of gates acting on at most three qubits. Using C_t to denote the product of the first t gates in C, we have after t gates, the state becomes:

$$|\psi_t\rangle = \sqrt{1-\epsilon}|0\rangle^{\otimes(n-1)}|0\rangle + \sqrt{\epsilon}C_t|0\rangle^{\otimes(n-1)}|1\rangle \tag{3.14}$$

After all m gates have been applied, a standard basis measurement on the first qubit is performed. The probability of it being 1 is $q = \epsilon p$, where p is the probability of getting 1 from the final state $C_m|0\rangle$ of the original circuit. Repeating the computation poly(n) times allows to estimate q with accuracy $1/\text{poly}(n)$. So it suffice to run the circuit $1/\epsilon^2$ times to obtain a estimation of p with polynomial accuracy.

It remains to show that each $|\psi_t\rangle$ has a small Von Neumann entropy. Consider an arbitrary bipartite split (A,B) of the system. Let $\rho_t = |\psi_t\rangle\langle\psi_t|$ and $\sigma = |0\rangle^{\otimes n}\langle 0|^{\otimes n}$. By calculation, we have $D(\rho_t,\sigma) = \sqrt{\epsilon}$. Denote ρ_t^A and σ^A as the state obtained from ρ_t and σ by tracing out all qubits in B. Using inequality (3.11), we have $D(\rho_t^A,\sigma^A) \leq \sqrt{\epsilon}$. We can always set $\epsilon \leq (2e)^{-2}$ to satisfy the condition of inequality (3.12), then we have

$$|S(\rho_t^A) - S(\sigma_t^A)| \leq 2D|A| - 2D\log_2(2D), \tag{3.15}$$

where $D = D(\rho_t^A, \sigma^A)$ and $|A|$ denotes the number of qubits in A. Since $S(\sigma_t^A) = 0$, the equation above turns into

$$S(\sigma_t^A) \leq 2\sqrt{\epsilon}|A| - 2\sqrt{\epsilon}\log_2(2\sqrt{\epsilon}). \tag{3.16}$$

So for any $\{\delta_n\}$ that decrease polynomially with n, we can find some $\{\epsilon_n\}$ that also decrease polynomially to ensure $S(\sigma_t^A) < \delta_n$, while at the same time we only need to run the circuit poly(n) times to simulate the circuit C. □

3.2.3 Gottesman-Knill theorem

Gottesman-Knill theorem [12] states that, for a standard quantum circuit (see definition 1.1.1), if we only use gates from $\{H, CNOT, S^2\}$, then we can simulate the measurement outcome strongly. Since with these gates, we can generate states that are highly entangled, the proof in section 3.2.1 will no longer work. To prove Gottesman-Knill theorem, we need exploit properties of Pauli group and Clifford operation.

A Pauli operator on n qubits has the form $P = \alpha P_1 \otimes \ldots \otimes P_n$, where $\alpha \in \{\pm 1, \pm i\}$ and where each P_j is one of the Pauli matrices X, Y, Z or the identity. An n-qubit operator U is a Clifford operation if $UPU^\dagger$ is a Pauli operator for every Pauli operator P. For example, every Pauli operator Q is a Clifford operation, since for any Pauli operator P, we have

$$QPQ^\dagger = \pm P, \tag{3.17}$$

where the sign depends on whether Q and P commutes or anti-commutes. We can also verify CNOT gate is a Clifford operation. For U being a CNOT gate acting on qubit 1 and 2, where qubit 1 is the control qubit, we have

$$UX_1U^\dagger = X_1X_2 \tag{3.18}$$

$$UX_2U^\dagger = X_2 \tag{3.19}$$

$$UZ_1U^\dagger = Z_1 \tag{3.20}$$

$$UZ_2U^\dagger = Z_1Z_2 \tag{3.21}$$

where X_j and Z_j are the X and Z operator on qubit j. And by the fact any Pauli operator on two qubits can be written as a product of X_1, X_2, Z_1, Z_2, we know that CNOT gate is a Clifford operation.

The set of all n-qubit Clifford operations is a group, called the Clifford group. A Clifford circuit is a quantum circuit composed of H, $CNOT$ and $P = S^2 = \mathrm{diag}(1, i)$ and Pauli matrices. It is known that every Clifford circuit realizes a Clifford operator, and for every Clifford operator, we can find a (polynomial-size) Clifford circuit realizing the Clifford operator efficiently [33] .

Now we can show how to calculate the expectation value of a Z measurement after a Clifford circuit C

$$\langle\alpha|C^\dagger ZC|\alpha\rangle, \tag{3.22}$$

where $|\alpha\rangle$ is any product state. C can be split into two part $C = C_1UC_2$, for some gate U in C. Since C_1 is again a Clifford operator, we know that $C_1^\dagger ZC_1$ is some Pauli operator P. Assuming we can calculate P efficiently, it is easy to see we can also efficiently calculate

$$(C_1U)^\dagger ZC_1U = U^\dagger PU. \tag{3.23}$$

By induction, we can calculate the Pauli operator $C^\dagger ZC$ efficiently. Thereby we can calculate the expectation value (3.22).

3.2.4 Strong simulation of nearest neighbour (n.n.) Matchgate circuits

n.n. Matchgate circuits is another class of circuits that can be strongly simulated, yet they can generate non-trivial entanglement. Matchgates are 2-local quantum gates of the following form

$$G(A,B) = \begin{bmatrix} a & 0 & 0 & b \\ 0 & e & f & 0 \\ 0 & g & h & 0 \\ c & 0 & 0 & d \end{bmatrix} \tag{3.24}$$

where

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \tag{3.25}$$

are both in $SU(2)$. Thus A only affect the even subspace (i.e., the subspace spanned by $|00\rangle$ and $|11\rangle$), and B only affect the odd subspace. We have the following theorem [13]

Theorem 3.2.7. *Consider any poly-sized quantum circuit family comprising only $G(A,B)$ gates such that*

1. *all $G(A,B)$ gates only act on nearest neighbour qubits;*
2. *the input state is any product state;*
3. *the output is a measurement in the computational basis on the first qubit.*

Then we can simulate the output strongly.

Here we follow the proof in [15]. First we need the formalism of Clifford algebras.

Definition 3.2.8. *A **Clifford algebra** C_{2n} has a set of $2n$ generators $\{c_j\}$. Each c_j is a hermitian operator, and they satisfy*

$$\{c_j, c_k\} \equiv c_j c_k + c_k c_j = 2\delta_{j,k} I \quad j,k = 1,\ldots,2n \tag{3.26}$$

We also define a **quadratic Hamiltonian** H to be an element of C_{2n} that has the form

$$H = i \sum_{1 \le j \ne k \le 2n} h_{jk} c_j c_k. \tag{3.27}$$

With the requirement that $H = H^\dagger$, we can assume the matrix $h = \{h_{jk}\}$ to be a real antisymmetric matrix. The following property is essential in the proof:

Theorem 3.2.9. *Let H be a quadratic Hamiltonian defined as above, we have*

$$e^{-iH} c_j e^{iH} = \sum_{k=1}^{2n} R_{jk} c_k = R\vec{c}, \tag{3.28}$$

where matrix $R = e^{4h}$ and $\vec{c} = [c_1 \ c_2 \ \cdots c_{2n}]^\top$.

Proof. We set $c_j(t) = e^{iHt}c_j(0)e^{-iHt}$, where $c_j(0) = c_j$. Then $c_j(t)$ satisfies

$$\frac{dc_j(t)}{dt} = i[H, c_j(t)]. \tag{3.29}$$

By equation (3.26), we have $[c_{j_1}c_{j_2}, c_k] = 0$ if $k \neq j_1, j_2$ and $[c_jc_k, c_j] = -2c_j$. So

$$\frac{dc_j(t)}{dt} = \sum_k 4h_{jk}c_k(t), \tag{3.30}$$

and we know the solution of the above differential equation is

$$c_j(t) = \sum_k R_{jk}(t)c_k(0), \tag{3.31}$$

where $R = e^{4ht}$. Since h is an antisymmetric matrix, we know that R is in $SO(n)$. The proof is finished by setting $t = 1$. □

Moreover, we can deduce that

$$e^{-iH_2}e^{-iH_1}c_je^{iH_1}e^{2iH_2} = R^{(1)}R^{(2)}\vec{c}, \tag{3.32}$$

where $R^{(j)}$ is the matrix corresponding to H_j. The matrix multiplication can be done efficiently because the dimension of R is only $2n$.

Now we need a way to connect Clifford algebra to operators of qubits. This can be done via the Jordan-Wigner representation:

$$\begin{array}{llll} c_1 = XI\ldots I & c_3 = ZXI\ldots I & \cdots & c_{2k-1} = Z\ldots ZXI\ldots I \quad \cdots \\ c_2 = YI\ldots I & c_4 = ZYI\ldots I & \cdots & c_{2k} = Z\ldots ZYI\ldots I \quad \cdots \end{array} \tag{3.33}$$

where X and Y are in the kth slot for c_{2k-1} and c_{2k}, and k ranges from 1 to n. It is straightforward to check these matrices satisfy the relations (3.26). With all these tools we can prove the main theorem 3.2.7.

Proof. First we show that in the Jordan-Wigner representation (3.33), quadratic Hamiltonian can implement all n.n. Matchgates. Consider just qubit lines 1 and 2 and corresponding quadratic Hamiltonians which involve 6 possible terms:

$$-ic_1c_2 = ZI \quad -ic_2c_3 = XX \tag{3.34}$$

$$ic_1c_3 = YX \quad -ic_2c_4 = XY \tag{3.35}$$

$$ic_1c_4 = YY \quad -ic_3c_4 = IZ \tag{3.36}$$

$$\tag{3.37}$$

We want to show with quadratic Hamiltonians, we can generate all unitary operators in even(odd) subspace. Notice that $\frac{1}{2}(XX+YY)$ and $\frac{1}{2}(ZI-IZ)$ are the X and Z operator for odd subspace (and map even subspace to 0). By the fact $e^{i\theta_1 X}e^{i\theta_2 Z}e^{i\theta_3 X}$ can generate all one qubit unitary operators with determinant 1 [7], we know that we can generate $SU(2)$ on odd subspace with $e^{iH_1}e^{iH_2}e^{iH_3}$. Here each H_j is a quadratic Hamiltonian. Similarly, we can generate all $SU(2)$ on even subspace. Hence we get precisely the $G(A, B)$ gates for lines 1 and 2 as a product of $U = e^{iH}$ with the quadratic Hamiltonian restricted to use of c_1, c_2, c_3, c_4 only. Similarly for any

pair of consecutive lines we can get all $G(A, B)$, since all the Z operators in in equation (3.33) that do not belong to the two lines cancelled in product $c_j c_k$.

