3

Quantum Computational Complexity

> Computer scientists widely believe that P $\neq$ NP, but no proof is known. One could say that a lot of quasi-empirical evidence points to P not being equal to NP. Should P $\neq$ NP be adopted as an axiom, then? In effect, this is what the computer science community has done. *-Gregory Chaitin*

In the first chapter, we surveyed the class of problems that the universe cannot compute. Yet, from a pragmatic perspective, there is little difference between such non-computable problems and those that would take more the length of several human lifetimes, or perhaps the lifetime of the universe, to solve.

The CSAT Problem

Take, for example, the circuit satisfiability problem (CSAT) of size n, where Alice is given a boolean circuit that takes an n bit that either outputs 0 or 1. A given n bit input *satisfies* the the circuit if the resulting output is 1. Alice is then asked whether such an input exists.

For a general circuit, there are no known shortcuts [1]. Alice is condemned to the torturous exercise of executing the circuit for each of its 2^n possible inputs. The amount of work she has to do grows exponentially with n. Even if each individual execution takes no more than a millisecond, a CSAT problem of size 50 would take more than three centuries, and one of size 70 would require more than twice the current age of the universe. Thus, the satisfiability of even modestly sized circuits size are, for all practical purposes, non-computable. This motivates us to define a computationally problems as *hard* or *intractable*. if cannot to be solved by any *efficient* algorithm [2].

Hard and Easy Problems

This contrasts with *easy* computational tasks where the amount of work required scales at most as polynomial of the input size n. For example, the task of verifying whether a

[1] For details about the CSAT problem, see [Coo71]

[2] An algorithm is efficient if the time required to execute it scales at most as a polynomial of the input size. For further reading in regards to this field, see books regarding computational complexity, such as [Pap94].

given bit-string of length n satisfies a circuit of size n [3].

This example motivates a general classification of computational tasks, such that we can classify each of the problems experienced in various scientific disciplines or every day life as either easy or hard (Fig 3.1).

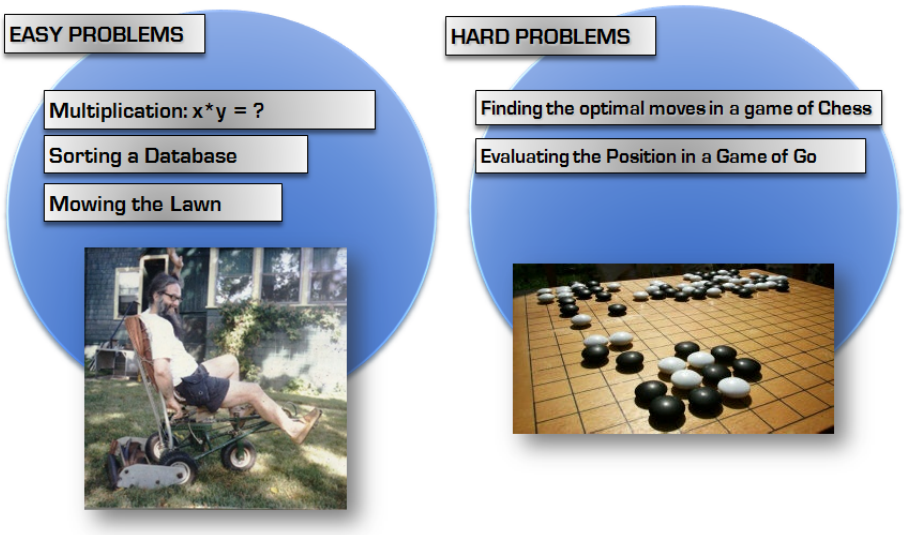

Figure 3.1: Any computational task or physical process may be classified as either easy or hard depending whether it can be solved in polynomial time. For example, mowing the lawn is easy since the amount of time required scales linearly with the size of the lawn. In contrast, finding the optimal sequence of chess moves in an $N \times N$ board is hard since the time required scales exponentially with N [Hea06].

Fundamentally hard problems

While the above arguments offer intuition on what it means for a problem to be hard, some formalities have been omitted. We have viewed computing as if it were performed by a person, and defined the scaling of computation time based on how it would scale for the person in question. The inclusion of a human being in one's definition is hardly ideal.

In this chapter, we maintain the perspective where computation is viewed as a physical process. This motivates a measure of complexity that is independent of human beings. A computational problem is *fundamentally hard* if there exists no physical process that can solve it in polynomial time, and easy otherwise.

Unfortunately, the task of determining whether a given problem is hard is highly nontrivial. There are numerous different ways in which physical processes may be harnessed for computation, and each model of computation admits many different algorithms that solves any given problem. To guarantee that a given computation problem is hard, we need to optimize over all reasonable models of computation, and all pos-

[3] The circuit need only be executed once to check that the supplied bit-string indeed satisfies the circuit

sible algorithms within each model.