For the quantum circuits described in theorem 3.2.7, the expectation value of measurement outcome is

$$\langle\psi|C^\dagger ZC|\psi\rangle, \tag{3.38}$$

where C is a n.n. matchgate circuit and $|\psi\rangle$ is a product state. By the argument we had above, we know C can be written as

$$C = \prod_{1\le j\le m} e^{iH_j} \equiv e^{iH_1}\cdots e^{iH_m}, \tag{3.39}$$

where H_j are quadratic. Since the Z operator on the first qubit $Z_1 = -ic_1c_2$, equation (3.38) can be rewritten as

$$-i\langle\psi|C^\dagger c_1 CC^\dagger c_2 C|\psi\rangle. \tag{3.40}$$

$C^\dagger c_1 C$ is a linear sum of c_j, which we can calculate efficiently using equation (3.32). The same is true for $C^\dagger c_2 C$. It is then straightforward to calculate the expectation value. □

3.3 Tensor network techniques

In this section, we will follow the argument in [34], and show that quantum circuits with certain topology can be simulated in the strong sense by using tensor network formalism. For a two-qubit unitary operator U, we define its tensor U^{cd}_{ab} by

$$U^{cd}_{ab} = \langle a|\langle b|U|c\rangle|d\rangle, \tag{3.41}$$

where $a, b, c, d \in \{0, 1\}$. To see that this is indeed a tensor structure, first we calculate $(VU)^{cd}_{ab}$ as an example, where V and U act on the same two qubits. We have

$$(VU)^{cd}_{ab} = \sum_{ef} U^{ef}_{ab} V^{cd}_{ef}, \tag{3.42}$$

where the sum is over all possible combination of 0 and 1. For two gates only share a single qubit (e.g., one of the input qubit of V is the output of U), it can be shown in a similarly way that they can be combined into a single gate by contracting the shared qubit. And to compute the expectation of the measurement outcome for circuit C, we only need to contract the tensor network

$$\langle 0\cdots 0|C^\dagger ZC|0\cdots 0\rangle, \tag{3.43}$$

where we have assumed that the measurement is the Z operator on the first qubit. However, note that when we combine more and more gates into a single one, the combined gate may have a large number of input and output qubits. Thus the storage and computation of these large gates will not be efficient, since the number of components of the tensors will grow exponentially with the number of qubits they act on.

So in order for the simulation to be efficient with this method, we need a way to contract the tensor network while keeping the number of each tensor's index small, or equivalent the combined gate not acting on too many qubits. This cannot always be achieved for all kind of circuits, since some of them are intrinsically hard to contract. To quantify this hardness, we need the concept of treewidth in graph theory. It is a measure of how close a certain graph is to a tree, and intuitively, a tree can be contracted efficiently without the issue we have discussed above. We first need to introduce the tree decomposition.

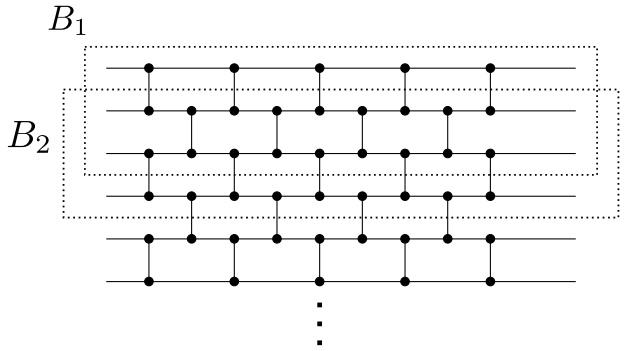

Figure 3.1: The way we arrange gates of a nearest neighbour circuit of constant depth into bags B_j. Here we only illustrate B_1 and B_2. A gate is in $B_1(B_2)$ if the whole gate is inside the square of $B_1(B_2)$.

Definition 3.3.1. *For a graph G, a tree decomposition is a tree Γ, together with a function that maps each vertex $w \in V(\Gamma)$ to a subset $B_w \subset V(G)$. These subsets B_w are called bags, with the following requirement:*

- *Each vertex must appear in at least one bag.*
- *For each edge, at least one bag must contain both of its end vertices.*
- *all bags containing a given vertex must be connected in Γ.*

The width of such a decomposition is defined by $\max_{w\in V(\Gamma)} |B_w| - 1$. The treewidth $\mathrm{tw}(G)$ is the minimum width over its tree decompositions. Then we have the following theorem [34]

Theorem 3.3.2. *Let C be a quantum circuit with T gates, and G_C be the underlying graph of tensor network (3.43). Then C can be simulated strongly in time $T^{O(1)} \exp[O(tw(G_C))]$.*

As an example, we will use this method to calculate $\langle 0\cdots 0|C_n|0\cdots 0\rangle$, where C_n are constant-depth circuits with nearest neighbour gates. Constant depth circuits are circuits which can be implemented in a constant time if any amount of gates which do not act on a common qubit can be implemented simultaneously. For example, in figure 3.1 the depth of the circuit is 9. Now Let $U^{(k)}_{j,j+1}$ being all the gates acting on qubit j or qubit $j+1$. Since C_n have some constant depth d, we know that the number of gates acting on each pair of qubit j and $j+1$ is smaller then $2d$. Then one tree decomposition of the underlying graph G_C of circuit C is by setting the bags B_j to be

$$B_j = \{U^{(k)}_{j,j+1}\} \quad j = 1, \ldots, n-1 \tag{3.44}$$

where k is taken over all possible values (see figure 3.1). And the tree Γ is constructed by connecting B_j to B_{j+1} for all j. It is easy to verify that this tree decomposition satisfies all requirement. So we have $\mathrm{tw}(G_C) < 2d$, which means it is bounded by a constant. Thus by theorem 3.3.2, we know that the contraction can be done efficiently

3.4 Weak simulation

In this section we will see, following [16], a very large class of quantum circuits can be simulated weakly. We start with an example. Recall in section 3.1.1 we have shown the circuits

$$|0\rangle|0\rangle \to \sum_{x\in\{0,1\}^n} |x\rangle|f_n(x)\rangle \tag{3.45}$$

are unlikely to be simulated in the strong sense (the measurement is on the last qubit). However, this type of circuits can be simulated in the weak sense via sampling. In more detail, the probability of last qubit being $|1\rangle$ is

$$p = \frac{1}{2^n}\#(f_n(x) = 1), \tag{3.46}$$

where $\#(f_n(x) = 1)$ is the number of x such that $f_n(x) = 1$. We can take $x_1, \cdots, x_K$ from the uniform distribution of $\{0,1\}^n$, and calculate

$$p' = \frac{1}{K}\sum_{1\le j\le K} f(x_j) \tag{3.47}$$

as an approximation of the above probability (3.46). By Chernoff-Hoeffding bound, the error $|p - p'|$ is of order $1/\text{poly}(K)$, which satisfies the weak simulation requirement.

To generalize this sampling technique to simulate more circuits, we need to introduce the following concepts.

3.4.1 CT states

Consider a family of n-qubit states $|\psi_n\rangle \equiv |\psi\rangle$ specified in terms of some classical description, say a quantum circuit preparing $|\psi\rangle$ from the state $|0\rangle$. $|\psi\rangle$ is said to be *computationally tractable* (CT) if

(a) it is possible to sample in poly(n) time with classical means from the probability distribution $\text{Prob}(x) = |\langle x|\psi\rangle|^2$ on the set of n-bit strings x, and

(b) for any bit string x, the coefficient $\langle x|\psi\rangle$ can be computed in poly(n) time on a classical computer with exponential precision.

Here are some examples of computationally tractable states:

- Product states are CT
- For a poly-size uniform family of Clifford circuits $\{C_n\}$, the family of states $\{C_n|0\rangle^{\otimes n}\}$ is CT. The two properties can be deduced from [12, 33].
- States obtained by applying a poly-size uniform family of n.n. Matchgate circuit to a computation basis state are CT. Both properties can be found in [13]

We call U a monomial unitary operator if U has only one non-zero element per row and per column. In other words, U always map a computational basis $|x\rangle$ to $\gamma(x)|\pi(x)\rangle$, where $\gamma(x)$ is a phase and $\pi(x)$ is a permutation of x. CT states have the following property:

Theorem 3.4.1. *For a monomial unitary operator U, if $\gamma(x)$, $\pi(x)$ and $\pi^{-1}(x)$ can be computed efficiently, then for any CT state $|\psi\rangle$, $U|\psi\rangle$ is also a CT state.*

Proof. It is straightforward to check $U|\psi\rangle$ satisfies both properties of CT states. For property (a), since the phase $\gamma(x)$ does not affect the probability of getting $|\pi(x)\rangle$, we can just sample from the CT state $|\psi\rangle$ to get x and then output $\pi(x)$. For property (b), to calculate the coefficient of $|x\rangle$ in $U|\psi\rangle$, we can first calculate the coefficient of $|\pi^{-1}(x)\rangle$ in state $|\psi\rangle$, and multiply it by $\gamma(\pi^{-1}(x))$. All these calculation can be done efficiently. □

We can also calculate the overlap between two CT states.

Theorem 3.4.2. *Let $|\psi\rangle$ and $|\varphi\rangle$ be two CT n-qubit states. Then there exists an efficient classical algorithm to approximate $\langle\varphi|\psi\rangle$ with polynomial accuracy.*

In other words, we can simulate $\langle\varphi|\psi\rangle$ weakly. The proof is based on the use of Chernoff-Hoeffding bound (3.5).

Proof. Denote $p_x = |\langle x|\psi\rangle|^2$ and $q_x = |\langle x|\varphi\rangle|^2$. Since $|\psi\rangle$ and $|\varphi\rangle$ are CT states, p_x and q_x can be computed efficiently. Define the function $\delta : \{0,1\}^n \to \{0,1\}$ by

$$\delta(x) = \begin{cases} 1 & \text{if } p_x > q(x), \\ 0 & \text{otherwise.} \end{cases} \tag{3.48}$$

We can then write $\langle\varphi|\psi\rangle$ as

$$\begin{aligned} &\sum_x \langle\varphi|x\rangle\langle x|\varphi\rangle \\ =&\sum_x \langle\varphi|x\rangle\langle x|\varphi\rangle\delta(x) + \sum_x \langle\varphi|x\rangle\langle x|\varphi\rangle(1-\delta(x)), \end{aligned} \tag{3.49}$$

where the sums are taken over all n-bit strings x. Defining the functions F and G by

$$F(x) = \frac{\langle\varphi|x\rangle\langle x|\varphi\rangle}{p_x}\delta(x), \quad G(x) = \frac{\langle\varphi|x\rangle\langle x|\varphi\rangle}{q_x}(1-\delta(x)), \tag{3.50}$$

we have $\langle\varphi|\psi\rangle = \langle F\rangle + \langle G\rangle$, where $\langle F\rangle = \sum_x p_x F(x)$ and $\langle G\rangle = \sum_x q_x G(x)$. Since $|\psi\rangle$ is CT, there is a way to sample from the probability distribution $\{p_x\}$. By the fact that $|F(x)| \leq 1$, we know from Chernoff-Hoeffding bound that we can approximate $\langle F\rangle$ efficiently with polynomial accuracy. Similarly we can obtain $\langle G\rangle$ with polynomial accuracy, which finishes the proof. □

Now we can revisit the circuit (3.45), and use this new framework to show it can be simulated in the weak sense. First, we notice that the state

$$\frac{1}{2^n}\sum_x |x\rangle|0\rangle \tag{3.51}$$

is a CT state, where the sum is taken over all n bit string. Since the operation

$$|x\rangle|y\rangle \to |x\rangle|y + f(x)\rangle \tag{3.52}$$

is a monomial operator which can be calculated efficiently. By theorem 3.4.1, we know the final state $|\psi\rangle = \sum_{x\in\{0,1\}^n} |x\rangle|f_n(x)\rangle$ of circuit (3.45) is a CT state. To finish the proof, we notice that by theorem 3.4.2 we can calculate the expectation value $\langle\psi|Z|\psi\rangle$ with a polynomial accuracy, since $Z|\psi\rangle$ is again a CT state.

3.4.2 Sparse operations

In this section we will show a generalized version of theorem 3.4.2. First we need to define a class of sparse operations that can be efficiently computed.

Definition 3.4.3. *A family of operations A_n on n qubits is called efficiently computable sparse (ECS) if*

- *A_n is s_n sparse, where s_n =poly(n). In other word, A_n has at most s_n non-zero elements per row and per column.*
- *A_n is efficiently computable. When given a row number or column number of A_n, it is efficient to list all positions of non-zero numbers in that row (column) and compute their value to exponential accuracy.*

The following are some examples of efficiently computable sparse operations (ECS).