One approach to this problem is to find increasingly more powerful models of computation, where a given model of computation is considered *as powerful* than another if it can efficiently solve any computational problem that is efficiently solvable in the other model, and *more powerful* if it can additionally efficiently solve some problems that the other model cannot. If we find a physically reasonable model of computation that is demonstratively as powerful as any other, then any task that has no efficient solution on this model is guaranteed to be fundamentally hard. The resulting simplification is immense, it would obviate the need to study all other computational models.

This chapter explores this approach. Section 3.1 surveys the increasingly more powerful models of computation. This motivates the development of quantum computers, devices that harness the laws of quantum mechanics to potentially solve numerous computational problems that may possess no efficient classical solution. We formalize the concepts of quantum processing in section 3.2. Section 3.3 then concludes the chapter with an outline on our current knowledge of what computational tasks quantum processes may efficiently solve.

3.1 Computational Complexity

Recall that a problem possesses an efficient solution if it may be solved in polynomial time. Thus, we first need to explicitly state a function that map each algorithm A in each model of computation to a real number $C(A)$ that is directly proportional to its runtime. $C(A)$ is commonly referred to as the *cost* of A. Each of the models of computation introduced so far (See Chapter 2) has a natural notion of cost

Turing Machines: We assume each step of a Turing machine takes equal time. Thus the cost corresponds to the number of discrete steps involved in the algorithm [NS94].

Circuit Model: Each basic logic gate is assumed to take equal time. Thus the cost corresponds to the size of the circuit [NS94].

Cellular Automata: Each iteration of a CA is assumed to take equal time. Thus the cost corresponds to the number of iterations that the CA is run [Wol94].

Optimal Algorithm

The cost in each model admits a definition of computational complexity. Given a computational problem, we define the *optimal algorithm* as the algorithm that solves the problem with minimal cost. The computational complexity is determined by the scaling of this minimal cost with respect to input size.

Each model thus has a corresponding computational complexity. In the context of Turing machines, this would be the scaling of the number of steps required by the a machine to solve the algorithm. In the case of circuit model, we would instead observe the scaling of the minimum number of primitive logic gates, and in CAs, the scaling of the minimal number of iterations.

A given computational problem is said to be easy or tractable with respect to a particular model of computation, if it may be efficiently solved by that model of computation. i.e, the cost of the optimal algorithm that solves the problem within the given model scales as a polynomial of input size. Thus, problems that are easy with respect to any particular model are considered easy, whereas a hard problem corresponds to one that is hard in all reasonable models of computation.

3.1.1 Towards More Powerful Models of Computation

Out of the three models of computation defined thus far, which one is the most powerful? It turns out that the three models of computation, along with many others proposed, are polynomially reducible to each other. This led to the speculation that not only are Turing machines universal, capable of mimicking the computational capabilities of an arbitrary physical process, but they were also able to do so efficiently. Such a line of thought is characterized by the deterministic strong Church-Turing thesis [BV97],

Deterministic Strong Church-Turing Thesis (DSCTT): *Any 'reasonable' model of computation can be efficiently simulated on a Turing machine.*

Should this thesis be true, then any task that has no efficient solution on a Turing machine will have no efficient solution in any reasonable model of computation, and is thus guaranteed to be hard. The resulting simplification is immense. Should our goal be to determine the difficulty of a computational task, the DSCTT would obviate the need to study any model of computation other than Turing machines.

Complexity class P

The possible truth of this statement motivates the complexity class P [4], the set of problems that can be solved efficiently by a Turing machine. Problems in P are guaranteed to be easy, while problems proven to lie outside P could be hard, conditioned on the truth of the DSCTT.

Like the Church-Turing thesis, the DSCTT is impossible to prove, and rather reflects a property of the universe. Even if every currently known physical model of computation was proven to offer no computational advantages over a Turing machine, the discovery of each additional physical process has the potential to violate the thesis.

[4] The study of complexity classes falls under the field of complexity theory. For a deeper discussion of this section, refer to a standard textbook on computational complexity, such as [Pap94].

Yet, even this is too optimistic, the reduction of an arbitrary model of computation to a Turing machine is highly non-trivial. There exists many physically reasonable models of computation whose computational power remains largely unknown.

For example, it remains an open question whether the addition of a random number generator to an Turing machine bestows it with greater computational power. *Polynomial identity testing* (determining whether a given polynomial is identically 0) has no known efficient solution on Turing machines, and yet can be solved in polynomial time on its probabilistic extension [Sch80]. While this could be taken as empirical evidence that probabilistic Turing machines are of greater computational power, it is just as likely that an efficient deterministic algorithm has not yet been discovered. Indeed, the problem of determining whether a given number is prime was in the same situation, till an efficient deterministic algorithm was discovered in 2004 [MA04].

Complexity Class BPP

This motivates the introduction of the complexity class BPP (bounded-error, probabilistic, polynomial time) [JTG74], the class of problems that can be efficiently solved by probabilistic Turing machines. Our inability to determine whether BPP and P coincide highlights the difficulties in this field. The distinction between whether a given computational task is intrinsically hard, or that we just haven't yet found an efficient solution, is highly non-trivial. These results motivated an amended version of the DSCTT [BV97].

Strong Church-Turing Thesis (SCTT): *Any 'reasonable' model of computation can be efficiently simulated on a probabilistic Turing machine.*

While this statement is weaker than its deterministic counterpart, the SCTT remains an empirical statement. Just as the DSCTT, new models of computation have been proposed that could potentially challenge its validity.

This time, the challenger is quantum computation. In 1994, Shor's Algorithm [Sho94] cast the truth of SCTT in serious doubt. Whereas the best known classical methods[5] for factoring a large number scaled exponentially with the number of bits it took to encode the number, Shor's algorithm exploited the laws of quantum mechanics to factor in $O(n^3)$ time. Yet, just as polynomial identity testing, it remains an open question whether this implies that quantum processes bestow additional computational power, or that an efficient algorithm for factoring on probabilistic (or standard) Turing machines is just waiting to be discovered.

Regardless, this observation motivates the introduction of another complexity class. BQP (Bounded Quantum Polynomial) represents the class of problems that have efficient solutions when quantum processes are also permitted[6]. Since quantum me-

[5] The phrase 'classical method', refers to any algorithm that can be implemented by a probabilistic Turing machines.

[6] A good summary of the complexity class is available in [NC00, KSV02].

chanics is naturally probabilistic[7], this implies that

$$\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{BPQ}. \tag{3.1}$$

Thus, although it remains unclear whether quantum computers offers computational advantages over classical computation, they are guaranteed to be at least as powerful. This motivates yet another revision of the Church Turing Thesis [8]:

Quantum Strong Church-Turing Thesis (QSCTT): *Any 'reasonable' model of computation can be efficiently simulated on a quantum computer.*

Quantum Complexity

This empirical observation, as its predecessors, is always open to revision. Alternative models of computation, based on equally reasonable physics, could well be discovered. Nevertheless, quantum computation represents the most powerful, reasonable model of computation that we currently know. This motivates the field of *quantum complexity*[BV97], where we investigate what operations quantum processes can efficiently perform.

3.2 Quantum Computation and Complexity

Quantum computation involves a fundamental paradigm shift in how we view information encoded within physical systems. In this section, we survey these key differences between classical and quantum information processing, and formalize the concepts on quantum computation.

3.2.1 The Uniquely Quantum

In classical models of computation, the intrinsic state of a physical system is synonymous with the observable properties of a system. A string of n bits is the abstraction of a physical system with 2^n possible configurations, with some corresponding experiment that allows us to distinguish between each of these possibilities. In particular, the amount of information that may be extracted by measurement from the system coincides with the amount of information needed to precisely define the state of the system (namely n bits).

In contrast, the amount of information needed to precisely define a quantum system far exceeds that of which can be extracted by measurement. Mathematical representation of the fundamental unit of quantum information requires more than a single bit. The state of a qubit is specified by a normalized vector on $\mathbb{C}^2$, i.e.,

$$c_0 |0\rangle + c_1 |1\rangle \qquad |c_0|^2 + |c_1|^2 = 1. \tag{3.2}$$

[7] Quantum measurements are generally probabilistic.

[8] This thesis has not been formally defined, but has often been alluded to in literature, such as in [EJ96]

Despite this, any measurement on the system[9] has only two possible outcomes, resulting in a single bit of information.

This difference becomes more pronounced as we scale to larger quantum systems. To properly define a quantum state $|\phi\rangle$ on a system on n qubits, a normalized vector in $\mathbb{C}^{2^n}$ is required, i.e.,

$$|\phi\rangle = \sum_{\mathbf{b}\in\mathbb{Z}_2^n} c_{\mathbf{b}} |\mathbf{b}\rangle , \qquad \sum_{\mathbf{b}} \|c_{\mathbf{b}}\|^2 \tag{3.3}$$

where the summation is taken over all bit strings of length n. Such a state requires $2^n - 1$ complex numbers to define, corresponding to each $c_{\mathbf{b}}$, while any measurement has at most 2^n distinct outcomes and can thus extract at most n bits of information. This leads to the important observations that the amount of information stored within a quantum system scales *exponentially* with respect to both (a) the size of the system, and (b) amount of information retrievable by measurement.

(a) suggests that there must exists states on n qubits that cannot be expressed as the sum of its individual constituents. The Bell states [Bel64]

$$|\phi_\pm\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle \pm |11\rangle\right) . \tag{3.4}$$

Quantum entanglement

are simple examples. Such states defy classical intuition. When we have two bits, the state of the composite system can always be specified by stating the state of each bit in sequence. In contrast, $|\phi_\pm\rangle$ admits no such procedure. There does not exists states $|\phi_1\rangle$ and $|\phi_2\rangle$ on the first and second qubit such that $|\phi_\pm\rangle = |\phi_1\rangle\,|\phi_2\rangle$. This phenomena, known as *quantum entanglement*[10], suggests the potential exponential speed-up of quantum processing.