- For $d = O(\log n)$, a d-qubit operator G acting on n qubits is ECS.
- An operator that can be written as a sum of poly(n) ECS operators, is ECS
- Any Pauli operator P is ECS. For any Clifford circuit C, $C^\dagger PC$ is again a Pauli operator. Thus $C^\dagger PC$ is also ECS.
- By equation (3.28) and (3.32), we know that for a Matchgate circuit C, $C^\dagger PC$ is ECS.

We have the following theorem

Theorem 3.4.4. *If A_n is ECS operators, and $|\psi\rangle, |\varphi\rangle$ are both CT states, then we can compute*

$$\langle\psi|A_n|\varphi\rangle \tag{3.53}$$

with polynomial accuracy.

The proof is very similar to the proof of theorem 3.4.2, so we will not give the proof here (see [16] for the proof). The strength of theorem is allowing you to see immediately that circuits with certain structure can be simulate weakly. We give two examples here. The initial state is $|0\rangle$, and the measurement is the Z operator on the first qubit for both case.

- Consider a circuit (family) $C = C_1C_2$, with C_1 and C_2 being a constant-depth circuit and a Clifford circuit respectively. Since $C_1^\dagger ZC_1$ only acts on a constant number of qubit, it is ECS. And we know that $C_2|0\rangle$ is CT. By theorem 3.53, we can calculate $\langle 0|C_2^\dagger C_1^\dagger ZC_1C_2|0\rangle$ with a polynomial accuracy.
- Again consider $C = C_1C_2$. This time we set C_1 to be a Clifford circuit and C_2 a Matchgate circuit. By noticing that $C_2|0\rangle$ is CT and $C_1^\dagger ZC_1$ is ECS, we conclude that we can simulate C weakly.

Chapter 4

Commuting circuits

In this section we will study the power of commuting quantum circuits. The contents starting from section 4.3 are based on our work [35].

Now our main results can be summarized as follows:

- *2-local circuits are easy.* All uniform families of commuting circuits consisting of 2-local gates acting on d-level systems and followed by a single qudit measurement can be strongly simulated by classical computation, for every d.
- *3-local circuits are hard.* Uniform families of commuting circuits consisting of 3-local gates acting on qubit systems and followed by a single qubit measurement cannot be strongly simulated by classical computation, unless every problem in #P has a polynomial-time classical algorithm.
- *Commuting Pauli circuits.* All uniform families of commuting circuits consisting of exponentiated Pauli operators $e^{i\theta P}$ and followed by a single-qubit measurement can be efficiently simulated classically weakly. Furthermore, even when such circuits display a small degree of non-commutativity, an efficient classical simulation remains possible.
- *Mapping non-commuting circuits to commuting circuits.* Certain non-commuting quantum processes (related to bounded-depth circuits) can be efficiently simulated by purely commuting quantum circuits.

4.1 Preliminary definitions

A *commuting circuit* is a quantum circuit consisting of pairwise commuting gates. For an operator A which acts on a system of n qudits labeled by $1 \cdots n$, the support of A is the subset of qudits on which it acts nontrivially. A k-local commuting circuit is in *standard form* if for every subset $S \subseteq \{1, \ldots, n\}$ consisting of k qudits there is at most one gate G_i with $\mathrm{supp}(G_i) \subseteq S$. A k-local commuting circuit $\mathcal{C} = G_m \cdots G_1$ in standard form contains at most $\binom{n}{k}$ gates, so that the size of the circuit scales polynomially with n if k is constant. For example, a two-local commuting circuit is in standard form if for every $i, j = 1 \cdots n$ with $i < j$ there is at most one gate in the circuit with support contained in $\{i, j\}$; such circuit has size $O(n^2)$. Every k-local commuting circuit can be brought into standard form by replacing all gates in the circuit with support contained in S by a single gate given by the total product of these gates, for every subset S consisting of k qudits. Furthermore if k is constant and if the original circuit

has size m then this procedure to bring a circuit in normal form can be carried out efficiently i.e. in poly(n, m) steps.

A simple example of a commuting circuit is $\mathcal{C} = G_m \cdots G_1$ with gates

$$G_i = U \otimes U D_i U^\dagger \otimes U^\dagger, \tag{4.1}$$

where U is a fixed single-qudit unitary operator (independent of i) and where each D_i is a diagonal unitary operator. In other words each gate is diagonal in the same local basis. This class of commuting circuits has been considered in [36, 19].

By their commutativity, all gates in any commuting circuit can be diagonalized simultaneously i.e. there exists a unitary operator V such that $VG_iV^\dagger$ is diagonal for every gate G_i. In the example (4.1), the diagonalizing operator is a simple tensor product $V = U \otimes \cdots \otimes U$. This example does however not represent the most general situation since V may be a global, entangling operation—even when the commuting circuit is k-local with k constant. Consider for example an n-qubit 3-local circuit with gates $G_j = e^{i\theta_j K_j}$ where the commuting operators K_j are defined as follows:

$$K_1 = X_1 Z_2, \quad K_i = Z_{i-1} X_i Z_i, \quad K_n = Z_{n-1} X_n, \quad \text{with} \quad i = 2, \ldots n-1. \tag{4.2}$$

Here Z_i and X_i denote the Pauli X and Z operators acting on qubit i. The operators K_j are known to be the stabilizers of the 1D cluster state [25]. Let H denote the Hadamard gate and let $\mathrm{CZ} = \mathrm{diag}(1, 1, 1, -1)$ denote the controlled-Z gate. It is then easily verified that the entangling operation

$$V = H^{\otimes n} \prod_{i=1}^{n-1} CZ_{i,i+1} \tag{4.3}$$

sends $K_j \to VK_jV^\dagger = Z_j$ and thus simultaneously diagonalizes the K_j. Furthermore it can be shown that no tensor product of single-qubit operations can perform such a diagonalization .

The example (4.2) shows that there exist k-local commuting circuits where the diagonalizing unitary V is a global, entangling operator. Nevertheless this example is still rather well-behaved as V can be computed efficiently and moreover has a relatively simple structure. In fact in section 4.6 we will investigate commuting circuits composed of exponentiated Pauli operators $e^{i\theta P}$ in more detail and show that such circuits have efficient classical simulations (relative to certain measurements). For general k-local commuting circuits, however, the unitary V may have a complex structure and be computationally difficult to determine. This feature is in part responsible for the complexity of commuting quantum circuits.

4.2 Previous results

Let us first look at three previous results of commuting quantum circuits [19, 36, 37]. These three papers confirm that commuting circuits can achieve certain tasks which are hard on classical computers. Noting that this does not imply we do not have a chance to simulate commuting circuits efficiently, since in [19] the requirement of accuracy is very high (for example, see the comparison of strong and weak simulation in section 3.1), and in [36] the hardness result is again based on some other conjecture.

4.2.1 Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy

In the paper [19], they show that if we can use a classical computer to sample from the output distribution of 2-local commuting circuits efficiently with certain accuracy, then the polynomial hierarchy will collapse. More precisely, for sampling here, we mean

Definition 4.2.1. *Let P_n denote the output distribution from a uniform circuit family $\{C_n\}$ when we measure all qubits in the computational basis. We say P_n can be classically sampled with multiplicative error c when there exist a family of distribution R_n such that*

- *R_n can be generated on classical computer efficiently.*
- *$\frac{1}{c}Prob[P_n(x)] \leq Prob[R_n(x)] \leq c\ Prob[P_n(x)]$ for all binary string x.*

Note that this is a quite strict requirement for sampling, in the sense that by running C_n a polynomial times, we cannot estimate $P_n(x)$ to such high accuracy in general.

On the other hand, Polynomial hierarchy **PH** is a infinite hierarchy of complexity classes that generalize the class **P**, **NP**. More precisely, **PH** contains an infinite tower of increasing classes $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \cdots$, in which $\Sigma_0 = \mathbf{P}$ and $\Sigma_1 = \mathbf{NP}$. The definition of Σ_j is beyond the scope of this thesis, but it is widely believed that all the inclusions above are strict. If this is not the case, that is, if $\Sigma_k = \Sigma_{k+1}$ for some k, then it can be shown that $\Sigma_k = \Sigma_l$ for all $l > k$. In this case, we say that **PH** collapse to Σ_k. Now we can state the main result of [19] formally:

Theorem 4.2.2. *If the output probability distributions generated by uniform families of IQP circuits could be sampled with multiplicative error $1 \leq c \leq \sqrt{2}$ then the polynomial hierarchy would collapse to its third level, i.e. **PH**$= \Sigma_3$.*

In other words, their result states that it is unlikely that we can sampling 2-local commuting circuits with this accuracy.

A key component used to prove this result is that in the presence of post-selection, commuting 2-local circuit is equivalent to the universal 2-local circuit. Post-selected quantum circuits are often used as a theoretical tool when study the power or quantum computation. In addition to the normal output qubits, a post-selected circuit also has polynomial many post-selection qubits. We only consider the runs of the circuit when the post-selection qubits are all measured in state $|0\rangle$. In other words, we assume we always get $|0\rangle$ when we measure post-selection qubits. In general, this process is not efficient, since the probability of all the post-selection qubits being $|0\rangle$ can be exponentially small. So we might have to run the circuit exponentially many times to have one successful output. But in a post-selection scenario these failed runs are not taken into account. The decision problems which can solve efficiently with post-selected quantum circuit form the class **Post-BQP**. It is coincides with the class **PP** [38], which can be shown to have the same difficulty level as #**P** in some sense. So this result implies that with the help of post-selected 2-local commuting circuits, we can solve problems in #**P** efficiently. This results can be state formally as

Lemma 4.2.3. *Let $\mathcal{U}$ be an n-qubit quantum circuit composed of the gates H, S and CZ and denote $|\psi\rangle = \mathcal{U}|0\rangle^n$. Then there exists a 2-local commuting circuit $\mathcal{C}$ on $k+n$ qubits such that $|\psi\rangle$ is obtained by postselecting $\mathcal{C}|0\rangle^{k+n}$ on the first k qubits; more precisely*

$$|0\rangle^k|\psi\rangle = \sqrt{2}^k \mathcal{P}\mathcal{C}|0\rangle^{k+n}. \tag{4.4}$$

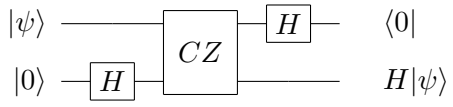

Figure 4.1: The structure to replace H gates

Here $\mathcal{P}$ denotes the projector $|0\rangle\langle 0|$ acting on the first k qubits. Furthermore $k = poly(n)$ and the description of $\mathcal{C}$ can be computed efficiently on input of the description of $\mathcal{U}$.

Since we will use this result later, we will give a proof of it.

Proof. We will show how to transform a universal 2-local quantum circuit containing $\{H, S,$ CZ$\}$ to a 2-local commuting circuit with post-selection. To do this, we first add in extra H gates to make sure each line of the (universal) circuit starts and end with H. This is possible since $H^2 = I$, and we can just add two successively H to the beginning and end of a line if necessary. Next we consider those H gates in the middle. We can replace each of them with the structure in figure 4.1. The post-selection qubit is labelled as $\langle 0|$ in the figure. It can be verified that with this structure, we can implement H gates. The success probability of the post-selection is always $\frac{1}{2}$, which is independent of the other part of the circuit.

After these two steps, the circuit only has diagonal gates except for the starting and ending H gates. Replacing each diagonal gate U in the circuit with HUH or $H^{\otimes 2}UH^{\otimes 2}$ and remove the starting and ending H gate, we get a 2-local commuting circuits with post-selection. □

We will not give the proof of their main results here, since it involves too much of computational complexity theory. However, there is a side result in their paper, which will be quite interesting to compare with our results later.