We illustrate this intuition by a thought experiment. Consider comparing the action of flipping the first bit on a bit-string of length n and its quantum analogue[11]. In the classical system, this action transforms exactly a single bit of information; as one expects from the application of a single, elementary operation. In contrast, the effect of the analogous operation on a quantum system would result in a sign flip on 2^{n-1} of its coefficients. This suggests that elementary quantum operations have the potential to manipulate an exponential amount of information. A general quantum process cannot be efficiently simulated by a Turing machine.

This observation, however, does not guarantee a quantum computer's enhanced ability to compute. (b) implies that while an exponential amount of information is being manipulated during a general quantum process, only a linear amount of bits may be extracted by measurement. Since measurement is essential for us to actually retrieve

[9] For addition information about quantum mechanics and quantum information, see the first Chapter of [NC00].

[10] For further reading on why quantum entanglement defies classical intuition, refer to literature on the EPR Paradox [EPR35] and the subsequent Bell's inequality [Bel64].

[11] In quantum information, a bit flip exchanges the states $|0\rangle$ and $|1\rangle$

the results of a computation, it remains unclear whether this internal complexity can be exploited for extra information processing.

Box 1 (Representing Quantum Systems) :

The technical details of quantum computation can be simplified by introducing special notation to denote unitary operators that we'll frequently use[a]. Given a single qubit, we refer to $|0\rangle$ and $|1\rangle$ as the *computational basis states*. Any physical operation on a single qubit can be represented by a operator in $SU(2)$[b]. We proceed to list the matrix representation and properties of some commonly used operators in this basis:

Pauli Operators

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.5}$$

At times, the alternate notation $X = \sigma_1 = \sigma_x$, $Y = \sigma_2 = \sigma_y$, $Z = \sigma_3 = \sigma_z$ is more appropriate. X is also referred to as a bit-flip. The Pauli operators satisfy the relations

$$\sigma_i^2 = I = \sigma_0 \qquad [\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k \tag{3.6}$$

where the indices are iterated from 1 to 3 and ϵ_{ijk} is the Levi-Cavita symbol.
Each operator σ_i has two distinct eigenstates $|0\rangle_i$ and $|1\rangle_j$, such that $\sigma_i |0\rangle_i = |0\rangle_i$ and $\sigma_i |1\rangle_i = -|1\rangle_i$. By convention, the subscript is omitted for eigenstates of Z, such that $|0\rangle_z = |0\rangle$ and $|1\rangle_z = |1\rangle$. Meanwhile we introduce the notation $|0\rangle_x = |+\rangle$ and $|0\rangle_x = |-\rangle$, where

$$|\pm\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle \pm |1\rangle \right). \tag{3.7}$$

Hadamard Operator

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.8}$$

The Hadamard operator satisfies the relations

$$HXH^\dagger = Z, \qquad HYH^\dagger = -Y, \qquad HZH^\dagger = X. \tag{3.9}$$

Rotation Operators

$$U_x(\phi) = e^{-i\phi X/2} \qquad U_y(\phi) = e^{-i\phi Y/2} \qquad U_z(\phi) = e^{-i\phi Z/2}. \tag{3.10}$$

defines rotations with respect to each of the three axes X, Y and Z. A general unitary in $SU(2)$ can be decomposed into rotations in two of the three axis. i.e., there exists real numbers α,β, γ such that

$$U(\theta, \phi) = U_z(\alpha) U_x(\beta) U_z(\gamma) \tag{3.11}$$

On systems of multiple qubits, we use the subscript notation to distinguish which qubit a given operator acts on. For example X_j represents the action of X on the j^{th} qubit.

[a] For further information, see Chapter 4 of [NC00].
[b] $SU(k)$ denotes special unitary operators on k dimensions.

3.2.2 Quantum Computation and Universality

Recall that any classical computation on n bits can be cast as a boolean function $f_n : \{0,1\}^n \to \{0,1\}^n$ that maps a bit-string of length n to another. A computation is specified by an uniform family of such functions $\{f_n\}_{n=1}^{\infty}$, where each f_n specifies its desired action on an input of size n.

Analogously, any quantum process on n qubits is specified by a mapping $U_n \in SU(2^n)$ between two arbitrary quantum states of the system, where $SU(2^n)$ represents the set of special unitary operators of dimension 2^n. A general computation on a quantum system is then specified by a uniform unitary family $\{U_n\}_{n=1}^{\infty}$, where U_n defines the desired action of the task on a string of n qubits[12].

To link the two formalisms, observe that any string of classical bits in state $b_1 b_2 \ldots$ can naturally be encoded within the computational basis state $|b_1 b_2 \ldots\rangle$, and retrieved by a sequence of measurement of each qubit in the computational basis. Thus any classical computation can be converted to a quantum computation by

1. Encoding the desired input in the computational basis, resulting in a quantum state $|\phi\rangle$.
2. Applying the appropriate unitary operator U.
3. Retrieving the desired output bits by measuring each qubit in the Z basis.

Any $f_n : \{0,1\}^n \to \{0,1\}^n$ may be recast as the action of some unitary operator on a suitable quantum system [13]. Thus the set of computational tasks on quantum systems subsumes its classical analogue. Any model of computation that allows the synthesis of an arbitrary unitary operator U_n is universal.

Approximate unitaries

In fact, this condition may be weakened further. It is sufficient for us to synthesize a unitary U_ϵ that is a *close* approximation of the desired operator U. Here, the term 'close' is defined in the operational sense. Suppose U_ϵ is applied on an arbitrary input state $|\phi\rangle$ in place of U, we say that U_ϵ approximates U to an accuracy ϵ if the measurement statistics of the resulting output states $U_\epsilon |\phi\rangle$ and $U |\phi\rangle$ differ by at most 2ϵ. It can be shown that this condition equates to the requirement that

$$\|U_{approx} - U\|_2 \leq \|U\|_2 = \max_{|\phi\rangle} \|U |\phi\rangle \|_2 \leq \epsilon, \tag{3.12}$$

where $\| |\phi\rangle \|_2 = \sqrt{\langle \phi, \phi \rangle}$ is the standard Euclidean norm of $|\phi\rangle$[14]. We refer to $\|U\|_2$ as

[12] In stand literature, a unitary operation is not constrained to have determinant 1. However, since global phases can no physical significance in quantum mechanics, we can restrict the set of unitaries to $SU(2^n)$ without loss of generality.

[13] We've have omitted some details in regards to the fact that U, being unitary, can only directly implement reversible boolean functions. However, this is not a problem, since any arbitrary boolean function can be implemented by a reversible function. For more details, refer to Chapter 4 of [NC00].

[14] For a proof of this statement, see page 194-195 of [NC00].

the L_2 norm. This gives more relaxed requirements for a model of quantum computation to be universal:

A model of quantum computation is universal if for any fixed $\epsilon > 0$, and any desired unitary U, it is capable of synthesizing a U_ϵ such that $\|U_\epsilon - U\|_2 \leq \epsilon$.

Each model of quantum computation admits its own measure of complexity. For each unitary $U_n \in SU(2^n)$, we can associate a real number $T(U, \epsilon)$ that is proportional to the minimal time required to synthesize U to an accuracy ϵ within the model. Given a computational problem with associated unitary family $\{U_n\}_{n=1}^{\infty}$, and any ϵ sufficiently small, the exponential scaling of $T(U_n, \epsilon)$ with respect to n would indicate that problem has no efficient solution within the model. In particular, the condition

$$T(U_n, \epsilon) \leq \text{poly}(n, 1/\epsilon) \tag{3.13}$$

implies that $\{U_n\}_{n=1}^{\infty}$ has an efficient solution[15]. Provided the given model of computation is physically reasonable, this would imply that the problem $\{U_n\}_{n=1}^{\infty}$ is not fundamentally hard and lies within BQP.

3.2.3 The Quantum Circuit Model

Just as there exists many models of classical computation, there are numerous models of quantum computation. Quantum analogues of classical circuits [Deu89], Turing machines [Deu85] have been proposed, in addition to a number of more exotic models (See, for example, quantum adiabatic computation [FGGS00] and cluster state computation [RB01]). Fortunately, the study of quantum complexity - what problems are difficult for quantum computers - does not require analysis of all these models.

In this section, we focus on the framework of quantum circuits. The rationale is that each of the aforementioned models is polynomially reducible to quantum circuits and thus offer no greater computational capability [Yao93, AvDK+04]. In fact, standard literature often treats quantum circuits and quantum computers interchangeability [NC00]. Thus any problem that cannot be efficiently solved with any physically reasonable quantum circuit is likely to lie outside BQP [16].

The quantum circuit model is a direct analogue of classical circuits, where unitary operations replace binary functions and the set of basic logical operators are replaced by elementary quantum gates. As in classical circuits, we assume that the application of each gate takes a constant amount of time. Thus, the execution time of each quantum circuit is directly proportional to the size of the circuit. The complexity of a given

[15] $\text{Poly}(x, y)$ denotes a polynomial of x and y. Thus $T(U_n, \epsilon) \in \text{Poly}(n, 1/\epsilon)$ is shorthand for saying there exists $C, k_1, k_2 \geq 0$ such that $T(U_n, \epsilon) \leq C n^{k_1} / \epsilon^{k_2}$.

[16] There is, of course, the possibility that models of quantum computation with superior computational power do exist.