Theorem 4.2.4. *Consider commuting quantum circuits with initial state $|0\cdots 0\rangle$ and gates of the form $H^{\otimes k}DH^{\otimes k}$, where D is a diagonal gate on k qubits. Here k can be arbitrary large, but the diagonal elements of D can be computed efficiently. Let P be the output distribution of $\log n$ qubits. Then P can be sampled on a classical computer efficiently.*

Proof. This type of commuting circuits can be viewed as following: we first apply H gates on every qubit of $|0\cdots 0\rangle$, and then we apply the diagonal gates. In the end we do the H gates again and measure. Let $|x\rangle$ denote the $\log n$ qubits contained in the distribution P, and $|y\rangle$ be the other qubits. After we apply H gates in the first step, the state is

$$\frac{1}{\sqrt{2^{|x|+|y|}}}\sum_{x,y}|x,y\rangle, \tag{4.5}$$

where the summation is over all binary string x and y, and $|x|, |y|$ is the number of qubits in $|x\rangle$ and $|y\rangle$ respectively. The state becomes

$$\frac{1}{\sqrt{2^{|x|+|y|}}}\sum_{x,y}f(x,y)|x,y\rangle, \tag{4.6}$$

after we apply the diagonal gates. For each x, y, $f(x, y)$ can be computed efficiently, which is due to the fact that each diagonal gate is efficiently computable. At this point, we only need to apply H gates to every qubit of (4.6) and then we can do the measurement. Since there is no

more interaction between qubits in $|x\rangle$ and $|y\rangle$, by the principle of non-signalling, we know that any gate or measurement in $|y\rangle$ will not change the distribution of qubits in $|x\rangle$. More precisely, this is because in principle we can arrange the measurements of qubits in $|x\rangle$ and $|y\rangle$ in a way that two measurements have a space-like interval. By this fact, we can first do a measurement of qubits in $|y\rangle$ in the computational basis without affecting the distribution of qubits in $|x\rangle$. It is easy to notice that, when we do a measurement of $|y\rangle$ in state (4.6) in the computational basis, the measurement outcome is a uniform distribution of all binary string. So we can first choose y_0 from the uniform distribution. The states (4.6) then collapse to

$$\frac{1}{\sqrt{2^{|x|}}} \sum_x f(x, y_0)|x\rangle \tag{4.7}$$

where we just neglect the qubits in $|y_0\rangle$. Now it is a quantum state with only $\log n$ qubits, so we can store all coefficients of the states in a polynomial space on a classical computer, and we can simulate any subsequent gates and measurement efficiently in a trivial way. □

4.2.2 The study of class IQP

In [36, 37], the authors study the power of certain class of commuting circuits, which is called **IQP**. The circuits in **IQP** consist of gates

$$e^{i\theta X_{j_1} \otimes X_{j_2} \otimes \cdots \otimes X_{j_k}}, \tag{4.8}$$

where X_{j_l} is the X operator on qubit j_l. There is no limit of how much qubits these commuting gates can act on, i.e., they can act on all qubits of the circuit. The initial state is $|0 \cdots 0\rangle$, and the authors are interested in the output distribution of a **IQP** circuit.

In [36], the authors proposed a two-party protocol that one party (Alice) can be convinced (under some conjecture) that the other party (Bob) has **IQP** circuits rather than a classical computer. To understand this result, let us first look at the case that Bob want to convince Alice that he has a quantum computer. In this case, if Alice do not believe that factoring can be done on a classical computer efficiently, she can send Bob some large numbers and ask Bob to factor them. If Bob can send back the correct results in a reasonable time, then Alice would be convinced. Note that this protocol works because we do not have a way to simulate quantum computers efficiently. So the result of [36] would imply that we cannot simulate the **IQP** circuits under certain conjecture. We also want to note that the conjecture used in [36] might not be as convincing as "factoring is hard".

In [37], the author characterize the output distribution by using tools from combinatorics. Here we only list two of the results:

- If in the **IQP** circuit, all θ in the gates (4.8) are $\frac{\pi}{4}$, then we can sample from the output distribution efficiently on a classical computer.
- If in the **IQP** circuit all θ are equal to $\frac{\pi}{8}$, then we can compute any correlation coefficients of the output distribution. Here correlation coefficients are related to the correlation between qubits in the output distribution. Detailed definition can be found in [37].

4.3 Efficient strong simulation of one qudit

Now we will start to show our results. Our first result is we can simulate 2-local commuting circuits with single qudit measurement in the strong sense.

Theorem 4.3.1. (Strong simulations of 2-local commuting circuits) *Let $\mathcal{C}$ be a uniform family of 2-local n-qudit commuting circuits, acting on a product input state and followed by measurement of an observable O acting on qudit i for some i. Any such computation can be efficiently simulated classically in the strong sense.*

Proof. We prove the result for $i = 1$; other i are treated fully analogously. Denote the input by $|\alpha\rangle = |\alpha_1\rangle \cdots |\alpha_n\rangle$ where each $|\alpha_i\rangle$ is a single-qudit state. We can assume without loss of generality that $\mathcal{C} = \prod U_{jk}$ is in standard form, where U_{jk} represents the unique gate in the circuit with support $S \subseteq \{j, k\}$, for every $j, k = 1 \cdots n$ and $j < k$. If U_{jk} does not act on qudit 1, then this gate commutes with O. Hence in the product $\mathcal{C}^\dagger O \mathcal{C}$ we can commute U_{jk} through $\mathcal{C}$ and O to the left until it cancels out with $U_{jk}^\dagger$. By doing so, we can remove all gates that do not act on qudit 1. Therefore the expectation value of O is

$$\langle O \rangle = \langle \alpha | \mathcal{C}^\dagger O \mathcal{C} | \alpha \rangle = \langle \alpha | (\prod U_{1j})^\dagger O \prod U_{1j} | \alpha \rangle \tag{4.9}$$

where the products are over all $j \geq 2$. Now our strategy will be to trace out qudits one by one in the above equation. Denote $|\alpha^{(1)}\rangle = |\alpha\rangle$, $\mathcal{C}^{(1)} = \mathcal{C}$ and $O^{(1)} = O$. Furthermore for every $k = 2, \ldots, n-1$ define

$$\begin{aligned} |\alpha^{(k)}\rangle &= |\alpha_1\rangle |\alpha_{k+1}\rangle \cdots |\alpha_n\rangle \\ \mathcal{C}^{(k)} &= U_{1k+1} \cdots U_{1n} \\ O^{(k)} &= [I \otimes \langle \alpha_k |]\ U_{1k}^\dagger O^{(k-1)} U_{1k}\ [I \otimes |\alpha_k\rangle]. \end{aligned} \tag{4.10}$$

Remark that each $O^{(k)}$ acts on a single qudit (namely qudit 1). Furthermore each of these operators can be computed classically with exponential precision in polynomial time: $O^{(1)}$ is given as an input and each update from $O^{(j)}$ to $O^{(j+1)}$ involves simple multiplications of 2-qudit operations which can be done in constant time (taking $O(d^6)$ steps where d denotes the dimension of one qudit).

With the above definitions one finds, for every $k = 2, \ldots, n-1$:

$$\langle \alpha^{(k-1)} | [\mathcal{C}^{(k-1)}]^\dagger O^{(k-1)} \mathcal{C}^{(k-1)} | \alpha^{(k-1)} \rangle = \langle \alpha^{(k)} | [\mathcal{C}^{(k)}]^\dagger O^{(k)} \mathcal{C}^{(k)} | \alpha^{(k)} \rangle. \tag{4.11}$$

Using this equation iteratively, we get

$$\langle O \rangle = \langle \alpha^{(1)} | [\mathcal{C}^{(1)}]^\dagger O^{(1)} \mathcal{C}^{(1)} | \alpha^{(1)} \rangle = \cdots = \langle \alpha^{(n-1)} | U_{1n}^\dagger O^{(n-1)} U_{1n} | \alpha^{(n)} \rangle. \tag{4.12}$$

The last expression is easily computed since $|\alpha^{(n-1)}\rangle$ is a 2-qudit state and U_{1n} and $O^{(n-1)}$ act on at most 2 qudits. □

The above result can readily be generalized in different ways. First, using a similar argument one shows that measurement of any observable acting on $O(\log n)$ qudits can be strongly simulated as well. Furthermore, interestingly, the result also generalizes to mutually *anticommuting* gates, and more generally to gates which commute "up to a phase" as follows. Let $\mathcal{C} = \prod G_i$ be a uniform family of 2-local n-qudit circuits such that $G_i G_j = \gamma_{ij} G_j G_i$ for all pairs of gates, where the γ_{ij} are complex phases. Input and measurement are as in theorem 4.3.1. Then such circuits can be efficiently simulated classically in the strong sense. Analogous to the first step in the proof of theorem 4.3.1, the proof starts by "removing" all gates which do not act on qudit i from the product $\mathcal{C}^\dagger O \mathcal{C}$ by commuting them through the circuit. This introduces an (easily computed) product of phases γ_{ij}. The remainder of the proof of theorem 4.3.1 carries over straightforwardly.

4.4 2-local commuting circuits cannot compute all functions in P

Here we show that two-local commuting circuits are not universal for classical computation by giving an explicit example of a function which is not computable with such circuits. To prove this result, we also need the formalism of density matrix.

4.4.1 The density matrix formalism

Generally, if there is entanglement in some quantum state (i.e., it cannot be written as a product state of its subsystems), then we cannot describe the states of its subsystems by using vectors from the Hilbert space of the subsystem $\mathcal{H}^{\otimes d}$. Instead, for each subsystem, it will behave like it is a probabilistic mixture of pure states. This is what we call mixed states. The density matrix (operator) of a mixed state is defined by

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j|, \tag{4.13}$$

where p_j is the probability of the system being in state $|\psi_j\rangle$ and they summed up to 1. The general properties of density matrix can be found in [7]. To our purpose, we state the following properties without giving the proof.

Lemma 4.4.1. *For a mixed state ρ_{AB} composed of subsystem A and B, the state of A can be obtained by the partial trace $\rho_A = \mathrm{tr}_B(\rho_{AB})$, where*

$$\mathrm{tr}_B(\rho_{AB}) = \sum_j \langle\varphi_j|\rho_{AB}|\varphi_j\rangle, \tag{4.14}$$

where $|\varphi_j\rangle$ is a set of basis of system B. We also call ρ_A the reduced density matrix (operator) of system A

Lemma 4.4.2. *For some Hermitian operator O, the expectation value $\langle O\rangle$ when measured in mixed state ρ is*

$$\langle O\rangle = \mathrm{tr}(\rho O) \tag{4.15}$$

In particularly, the second Lemma here would imply that when two density matrix is very close, the measurement outcomes of them will also be close, since the trace operation is a continuous operation. Formally, we have the following theorem [7]

Theorem 4.4.3. *Recall that the trace distance between two density operators is*

$$D(\rho,\sigma) = \frac{1}{2}\,\mathrm{tr}\,\sqrt{(\rho-\sigma)^\dagger(\rho-\sigma)}. \tag{4.16}$$

For a measurement with m possible outcomes on the system of ρ and σ, denote the corresponding probability distribution by $\{p_j\}_{1\le j\le m}$ and $\{q_j\}_{1\le j\le m}$. We can also define

$$D(p_j,q_j) = \frac{1}{2}\sum_j |p_j - q_j|. \tag{4.17}$$

Then we have for any measurement,

$$D(p_j,q_j) \le D(\rho,\sigma) \tag{4.18}$$

We also need the concept of purification.

Definition 4.4.4. *The state $|\psi\rangle_{AB}$ is called a purification of the mixed state ρ_A when*

$$\rho_A = \mathrm{tr}_B(|\psi\rangle\langle\psi|_{AB}) \tag{4.19}$$

Now we can state and proving the following Lemma.

Lemma 4.4.5. *Let $\sigma_1, \ldots \sigma_N$ be a collection of $d \times d$ density operators. For any $\epsilon > 0$, if $N > \left(\frac{5}{\epsilon}\right)^{2d^2}$, then there exists two operators σ_j and σ_k such that $\|\sigma_j - \sigma_k\|_{tr} \leq \epsilon$, where $\|A\|_{tr} \equiv \frac{1}{2} tr\sqrt{A^\dagger A}$ denotes the trace distance.*

Proof. We will show for any $\epsilon > 0$, there exists a finite set $\mathbf{E}$ of $d \times d$ density operators, such that for every density operator ρ, there exists $\sigma \in \mathbf{E}$ with $\|\rho - \sigma\|_{\mathrm{tr}} < \epsilon$ (we call $\mathbf{E}$ a ϵ-net). To do this, first we recall that every density operator of dimension d has a purification by introducing an ancillary d-dimensional space R (see [7]). And in [39], it was shown that for pure states of dimension d^2, there exists a ϵ-net $\mathbf{F}$ with cardinality $|\mathbf{F}| \leq \left(\frac{5}{2\epsilon}\right)^{2d^2} \equiv M$. We can then choose set $\mathbf{E}$ to be $tr_R\mathbf{F}$, which is the partial trace of each element of set $\mathbf{F}$. Since partial trace is a contractive operation [7], i.e. $\|tr_R(\mu - \tau)\|_{\mathrm{tr}} \leq \|\mu - \tau\|_{\mathrm{tr}}$, we know that set $\mathbf{E}$ obtained this way is indeed an ϵ-net.

Note that $|\mathbf{E}| \leq |\mathbf{F}| = M$. Now if there are more then M density operators, then there must be two density operators σ_j, σ_k that are ϵ-close to the same element of $\mathbf{E}$. Thus by triangle inequality $\|\sigma_j - \sigma_k\|_{\mathrm{tr}} < 2\epsilon$. The proof can be finished by a rescaling of ϵ. □

4.4.2 The proof of the main theorem

For every d we let $\mathbf{Z}_d$ denote the set of integers modulo d. Let $\mathcal{C}$ denote a two-local commuting circuit acting on m d-level systems. Consider a function $f : \mathbf{Z}_d^k \to \mathbf{Z}_d$. We say that $\mathcal{C}$ computes f with probability at least p if the circuit $\mathcal{C}$ acing on $|x, 0\rangle$ (where 0 denotes a string of $m - k$ zeroes) and followed by a standard basis measurement of the first qudit yields the outcome $f(x)$ with probability at least p.

We will in particular consider the "inner product function" $f_{\mathrm{ip}} : \mathbf{Z}_d^{2n} \to \mathbf{Z}_d$ defined by

$$f_{\mathrm{ip}}(x^a, x^b) = (x^a)^T x^b \mod d, \tag{4.20}$$

for every $x^a, x^b \in \mathbf{Z}_d^n$.

Theorem 4.4.6. *Consider an arbitrary d and an arbitrary constant $p > 1/2$. For sufficiently large n, the inner product function f_{ip} is not computable by any two-local commuting circuit.*

Proof. Suppose there exists an m-qudit quantum circuit $\mathcal{C}$, for some $m \geq 2n$, which computes f with probability $p > 1/2$. We show that this leads to a contradiction. Repeating the argument of theorem 4.3.1 we can remove all gates from the circuit which do not act on qudit 1. We denote this simplified circuit again by $\mathcal{C}$. Now write $\mathcal{C} = \mathcal{C}_b\mathcal{C}_a$, where $\mathcal{C}_a$ consists of all gates in the circuit acting on qudits $\{1, i\}$ with $i = 1 \ldots n$ and where $\mathcal{C}_b$ consists of all gates acting on qudits $\{1, j\}$ with $i = n + 1 \ldots m$. Furthermore, let $x = (x^a, x^b)$ be an arbitrary input of f. Finally, denote

$$\sigma(x^a) := \mathrm{Tr}_{n\ldots 2}\, \mathcal{C}_a|x^a\rangle\langle x^a|\mathcal{C}_a^\dagger, \tag{4.21}$$

which is the reduced density operator for qudit 1 of the state $\mathcal{C}_a|x^a\rangle$.

The final state of the entire circuit is $\mathcal{C}|x,0\rangle$ where 0 denotes a string of $m-n$ zeroes. With the notations above, the reduced density operator of the first qudit is

$$\begin{aligned}\rho(x^a,x^b) &:= \mathrm{Tr}_{m\ldots 2}\ \mathcal{C}|x,0\rangle\langle x,0|\mathcal{C}^\dagger = \mathrm{Tr}_{m\ldots n+1}\ \mathrm{Tr}_{n\ldots 2}\ \mathcal{C}_b\mathcal{C}_a|x,0\rangle\langle x,0|\mathcal{C}_a^\dagger\mathcal{C}_b^\dagger \\ &= \mathrm{Tr}_{m\ldots n+1}\ \mathcal{C}_b\left\{\sigma(x^a)\otimes|x^b,0\rangle\langle x^b,0|\right\}\mathcal{C}_b^\dagger \qquad (4.22)\end{aligned}$$

We now use lemma 4.4.5. This implies for every $\epsilon>0$ there exists an n sufficiently large and two n-tuples $x^a\neq y^a$ such that $\|\sigma(x^a)-\sigma(y^a)\|_{\mathrm{tr}}\leq\epsilon$. Using (4.22) and the fact that the trace norm is contractive, it follows that $\|\rho(x^a,x^b)-\sigma(y^a,x^b)\|_{\mathrm{tr}}\leq\epsilon$ for *every* n-tuple x^b! This implies the following: if a standard basis measurement on $\rho(x^a,x^b)$ yield some outcome u with probability $p(u)$, then standard basis measurement on $\rho(y^a,x^b)$ will yield the same outcome with probability $q(u)$ where $|p(u)-q(u)|\leq\epsilon$. Setting $\epsilon=p-\frac{1}{2}$ and using that $\mathcal{C}$ computes f with probability at least p, it then follows that $f(x^a,x^b)=f(y^a,x^b)$ for all x^b. Using the definition of f, this straightforwardly implies that $x^a=y^a$, thus leading to a contradiction. □

4.5 3-Local commuting circuits are hard

Next we show that strong simulations of 3-local commuting circuits are unlikely to exist.

Theorem 4.5.1 (Hardness of simulating 3-local commuting circuits). *Let $\mathcal{C}$ be a uniform family of n-qubit 3-local commuting quantum circuits acting on the input $|0\rangle$ and followed by Z measurement of the first qubit. If all such circuits could be efficiently simulated classically in the strong sense then every problem in $\#P$ has a polynomial time algorithm.*

In other words, there is a drastic increase in complexity in the seemingly innocuous transition from 2-local to 3-local gates. Remark that hardness already holds for the simplest case i.e. qubit systems—even though d-level 2-local commuting circuits have efficient simulations for any d. Hardness of strong simulations does not necessarily imply that weak simulations are hard as well since strong and weak simulations are generally inequivalent concepts (cf. 3.1 for a discussion). In section 4.7 we will provide evidence that k-local commuting circuits with constant k can efficiently perform certain tasks that appear to be nontrivial for classical computers, thereby providing evidence that efficient weak simulations might not exist in general.

The proof of theorem 4.5.1 is given below. Our approach is to relate simulations of 3-local commuting circuits to the evaluation of matrix elements of *universal* unitary quantum circuits, which is known to be hard. The following three lemmata collect preliminary results. First we recall that the evaluation of matrix elements of universal quantum circuits is known to be hard. This is very similar to the theorem 3.1.1.

Lemma 4.5.2. *Let $\mathcal{U}$ be a uniform family of n-qubit quantum circuits composed of the gates H, S and CZ. If there existed an algorithm with runtime poly$(n,\log\frac{1}{\epsilon})$ which outputs an ϵ-approximation of $\langle 0|\mathcal{U}|0\rangle$ for any such circuit family, then every problem in $\#P$ has a polynomial-time algorithm.*

Proof. Consider an efficiently computable Boolean function $f:\{0,1\}^n\to\{0,1\}$. Let $s(f)$ denote the number of bit strings x satisfying $f(x)=0$. The problem of computing $s(f)$ is well known to **#P-complete**. Now define the $(n+1)$-qubit state $|f\rangle:=2^{-n/2}\sum_x|x,f(x)\rangle$ where the sum is over all n-bit strings x. Let $\mathcal{H}$ be the operator which acts as H on qubits 1 to n and as the identity on qubit $n+1$. Then an easy calculation shows

$$\langle 0|\mathcal{H}|f\rangle = s(f)/2^n. \qquad (4.23)$$

Since H, CZ and S satisfy the condition of Solovay-Kitaev theorem, there exists a uniform circuit family $\mathcal{V}$ composed of these gates such that $\mathcal{V}|0\rangle$ is δ-close to $|f\rangle$ with $\delta := 2^{-n^2}$. Denote the circuit $\mathcal{U} := \mathcal{H}\mathcal{V}$. Using (4.23) it follows that

$$|\langle 0|\mathcal{U}|0\rangle - \frac{s(f)}{2^n}| \leq \delta. \tag{4.24}$$

Now suppose that there exists a $\text{poly}(n, \log \frac{1}{\epsilon})$ classical algorithm to compute $\langle 0|\mathcal{U}|0\rangle$ with accuracy ϵ. Setting $\epsilon = \delta$, this would imply the existence of a polynomial time classical algorithm that outputs an δ-approximation γ of $\langle 0|\mathcal{U}|0\rangle$. Using (4.24) and the triangle inequality this implies that γ approximates $s(f)/2^n$ with accuracy 2δ. Since $s(f)/2^n = k/2^n$ for some integer between 0 and 2^n, this accuracy would allow to compute $s(f)$ exactly in polynomial time, hence implying that every problem in #P has a poly-time algorithm. □

Note that this theorem can also be proved by combining theorem 3.1.1 and a technique called Hadamard test, which we will introduce later.

Second, we recall the lemma 4.2.3 from subsection 4.2.1 which relates universal quantum circuits to post-selected 2-local commuting circuits.

Lemma 4.5.3. *Let $\mathcal{U}$ be an n-qubit quantum circuit composed of the gates H, S and CZ and denote $|\psi\rangle = \mathcal{U}|0\rangle^n$. Then there exists a 2-local commuting circuit $\mathcal{C}$ on $k+n$ qubits such that $|\psi\rangle$ is obtained by postselecting $\mathcal{C}|0\rangle^{k+n}$ on the first k qubits; more precisely*

$$|0\rangle^k|\psi\rangle = \sqrt{2}^k \mathcal{P}\mathcal{C}|0\rangle^{k+n}. \tag{4.25}$$

Here $\mathcal{P}$ denotes the projector $|0\rangle\langle 0|$ acting on the first k qubits. Furthermore $k = poly(n)$ and the description of $\mathcal{C}$ can be computed efficiently on input of the description of $\mathcal{U}$.

Combining the above two lemmata shows that approximating matrix elements of commuting 2-local circuits is hard.

Lemma 4.5.4. *Let $\mathcal{C}$ be a uniform family of n-qubit 2-local commuting quantum circuits. If there existed a classical algorithm with runtime $poly(n, \log \frac{1}{\epsilon})$ which outputs an ϵ-approximation of $\langle 0|\mathcal{C}|0\rangle$ for any such $\mathcal{C}$, then every problem in $\#P$ has a poly-time algorithm.*

Proof. Let $\mathcal{U}$ be a uniform family of n-qubit quantum circuits composed of the gates H, S and CZ and let $\mathcal{C}$ be the associated commuting circuit family as in lemma 4.2.3. Using (4.4) one finds

$$\langle 0|^n\mathcal{U}|0\rangle^n = \sqrt{2}^k \langle 0|^{n+k}\mathcal{C}|0\rangle^{n+k}. \tag{4.26}$$

If an efficient classical algorithm existed to estimate $\langle 0|\mathcal{C}|0\rangle$ with exponential precision, then there also exists an algorithm to estimate $\langle 0|\mathcal{U}|0\rangle$ with exponential precision. This implies that every problem in $\#P$ has a poly-time algorithm owing to lemma 4.5.2. □

The proof of theorem 4.5.1 now proceeds by relating the simulation of 3-local commuting circuits to the evaluation of matrix elements of 2-local commuting circuits, via the Hadamard test.