Box 2 (Quantum Gates and their Representations) :

Quantum gates are a direct generalization of classical logic gates, they represent elementary elements of a quantum computation. Each quantum gate on n qubits is represented by a unitary operation on $SU(2^n)$. The simplest examples are single qubit gates, which include the Pauli operators X, Y and Z, and the Hadamard gate H. Like classical circuits, each quantum gate can be pictorially represented in the circuit picture. For example, the action of a single qubit gate U on an input state $|\phi\rangle$ has representation

$$|\phi\rangle \; —\boxed{U}—$$

More complex gates act on multi-qubits. One such gate that will be frequently used in this course are is the CSIGN gate, with circuit and the matrix representations

$$, \qquad e^{i\pi/4}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \tag{3.14}$$

respectively [a]. When we apply an n-qubit quantum gate to a quantum system with $m > n$ qubits, we specify which n of the m qubits the gate is applied to. The remaining qubits are left unchanged. Thus, the application of this gate can correspond up to $n!/m!$ distinct unitary operations.

Given a set of quantum gates $\mathcal{G}$ and a n qubit system, we can specify $\mathcal{G}_n \in SU(2^n)$ as the set of unitary operations that directly correspond to the application of a single quantum gate in $\mathcal{G}$. For example, on a 3 qubit system, a gate set $\mathcal{G} = \{X\}$ would have corresponding

$$\mathcal{G}_n = \{I, X_1, X_2, X_3\}. \tag{3.15}$$

We refer to $\mathcal{G}_n$ as the set of *allowable unitaries* on n qubits with respect to $\mathcal{G}$.

[a] The factor of $e^{i\pi/4}$ is added to ensure the gate has determinant 1, and has no physical significance. In cases where this is unnecessary, this factor is generally omitted.

unitary operation U is then the minimal number of basic quantum gates required to synthesize U.

Formally denote the set of allowable quantum gates by $\mathcal{G}$. On a system of n qubits, this leads to a set of allowable unitaries $\mathcal{G}_n$ (See Box 2). A quantum circuit that synthesizes U corresponds to a sequence of (possibly identical) unitaries $V_1, V_2, \ldots V_m$ in $\mathcal{G}_n$ such that

$$U_n = \prod_{i=1}^{m} V_i. \tag{3.16}$$

The number of unitaries used m, defines the size of the circuit and thus is directly proportional to time required to execute the circuit. The optimal circuit is the circuit that implements U with the smallest m. We denote the size of this optimal circuit by $m_{\mathcal{G}}(U)$, which is referred to as the *gate complexity* of U.

Gate Complexity

Recall, however, the exact synthesis of a unitary U is unnecessary for computation. It is thus also useful to introduce the *approximate gate complexity* $m_{\mathcal{G}}(U, \epsilon)$, which defines minimal number of quantum gates required to synthesize a unitary U_ϵ such

Approximate Gate Complexity

that $\|U_\epsilon - U\| \leq \epsilon$ [17].

Given the assumption that each element of $\mathcal{G}$ takes roughly equal time, $m_\mathcal{G}(U,\epsilon)$ becomes a measure of optimal time required to synthesize U. Thus, the polynomial dependence of $m_\mathcal{G}(U_n,\epsilon)$ with respect to n and $1/\epsilon$ determines whether a given computational process defined by $\{U_n\}_{n=0}^\infty$ has an efficient quantum circuit implementation, and hence in BQP.

So far, we have avoided an explicit description of the set of allowable gate set $\mathcal{G}$. Since we have taken BQP to be the class of computational problems that can be solved by any physically reasonable quantum circuit, we need to choose a $\mathcal{G}$ such that it is (a) physically reasonable, and (b) results in a computational model that is as powerful as one with any other physically reasonable choice of $\mathcal{G}$.

One clear class of unitaries that could well violate the criterion of physical reasonability are those that interact an unbounded number of qubits. While a priori, this still gives us a vast number of different choices for $\mathcal{G}$, it is sufficient to take $\mathcal{G}$ as the set of all single qubit gates, together with the CSIGN. This is justified by

Theorem 1 *Let $\mathcal{G} = SU(2) \cup \{\mathrm{CSIGN}\}$, and $\mathcal{G}'$ be any set of quantum gates that do not involve unitaries which interact an unbounded number of qubits. Then, for any computational task with unitary family $\{U_n\}_{n=0}^\infty$,*

$$m_\mathcal{G}(U,\epsilon) \leq \mathrm{poly}(n, 1/\epsilon, m_{\mathcal{G}'}(U,\epsilon)). \tag{3.17}$$

Thus a quantum circuit constructed from a single qubit and CSIGN gates can efficiently simulate any other reasonable quantum circuit.

The proof of this theorem lies in the observation that gates in $\mathcal{G}'$ may be implemented by at most K gates from $\mathcal{G}$ up to some desired accuracy δ (see for example, Chapter 4 of [NC00]). Thus $Km_{\mathcal{G}'}(U,\epsilon)$ gates from $\mathcal{G}$ can simulate a circuit constructed from elements of $\mathcal{G}'$ up to an accuracy $m_{\mathcal{G}'}(U,\epsilon)\delta$. A suitable choice of δ completes the proof.