Proof of theorem 4.5.1: Suppose that an efficient algorithm existed to strongly simulate the circuits described in the theorem. Consider an arbitrary n-qubit commuting circuit $\mathcal{C} = G_T \cdots G_1$ with two-qubit gates G_i. Consider the following $(n+1)$-qubit quantum circuit (the

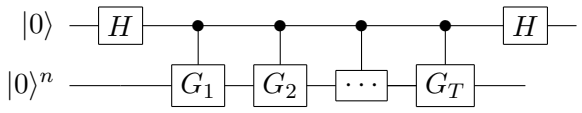

Figure 4.2: The Hadamard test

"Hadamard test") with input $|0\rangle$ as depicted in Fig. 4.2. First H is applied to the first qubit. Then each gate G_i is applied controlled on the first qubit being in the state $|1\rangle$; we denote these 3-qubit gates by CG_i. Finally, H is again applied to the first qubit. Measuring the first qubit yields the outcome 0 with probability

$$p(0) = \frac{1}{2}(1 + \ \mathrm{Re}(\langle 0|\mathcal{C}|0\rangle)). \tag{4.27}$$

Now for each i define the 3-qubit gate $U_i := [H \otimes I]CG_i[H \otimes I]$, where H acts on the first qubit, and let $\mathcal{C}'$ denote the circuit composed of the gates U_i. Since the gates G_i commute, also the gates U_i commute. Furthermore, it is straightforward to show that the circuit $\mathcal{C}'$ acting on $|0\rangle$ and followed by measurement of the first qubit is equivalent to the circuit in Fig. 4.2, since the hadamard operations "in the middle" cancel out. Thus $\mathcal{C}'$ also yields the outcome 0 with probability $p(0)$. It follows that the existence of an efficient classical algorithm to strongly simulate the circuit $\mathcal{C}'$ yields an efficient classical algorithm to compute the real part of $\langle 0|\mathcal{C}|0\rangle$ with exponential precision. Adding $P = \mathrm{diag}(1, i)$ before the second Hadamard gate in Fig. 4.2 and arguing analogously yields an efficient algorithm to estimate the imaginary part of $\langle 0|\mathcal{C}|0\rangle$ with exponential precision. Using lemma 4.2.3 we conclude that this would imply that every problem in $\#P$ has a poly-time algorithm. □

4.6 Efficient simulation of commuting Pauli Circuits

A circuit composed of unitary operators of the form $e^{i\theta P}$, where the Ps are (Hermitian) Pauli operators, is called a Pauli circuit. By theorem 1.2.4, Pauli circuits can be shown to be universal for quantum computation. Here we investigate *commuting* Pauli circuits. We allow P to act on arbitrarily many qubits i.e. we do not restrict to local gates[1].

Given the distinguished status of Pauli operators, commuting Pauli circuits constitute a simple and natural class of commuting quantum circuits. This class in fact encompasses the model of "instantaneous quantum computation" (IQP) introduced earlier in subsection 4.2.2. IQP corresponds to the subclass of commuting Pauli circuits where each P is restricted to be a tensor product of identities and Pauli X matrices, so that every gate $e^{i\theta P}$ is diagonalized by the tensor product operator $H \otimes \cdots \otimes H$. Generalizing IQP to arbitrary commuting Pauli circuits adds the interesting feature that the unitary operator which simultaneously diagonalizes the gates in the circuit is generally no longer a tensor product of single-qubit operators, but rather a global entangling operation; see example (4.2).

Whereas arbitrary Pauli circuits are universal, we will show that commuting Pauli circuits can be efficiently simulated classically in the following sense.

[1]Remark that, even for such non-local gates, every gate $e^{i\theta P}$ can be efficiently implemented on a quantum computer i.e. it can be realized by a polynomial size quantum circuit of elementary gates

Theorem 4.6.1. (Weak simulation of Commuting Pauli circuits) *Every uniform family of commuting Pauli circuits acting on a standard basis input and followed by measurement of Z acting on one of the qubits can be weakly simulated classically.*

Here we want to remark that our result can also be generalized to simulate $O(\log n)$-qubit computational basis measurements. Recall that it was shown in 4.2.4 that IQP circuits followed by such measurement can be simulated efficiently weakly . Theorem 4.6.1 hence generalizes this result to arbitrary commuting Pauli circuits. Furthermore, in subsection 4.2.1 it was shown that efficient weak classical simulation (relative to a certain special type of approximations) of 2-local IQP circuits followed by $O(n)$ computational basis measurements are highly unlikely to exist: the existence of such simulations would imply a collapse of the polynomial hierarchy. Thus a fortiori simulations of $O(n)$ computational basis measurements are unlikely to exist for general commuting Pauli circuits as well.

One can in fact show a stronger version of theorem 4.6.1: a general Pauli circuit containing a limited degree of non-commutativity can still be simulated classically efficiently.

Theorem 4.6.2. (Weak simulation of slightly non-commuting Pauli circuits) *Consider a uniform family of n-qubit commuting Pauli circuits interspersed with $O(\log n)$ gates of the form $e^{i\theta Q}$ with Q an arbitrary (Hermitian) Pauli operator. Any such circuit family acting on standard basis input and followed by measurement of Z acting on one of the qubits can be weakly simulated classically.*

The proofs of theorem 4.6.1 and 4.6.2 are given in section 4.6.2. In the preceding sections we develop the necessary tools. It is interesting that the simulation techniques used here are completely different from those used our simulations of 2-local commuting circuits (theorem 4.3.1). In particular the latter involved strong simulations whereas commuting Pauli circuits will be simulated using weak simulations combined with stabilizer methods.

4.6.1 Pauli and Clifford operators

Recall that a Pauli operator on n qubits has the form $P = \alpha P_1 \otimes \ldots \otimes P_n$, where $\alpha \in \{\pm 1, \pm i\}$ and where each P_j is one of the Pauli matrices X, Y, Z or the identity. A Pauli operator is said to be of Z-type if each P_j is either Z or the identity; X-type Pauli operators are defined analogously. Since X, Y and Z are Hermitian, a Pauli operator is Hermitian if and only if $\alpha \in \{1, -1\}$. Letting Z_k and X_k denote the operators Z and X acting on qubit k, respectively, it can be verified that every Pauli operator P can be written as

$$P = i^t \prod_k X_k^{a_k} Z_k^{b_k}, \quad \text{where } t \in \{0,1,2,3\},\ a_k, b_k \in \{0,1\}. \tag{4.28}$$

Defining the $2n$-dimensional bit string

$$r(P) = (a_1, \cdots, a_n, b_1, \cdots, b_n), \tag{4.29}$$

it is easily verified that $r(PQ) = r(P) + r(Q)$ for all Pauli operators P and Q, where addition is modulo 2.

Again recall that an n-qubit operator U is a Clifford operation if $UPU^\dagger$ is a Pauli operator for every Pauli operator P. We have the following lemma.

Lemma 4.6.3. *Let $P_1, \ldots, P_m$ be a collection of commuting n-qubit Pauli operators. Then there exists a Clifford operation $\mathcal{C}$ such that $\mathcal{C}^\dagger P_i \mathcal{C} = Q_i$ for every i, where each Q_i is a Z-type Pauli operator. Moreover each Q_j as well as the description of a poly-size Clifford circuit realizing $\mathcal{C}$ can be determined efficiently.*

Proof. It suffices to prove the result for Hermitian Pauli operators since every Pauli operator can be made Hermitian by providing it with a suitable overall phase. Thus henceforth we assume that the P_i are Hermitian. We can write all m vectors $r(P_i)$ in a $m \times 2n$ matrix and pick out a maximal set of independent row vectors over $\mathbb{Z}_2$ efficiently by Gaussian elimination. W.l.o.g. we assume these are the first l vectors. The corresponding Pauli operators $\{P_1, \ldots, P_l\} =: \mathcal{S}$ form an independent set i.e. no operator in $\mathcal{S}$ can be written as a product of the other elements of $\mathcal{S}$. In addition, no product of operators in $\mathcal{S}$ yields $-I$. Indeed suppose there exist bits x_j, not all zero, such that $P_1^{x_1} \ldots P_l^{x_l} = -I$. This would imply that $\sum x_j r(P_j) = 0$, contradicting with the linear independence of the $r(P_j)$. Since the operators in $\mathcal{S}$ are Hermitian, independent and commuting and since no product of some of these operators yields $-I$, there exists a stabilizer code $\mathcal{V}$ of dimension 2^{n-l} stabilized by $\mathcal{S}$ [7]. This implies in particular that $l \leq n$. Using standard stabilizer techniques one can efficiently compute additional Hermitian Pauli operators $\mathcal{S}' = \{R_{l+1}, \ldots, R_n\}$ such that all operators in the set $\mathcal{T} = \mathcal{S} \cup \mathcal{S}'$ mutually commute, are independent and no product of these operators yields $-I$ [7]. These n operators are the stabilizers of a 1-dimensional stabilizer code i.e. a stabilizer state $|\psi\rangle$. In other words $|\psi\rangle$ satisfies $P_i|\psi\rangle = |\psi\rangle = R_j|\psi\rangle$ for every $i = 1, \ldots, l$ and $j = l+1, \ldots, n$, and moreover it is the unique state doing so. It is well known that there exists a poly-size n-qubit Clifford circuit $\mathcal{C}$ such that $|\psi\rangle = \gamma \mathcal{C}|0\rangle^n$ for some global phase γ; moreover a description of $\mathcal{C}$ can be computed efficiently [33]. Now define $Q_i = \mathcal{C}^\dagger P_i \mathcal{C}$ for every $i = 1, \ldots, m$. Each Q_i is an efficiently computable Pauli operator since $\mathcal{C}$ is a poly-size Clifford circuit. Since $\mathcal{C}|0\rangle = |\psi\rangle$ and $P_j|\psi\rangle = |\psi\rangle$ for every $P_j \in \mathcal{S}$ one has $Q_j|0\rangle = |0\rangle$. This last property together with the fact that each Q_j is a Pauli operator implies that Q_j must be of Z-type. Finally, since each P_k with $k \geq l+1$ can be written, up to a global phase, as a product of operators within $\mathcal{S}$ and since products of Z-type Pauli operators are again of Z-type, it follows that also Q_k is of Z-type. □

In our simulation of commuting Pauli circuits we will use the technique we described in section 3.4 For our purposes, it will be relevant that every stabilizer state is CT. More precisely, for every polynomial-size Clifford circuit $\mathcal{C}$ and standard basis state $|x\rangle$ (where x is an n-bit string), the state $|\psi\rangle = \mathcal{C}|x\rangle$ is CT relative to the description of $\mathcal{C}$ and the input x. Property (a) is the content of the Gottesman-Knill theorem [12]. Property (b) was shown in [33]; in fact for every stabilizer state $|\psi\rangle$ the standard basis coefficients $\langle y|\psi\rangle$ can be computed exactly.