There exists many possible physical implementations of single qubit and CSIGN gates [18]. Thus a quantum circuit model where they form the basic gates is regarded as physically reasonable. This observation, together with Theorem 1, indicates $\mathcal{G} = SU(2) \cup \{\mathrm{CSIGN}\}$ satisfies both (a) and (b) and hence is an appropriate choice for a universal gate set.

We will adopt a more lenient set of elementary gates. We take $\mathcal{G} = SU(4)$, the set of all single and two qubit operations. Since this gate set is a superset of single qubit gates

[17] The term approximate gate complexity was introduced in Michael A. Nielsen and Mark R. Dowling and Mile Gu and Andrew C. Doherty [NDGD06].

[18] The number of different physical models is too numerous to mention, see for example, Chapter 7 of [NC00] for a quick survey.

and CSIGN, the resulting computational model is at least as powerful. Theorem 1 then guarantees that the two models are polynomially reducible to each other. Thus, from the perspective of analyzing what problems are hard, whether the set of elementary gates is $SU(4)$ or $SU(2) \cup \{\text{CSIGN}\}$ is irrelevant. We use the former due to its mathematical simplicity.

From this point, any mention of gate complexity and approximate gate complexity will implicitly assume that the set of elementary gates is $SU(4)$

Thus, the study of *quantum complexity* involves the determination of what problems can be efficiently decomposed into single and two qubit gates.

3.3 Quantum Complexity

We motivated this chapter with the question, 'What makes a problem hard?'. In seeking to answer this problem within the framework of physical processes, we sequentially reviewed computational models of increasing power. Eventually, this motivated the the proposal of quantum computers, and the class of problems BPQ which they can efficiently solve. Given that quantum ccomputers have the potential to be the most powerful model of computation allowable in physical reality, a problem that is intractable in this framework could potentially be fundamentally difficult within the laws of physics.

These results suggest that to determine whether a problem is hard, a good start would be to classify whether or not it lies within BQP. Alas this is far trickier in practice. There exists no known method that systematically decides whether or not a given problem is either easy or hard, whether with respect to Turing machines, quantum circuits, or even a proof that quantum circuits are the most powerful model of quantum computation.

When we are a given a computational problem that has no known efficient solution, there exists one of two possibilities. Either (a) the of computational problem is intrinsically hard, and truly lies outside BQP or (b) an efficient quantum algorithm exists, but has not yet been discovered. There exists no easy method to distinguish between these two possibilities, and thus we have little idea on the exact size of BQP, or what lies within it.

The problem of finding which problems are difficult, appears to be a difficult problem.

Complexity class: PSPACE

So far, we know that quantum computers are capable of efficiently solving any problem that is efficiently solvable on probabilistic Turing machines. Thus BPP $\subseteq$ BQP. In

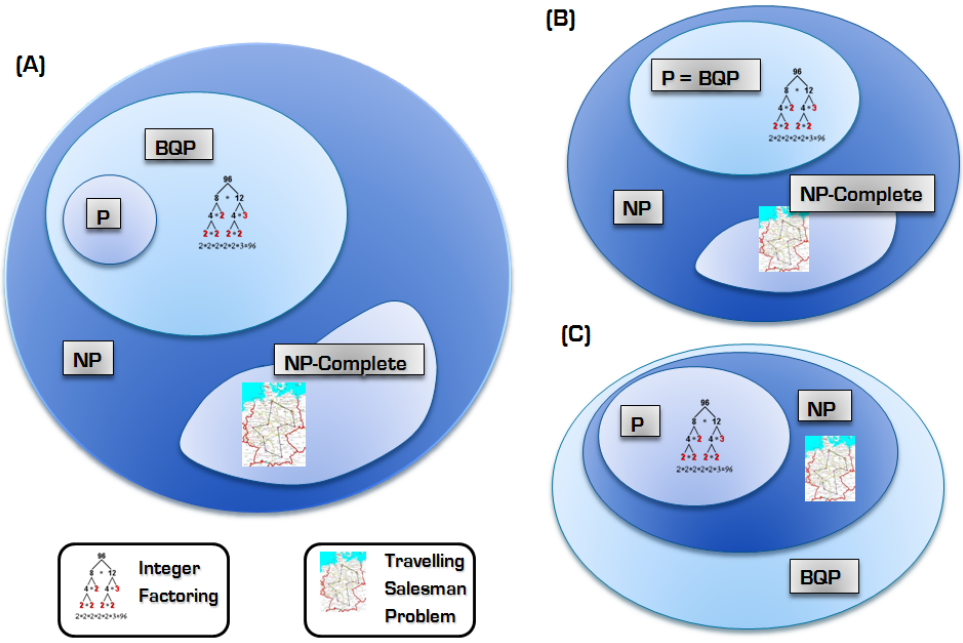

Figure 3.2: The relations between P, NP and BQP are currently not known. (A) represents popular opinion, where quantum computers are able to perform certain tasks (e.g. factoring) better than classical computers, but are unable to solve all NP problems. There remains the possibility, however, that classical and quantum processors are of equal power (B), or that quantum processors can solve all NP problems (C).