It is also easy to show that every Pauli operator is unitary, monomial and efficiently computable. Second, every unitary operator of the form $\exp[i\theta Q]$, where Q is any (Hermitian) Z-type Pauli operator, is diagonal and hence monomial. Furthermore it is straightforward to show that any such operator is efficiently computable. More generally, it is useful to note (and easy to show):

Lemma 4.6.4. *If $U_1, \ldots, U_k$ are efficiently computable monomial unitary n-qubit operators and $k = poly(n)$, then also $\prod_{i=1}^{k} U_i$ is efficiently computable monomial.*

4.6.2 Proof of theorem 4.6.1

For clarity we prove theorem 4.6.1 separately even though it is superseded by theorem 4.6.2. Denote the input by $|x\rangle$ where x is an n-bit string. Denote the Pauli circuit by $\mathcal{U}$ and let $e^{i\theta_j P_j}$

denote its gates ($1 \leq j \leq m$). Let $\langle Z_i \rangle$ denote the expectation value of Z_i. First we invoke lemma 4.6.3, yielding a Clifford circuit $\mathcal{C}$ satisfying $\mathcal{C}^\dagger P_j \mathcal{C} = Q_j$ for some efficiently computable Hermitian Z-type operators Q_j. It follows that

$$e^{i\theta_j P_j} = \mathcal{C} e^{i\theta_j Q_j} \mathcal{C}^\dagger \tag{4.30}$$

and therefore $\mathcal{U} = \mathcal{C}\mathcal{D}\mathcal{C}^\dagger$ where $\mathcal{D}$ is given by the product of the m diagonal operators $e^{i\theta_j Q_j}$. Denote $P = \mathcal{C}^\dagger Z_i \mathcal{C}$ which is an efficiently computable Pauli operator. Furthermore denote $|\psi\rangle := \mathcal{C}^\dagger |x\rangle$. Then

$$\langle Z_i \rangle = \langle x|\mathcal{U}^\dagger Z_i \mathcal{U}|x\rangle = \langle \psi|\mathcal{D}^\dagger P \mathcal{D}|\psi\rangle. \tag{4.31}$$

Since $\mathcal{C}$ is a Clifford circuit, $|\psi\rangle$ is a CT state. Finally $M := \mathcal{D}^\dagger P \mathcal{D}$ is monomial and efficiently computable: indeed the Pauli operator P as well as each $e^{\theta_j Q_j}$ are efficiently computable monomial, as discussed in section 3.4.1. Applying lemma 4.6.4 then shows that M is efficiently computable monomial as well. Theorem 3.53 can now be applied.

4.6.3 Proof of theorem 4.6.2

We assume w.l.o.g. that Z is measured on the first qubit. Let $\mathcal{C}'$ be obtained by interspersing the commuting Pauli circuit $\mathcal{C} = \prod e^{i\theta P_j}$ with k additional gates $e^{i\theta Q_j}$ at arbitrary places in the circuit. Write

$$e^{i\theta Q_j} = [\cos\theta] I + [i \sin\theta] Q_j \tag{4.32}$$

for ever such additional gate. Doing so, the circuit $\mathcal{C}'$ is written as a linear combination of 2^k circuits (with coefficients of the form $(\cos\theta)^l (i\sin\theta)^{k-l}$), each of which being obtained by replacing $e^{i\theta Q_j}$ by either I or Q_j. Thus every circuit in the linear combination is obtained by interspersing $\mathcal{C}$ with k Pauli operators. Using that $e^{i\theta P} Q = Q e^{\pm i\theta P}$ for every two Pauli operators P and Q, the Q_j can all be commuted to the right. As a result, we find that $\mathcal{C}'$ is written in the form

$$\mathcal{C}' = \sum_{\alpha=1}^{2^k} a_\alpha \mathcal{C}_\alpha \Sigma_\alpha, \tag{4.33}$$

where each coefficient a_α is efficiently computable, where each Σ_α is a Pauli operator and where each $\mathcal{C}_\alpha$ is a commuting Pauli circuit obtained by flipping a subset of the signs $P_j \to -P_j$ in the commuting circuit $\mathcal{C}$. Furthermore there are only $\mathrm{poly}(n)$ terms in the sum since $k = O(\log n)$ by assumption. To arrive at an efficient weak simulation of $\mathcal{C}'$ followed by measurement of Z_1, it suffices to show that each of the matrix elements

$$\langle x|\Sigma_\alpha \mathcal{C}_\alpha^\dagger Z_1 \mathcal{C}_\beta \Sigma_\beta |x\rangle \tag{4.34}$$

can be estimated efficiently with polynomial accuracy. First we can commute Z_1 to the right, transforming $\mathcal{C}_\beta$ into a commuting Pauli circuit $\overline{\mathcal{C}}_\beta$ obtained by changing some of the signs $P_j \to \pm P_j$ as before. Note that the combined circuit $\mathcal{C}_\alpha^\dagger \overline{\mathcal{C}}_\beta$ is a commuting Pauli circuit since all gates have the form $e^{\pm i\theta_j P_j}$. Furthermore $\Sigma_\alpha|x\rangle$ and $Z_1\Sigma_\beta|x\rangle$ are, up to global phases, simple standard basis states, say $|y\rangle$ and $|z\rangle$ resp., which can be computed efficiently. Analogous to the proof of theorem 4.6.1 we write $\overline{\mathcal{C}}_\alpha^\dagger \mathcal{C}_\beta = \mathcal{U}\mathcal{D}\mathcal{U}^\dagger$ where $\mathcal{U}$ is a polynomial size Clifford circuit

and $\mathcal{D}$ is a product of diagonal gates. Putting everything together we find that (4.34) is, up to an efficiently computable overall phase, of the form $\langle y|\mathcal{U}\mathcal{D}\mathcal{U}^\dagger|z\rangle$ for some standard basis states $|y\rangle$ and $|z\rangle$. Since $\mathcal{U}^\dagger|y\rangle$ and $\mathcal{U}^\dagger|z\rangle$ are CT states (see section 3.4.1) and since $\mathcal{D}$ is efficiently computable monomial, we can apply theorem 3.53 yielding an efficient classical algorithm to estimate (4.34). This proves the result.

4.7 Mapping non-commuting circuits to commuting circuits

Here we show that commuting circuits can be used to efficiently reproduce the output of certain non-commutative processes. These results will provide evidence that commuting circuits can be used to solve tasks that appear nontrivial for classical computers.

4.7.1 Two-layer circuits

For every constant k we let Γ^k denote a computational model involving a universal classical computer supplemented with a restricted quantum computer operating with uniformly generated families of k-local commuting circuits acting on an arbitrary product input state and followed by Z measurement of the first qubit. By construction, Γ^k has the power to efficiently solve every problem in the complexity class P, for every k. Our goal is to investigate whether Γ^k-computations have the potential to outperform classical computers.

Theorem 4.7.1. (Mapping k-local non-commuting to $(k+1)$-local commuting circuits) *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be uniform families of k-local n-qubit commuting circuits, where the gates in $\mathcal{C}_1$ need not commute with those in $\mathcal{C}_2$. Then there exists a polynomial time Γ^{k+1}-algorithm which approximates $\langle 0|\mathcal{C}_1\mathcal{C}_2|0\rangle$ with polynomial accuracy (with success probability exponentially close to 1).*

The above result shows that the non-commutativity in the two-layer circuit $\mathcal{C}_1\mathcal{C}_2$ can be "removed" by allowing gates to act on $k+1$ qubits. The proof is an immediate consequence of the following alternate version of the Hadamard test (which regards arbitrary, i.e. not necessarily commuting, circuits).

Lemma 4.7.2. (Alternate Hadamard test) *Let $\mathcal{U} = U_{2m}\cdots U_1$ be an n-qubit quantum circuit of even size $2m$. Add one extra qubit line (henceforth called qubit 1) and for every $i = 1\cdots m$ define the gate*

$$W_i = |0\rangle\langle 0| \otimes U_{2m+1-i}^\dagger + |1\rangle\langle 1| \otimes U_i, \tag{4.35}$$

which acts on qubit 1 and the qubits on which U_i and U_{2m+1-i} acted in the initial circuit $\mathcal{U}$. Consider the following circuit $\mathcal{U}'$ acting on the $(n+1)$-qubit input $|0\rangle$: first, apply H to qubit 1; second, apply the gates $W_1,\ldots,W_m$; third, apply H to qubit 1; finally measure Z on qubit 1. Then the probability of outputting 0 is

$$p(0) = \frac{1}{2}(1 + Re\langle 0|\mathcal{U}|0\rangle). \tag{4.36}$$

Analogously, replacing H in the third step by HP with $P = diag(1,i)$ yields the imaginary part of $\langle 0|\mathcal{U}|0\rangle$.

Remark that lemma 4.7.2 requires $\mathcal{U}$ to have even size. This is however not an essential requirement since a circuit of odd size $2m+1$ can be "padded" with an additional identity. This yields a circuit $\mathcal{U}'$ of size $m+1$.

The proof of the lemma is obtained by directly computing $p(0)$. Similar to the Hadamard test, the above result provides a simple quantum algorithm to estimate matrix elements of unitary quantum circuits with polynomial accuracy (and with success probability exponentially close to 1). Different from the standard Hadamard test, however, is that the *size* of the circuit $\mathcal{U}'$ used in lemma 4.7.2 is *half* the size of the original circuit $\mathcal{U}$ i.e. the alternate Hadamard test is "twice as fast". The price to pay for this is that the gates in $\mathcal{U}'$ act on a larger number of qubits: if $\mathcal{U}$ is a k-local circuit then $\mathcal{U}'$ can be as much as $(2k+1)$-local.

Proof of theorem 4.7.1: Without loss of generality we can assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ are in standard form, say $\mathcal{C}_1 = G_m \cdots G_1$ and $\mathcal{C}_2 = G'_m \cdots G'_1$ where $m = \binom{n}{k}$. By definition of the standard form, for every subset S of k qubits there is precisely one gate G_i and one gate G'_j such that $\mathrm{supp}(G_i) \subseteq S$ and $\mathrm{supp}(G'_j) \subseteq S$. By suitably labeling the gates in both circuits we can ensure that always $j = m+1-i$. Now apply lemma 4.7.2 to the circuit $\mathcal{U} := \mathcal{C}_1\mathcal{C}_2$ with the identification $U_i := G_i$ and $U_{m+i} := G'_i$ for every $i = 1 \cdots m$. Then each gate (4.35) acts on the qubits in S together with qubit 1 so that this gate is $(k+1)$-local (at most). Note furthermore that all gates W_i mutually commute. Finally, define $W'_i := [H \otimes I]W_i[H \otimes I]$ where H acts on qubit 1. Since all H gates in the middle cancel out, the $(k+1)$-local commuting circuit $\mathcal{C} = \prod_i W'_i$ acting on $|0\rangle$ followed by measurement of Z_1 yields the same output as the circuit $\mathcal{U}'$ of lemma 4.7.2. This allows to estimate the real part of $\langle 0|\mathcal{C}_1\mathcal{C}_2|0\rangle$ with polynomial accuracy within the class Γ^{k+1}. The imaginary part is treated analogously. □

4.7.2 Constant-depth circuits

Here we will relate commuting circuits with constant-depth circuits comprising *arbitrary* gates.

Theorem 4.7.3. (Estimating constant-depth matrix elements) *Let $\mathcal{U}$ be a n-qubit quantum circuit from a uniform family of circuits that has constant depth m. Then there exists a polynomial time Γ^k-algorithm to approximate $|\langle 0|\mathcal{U}|0\rangle|^2$ with polynomial accuracy (and with success probability exponentially close to 1) where $k = 2^m + 1$.*

Recall that the problem of estimating matrix elements $|\langle 0|\mathcal{U}|0\rangle|$ of polynomial size quantum circuits of arbitrary depth is known to be BQP-hard (and the naturally corresponding decision problem is BQP-complete). Theorem 4.7.3 shows that such matrix elements can be estimated efficiently with k-local commuting circuits with constant k as long as $\mathcal{U}$ has constant depth (with an exponential scaling of k with m). Although one would not expect the constant-depth matrix problem to be BQP-hard, this task appears to be nontrivial for classical computers and, to our knowledge, no efficient classical algorithm is known.