addition, one may define PSPACE, the class of computational problems that can be solved by a Turing machine with a polynomial amount of memory, but unlimited time and show that BQP $\subseteq$ PSPACE [KL01]. Combining these results gives

$$\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PSPACE} \tag{3.18}$$

To highlight our lack of knowledge in this field, it remains an open question whether any or all of the above subset relations are strict (See Fig. 3.3). It is quite possible that BPP $=$ PSPACE, or even P $=$ PSPACE, which would imply that quantum computers, at least in their most recognized form, offer no computational advantages over their classical counterpart.

The significant number of important open questions (for another example, see Box 3) highlight our dearth of knowledge in the field of computational complexity. Indeed, there's still a long way to go before we're close to our intended goal, the characterization of all computational problems that are fundamentally hard.

Box 3 (Does P equal NP?) :

Another complexity class worth mentioning is the class NP, the set of computational problems whose solutions may be checked efficiently on a Turing machine. Many well known computational problems lie within this category, including integer factoring and the CSAT problem. P $\subseteq$ NP since solving a problem directly and comparing it to a proposed solution is a valid way of checking the proposal. However, whether P is a strict subset of NP remains an open problem of great importance [a].

The question holds much philosophical significance. The process of solving a problem is considered to take a good deal of thought. In contrast, verifying whether a given theorem is correct is generally considered a much simpler process. Thus the question of whether P is equal to NP mirrors a question of whether or not problem solving is fundamentally harder than solution verification, at least on Turing machines.

The advent of quantum computers adds an extra element to the question. Given that nature admits quantum processes, one could ask whether quantum computation could solve NP problems efficiently, and thus undermine the difficulty of problem solving. Shor's factoring algorithm indicates that quantum computers can solve certain NP problems that are speculated to lie outside P. Yet, other NP problems, such as CSAT, currently do not have an efficient quantum solution. Meanwhile, there exists no proof that quantum computers cannot solve problems outside NP.

[a] This is reflected by its inclusion as one of the eight millennium problems by the Clay Mathematics Institute

References

[AvDK+04] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 42–51, Washington, DC, USA, 2004. IEEE Computer Society.

[Bel64] John S. Bell. On the Einstein-Podolsy-Rosen paradox. *Physics*, 1:195–200, 1964. Reprinted in J.S. Bell, *Speakable and Unspeakable in Quantum Mechanics* (Cambridge University Press, Cambridge, 1987).

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comp.*, 26(5):1411–1473, 1997. arXiv:quant-ph/9701001.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pages 151–158, New York, 1971. Association for Computing Machinery.

[Deu85] D. Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proc. R. Soc. Lond. A*, 400:97, 1985.

[Deu89] D. Deutsch. Quantum computational networks. *Proc. R. Soc. Lond. A*, 425:73, 1989.

[EJ96] A. Ekert and R. Jozsa. Quantum computation and Shor's factoring algorithm. *Rev. Mod. Phys.*, 68:1, 1996.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, 1935.

[FGGS00] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. *arXiv:quant-ph/0001106*, 2000.

[Hea06] Robert Hearn. *Games, Puzzles and Computation*. PhD thesis, 2006.

[JTG74] III John T. Gill. Computational complexity of probabilistic turing machines. pages 91–95, 1974.

[KL01] E. Knill and R. Laflamme. Quantum computation and quadratically signed weight enumerators. *Information Processing Letters*, 79:173–179, 2001. arXiv:quant-ph/9909094.

[KSV02] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and quantum computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 2002.

[MA04] Nitin Saxena Manindra Agrawal, Neeraj Kayal. Primes is in p. *Annals of Mathematics*, 2:781–793, 2004.

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000.

[NDGD06] M. A. Nielsen, M. R. Dowling, M. Gu, and A. C. Doherty. Optimal control, geometry, and quantum computing. *Phys. Rev. A*, 311(5764):062323, 2006.

[NS94] Noam Nisan and M. Szegedy. On the degree of Boolean functions as areal polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[Pap94] C. M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[RB01] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86(22):5188–5191, 2001.

[Sch80] J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, pages 701–717, 1980.

[Sho94] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings, 35th Annual Symposium on Fundamentals of Computer Science*, Los Alamitos, 1994. IEEE Press.

[Wol94] S. Wolfram. *Cellular automata and complexity*. Addison-Wesley, Redwood City, CA, 1994.

[Yao93] A. C. Yao. Quantum circuit complexity. In *Proc. of the 34th Ann. IEEE Symp. on Foundations of Computer Science*, pages 353–361, 1993.