Proof of theorem 4.7.3: Letting Z_j denote the operator Z acting on qubit j, we define $Z(S) = \prod_{j\in S} Z_j$ for every subset $S \subseteq \{1, \ldots, n\}$. Using

$$|0\rangle\langle 0| = \frac{1}{2^n} \sum Z(S), \tag{4.37}$$

where the sum is over all subsets S, one finds

$$|\langle 0|\mathcal{U}|0\rangle|^2 = \langle 0|\mathcal{U}^\dagger|0\rangle\langle 0|\mathcal{U}|0\rangle = \frac{1}{2^n} \sum \langle 0|\mathcal{U}^\dagger Z(S)\mathcal{U}|0\rangle. \tag{4.38}$$

Setting $G_j := \mathcal{U}^\dagger Z_j \mathcal{U}$ yields

$$\langle 0|\mathcal{U}^\dagger Z(S)\mathcal{U}|0\rangle = \langle 0|\prod_{j\in S} G_j|0\rangle =: F(S) \tag{4.39}$$

for every subset S. Since the Z_j mutually commute, the G_j mutually commute as well as these operators are obtained by simultaneously conjugating the Z_j. Furthermore since $\mathcal{U}$ has depth m, each G_j acts on at most 2^m qubits. Thus $F(S)$ is a matrix element of a 2^m-local commuting circuit. Via the standard Hadamard test (recall Fig. 4.2 and the proof of theorem 4.5.1) one constructs a k-local commuting circuit with $k = 2^m+1$ which allows to estimate any such matrix element with polynomial accuracy in polynomial time, with success probability exponentially close to 1.

We now use these findings to give an efficient Γ^k-algorithm to estimate $\gamma := |\langle 0|\mathcal{U}|0\rangle|^2$ with polynomial accuracy. Owing to (4.38)-(4.39), one has $\gamma := 2^{-n}\sum F(S)$. Thus γ equals the expectation value of a random variable over the collection of all 2^n subsets S which takes the value $F(S)$ with uniform probability. Fix $\epsilon > 0$. First we generate K subsets $S_\alpha \subseteq \{1,\ldots,n\}$ uniformly at random. Applying the Chernoff-Hoeffding bound we find that, for some sufficiently large $K = \mathrm{poly}(n, 1/\epsilon)$, one has

$$\left|\frac{1}{K}\sum_{\alpha=1}^{K} F(S_\alpha) - \gamma\right| \leq \epsilon/2 \tag{4.40}$$

with probability exponentially close to 1. Next, as described above we can efficiently compute an estimate f_α of each $F(S_\alpha)$ using Γ^k-circuits with $k = 2^m + 1$; more precisely, we compute K numbers f_α satisfying $|f_\alpha - F(S_\alpha)| \leq \epsilon/2$. The runtime of the computation will be $\mathrm{poly}(n, 1/\epsilon)$ and the success probability exponentially close to 1. Finally, we compute $c := [\sum f_\alpha]/K$ which takes $\mathrm{poly}(n, 1/\epsilon)$ time as well. Using (4.40) and the triangle inequality it follows that $|c-\gamma| \leq \epsilon$. Thus c is our desired polynomial approximation of γ. □

Finally we note that theorem 4.7.3 can be generalized in the following rather intriguing sense: using Γ^k-circuits one can also efficiently estimate matrix elements of the form $|\langle 0|\mathcal{U}\mathcal{C}|0\rangle|^2$ where $\mathcal{U}$ is again a constant-depth circuit and where $\mathcal{C}$ represents an arbitrary uniform family of *Clifford circuits*. Interestingly, these Clifford circuits need not have constant depth.

Theorem 4.7.4. *Let $\mathcal{U}$ be a (uniform family of) n-qubit quantum circuit(s) of depth m. Let $\mathcal{C}$ be a (uniform family of) n-qubit Clifford circuit(s). Then the problem of estimating the matrix element $|\langle 0|\mathcal{C}\mathcal{U}|0\rangle|^2$ with polynomial accuracy and with success probability exponentially (in n) close to 1 is in Γ^k with $k = 2^m + 1$.*

Proof. Similar to (4.38) one has

$$|\langle 0|\mathcal{C}\mathcal{U}|0\rangle|^2 = \frac{1}{2^n}\sum \langle 0|\mathcal{U}^\dagger\mathcal{C}^\dagger Z(S)\mathcal{C}\mathcal{U}|0\rangle. \tag{4.41}$$

Since $\mathcal{C}$ is Clifford, $\mathcal{C}^\dagger Z(S)\mathcal{C} =: P$ is a Pauli operator which can moreover be determined efficiently; that we suppress dependence of P on S to simplify notation. Following (4.28), we can write

$$P = i^t \prod X_k^{a_k} Z_k^{b_k}, \quad \text{where } t \in \{0,1,2,3\},\ a_k, b_k \in \{0,1\}. \tag{4.42}$$

Now define $G_k := \mathcal{U}^\dagger X_k^{a_k} \mathcal{U}$ and $H_k := \mathcal{U}^\dagger Z_k^{a_k} \mathcal{U}$ as well as $\mathcal{C}_1 := \prod G_k$ and $\mathcal{C}_2 := \prod H_k$. Then

$$\langle 0|\mathcal{U}^\dagger \mathcal{C}^\dagger Z(S) \mathcal{C} \mathcal{U}|0\rangle = i^t \langle 0|\mathcal{C}_1 \mathcal{C}_2|0\rangle. \tag{4.43}$$

Since the Z_k mutually commute, the G_k mutually commute as well. Furthermore each G_j acts on at most 2^m qubits. Therefore $\mathcal{C}_1$ is a 2^m-local commuting circuit. Similarly, $\mathcal{C}_2$ is a 2^m-local commuting circuit as well. We can now apply theorem 4.7.1, showing that $\langle 0|\mathcal{C}_1 \mathcal{C}_2|0\rangle$ can be estimated with polynomial accuracy using Γ^k-circuits with $k = 2^m + 1$. Continuing the argument as in the proof of theorem 4.7.3 completes the proof.

4.8 Acknowledgements

First of all, I want to thank Prof. Dr. Ignacio Cirac for being my master thesis supervisor and for giving me the opportunity to work in such a fascinating academic environment as the MPQ Theory division.

I am thankful to my thesis supervisor Dr. Maarten Van den Nest, for introducing me to the interesting field, which is the classical simulation of quantum computation. His patience and guidance of all aspects of science is very much appreciated.

I want to thank Juan Bermejo Vega for many helpful discussion, the jokes and laughters he provides and the calmness when we fell into the cold river. I also want to thank my office mates Christine Muschik, Dominic Kohler, Thorsten Wahl, Alexander Müller-Hermes, Eliska Greplova for their companion and the always very good atmosphere; and to Veronika Lechner and Dr. Maria Eckholt Perotti for countless hours of administrative help.

The deepest thank of all goes to my family for unconditional support, foremost to my parents.

Bibliography

[1] Y. Manin, "Computable and uncomputable (in russian)," *Sovetskoye Radio, Moscow*, 1980.

[2] R. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, 1982.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[4] J. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Physical Review Letters*, vol. 74, no. 20, pp. 4091–4094, 1995.

[5] Y. Nakamura, Y. Pashkin, and J. Tsai, "Coherent control of macroscopic quantum states in a single-cooper-pair box," *Nature*, vol. 398, no. 6730, pp. 786–788, 1999.

[6] C. Lu, D. Browne, T. Yang, and J. Pan, "Demonstration of a compiled version of shor's quantum factoring algorithm using photonic qubits," *Physical Review Letters*, vol. 99, no. 25, p. 250504, 2007.

[7] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2000.

[8] D. S. Abrams and S. Lloyd, "Nonlinear quantum mechanics implies polynomial-time solution for *NP*-complete and $\#P$ problems," *Phys. Rev. Lett.*, vol. 81, pp. 3992–3995, Nov 1998.

[9] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, ACM, 1996.

[10] R. Jozsa and N. Linden, "On the role of entanglement in quantum computational speed-up," *Proc. R. Soc. A*, vol. 459, pp. 2011–2032, Mar. 2003.

[11] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, vol. 91, p. 147902, Feb. 2003.

[12] D. Gottesman, "Stabilizer Codes and Quantum Error Correction," May 1997.

[13] L. G. Valiant, "Quantum Circuits That Can Be Simulated Classically in Polynomial Time," *SIAM J. Comput.*, vol. 31, pp. 1229–1254, Apr. 2002.

[14] E. Knill, "Fermionic Linear Optics and Matchgates," Aug. 2001.

[15] R. Jozsa and A. Miyake, "Matchgates and classical simulation of quantum circuits," *Proc. R. Soc. A*, vol. 464, pp. 3089–3106, Nov. 2008.

[16] M. Van den Nest, "Simulating quantum computers with probabilistic methods," *Quantum Inf. and Comp.*, vol. 11, pp. 784–812, Apr. 2010.

[17] M. Van den Nest, "Efficient classical simulations of quantum Fourier transforms and normalizer circuits over Abelian groups," Jan. 2012.

[18] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pp. 333–342, ACM, 2011.

[19] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proc. R. Soc. A*, vol. 467, pp. 459–472, May 2011.

[20] S. Jordan, "Permutational quantum computing," *Quantum Inf. and Comp.*, vol. 10, no. 5, pp. 470–497, 2010.

[21] B. Terhal and D. DiVincenzo, "Adptive quantum computation, constant depth quantum circuits and arthur-merlin games," *Quantum Inf. and Comp.*, vol. 4, no. 2, pp. 134–145, 2004.

[22] S. Fenner, F. Green, S. Homer, and Y. Zhang, "Bounds on the power of constant-depth quantum circuits," in *Fundamentals of Computation Theory*, pp. 44–55, Springer, 2005.

[23] J.-L. Brylinski and R. Brylinski, "Universal quantum gates," *Mathematics of Quantum Computation*, 2002.

[24] C. Dawson and M. Nielsen, "The solovay–kitaev algorithm," *Quantum Inf. and Comp.*, vol. 6, no. 1, pp. 81–95, 2006.

[25] R. Raussendorf and H. Briegel, "A one-way quantum computer," *Physical Review Letters*, vol. 86, no. 22, pp. 5188–5191, 2001.

[26] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[27] A. Ambainis and O. Regev, "An elementary proof of the quantum adiabatic theorem," *arXiv preprint quant-ph/0411152*, 2004.

[28] D. Aharonov, W. Van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM review*, vol. 50, no. 4, pp. 755–787, 2008.

[29] S. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, pp. 151–158, ACM, 1971.

[30] R. Karp, "Reducibility among combinatorial problems," *50 Years of Integer Programming 1958-2008*, pp. 219–241, 2010.

[31] M. Van den Nest, "Universal quantum computation with little entanglement," *preprint arXiv:1204.3107*, 2012.

[32] M. Fannes, "A continuity property of the entropy density for spin lattice systems," *Communications in Mathematical Physics*, vol. 31, no. 4, pp. 291–294, 1973.

[33] J. Dehaene and B. De Moor, "The clifford group, stabilizer states, and linear and quadratic operations over GF(2)," *Phys. Rev. A*, vol. 68, p. 042318, Apr. 2003.

[34] I. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, 2008.

[35] X. Ni and M. Nest, "Commuting quantum circuits: efficient classical simulations versus hardness results," *Accepted by Quantum Information & Computation, preprint arXiv:1204.4570*, 2012.

[36] D. Shepherd and M. J. Bremner, "Temporally unstructured quantum computation," *Proc. R. Soc. A*, vol. 465, pp. 1413–1439, Jan. 2009.

[37] D. Shepherd, "Binary matroids and quantum probability distributions," *preprint arXiv:1005.1744*, May 2010.

[38] S. Aaronson, "Quantum computing, postselection, and probabilistic polynomial-time," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 461, no. 2063, pp. 3473–3482, 2005.

[39] P. Hayden, D. Leung, P. Shor, and A. Winter, "Randomizing quantum states: Constructions and applications," *Commun. Math. Phys.*, vol. 250, no. 2, pp. 371–391, 2004.