Quantum Computing 1/25/2022

Lecture 1: Introduction

Instructor: Dieter van Melkebeek

Welcome to CS 710, which focuses on quantum computing this semester. The area has seen a lot of development in the last thirty years, with interest increasing after Peter Shor found a polynomial-time quantum algorithm for factoring in 1994. In this course, we focus on the theoretical power of quantum computing, i.e., on algorithms and lower bounds. We do not focus on the issue of physical realizability of a quantum computer, nor on error correction and fault tolerance.

Today we begin with selected historical highlights of quantum computing, give an overview of the topics covered in this course, and discuss course logistics.

1 Scope

Quantum computing is a mode of computation that differs from classical computation by harnessing quantum mechanical phenomena at the logical level in order to solve computational tasks. Examples of quantum mechanical phenomena are superposition, interference, and entanglement – terms that will be explained later in the course. Computational tasks often come in an input-output form. Either the task is a transformation from inputs to outputs, or the goal is to realize some relationship between the inputs and outputs (e.g., the input is a graph, and the output is one of the shortest paths on the graph). Computational tasks can take other forms as well, such as cryptographic tasks involving message sending. In this course, we will focus on input-output computational tasks, but we will also touch on cryptographic ones.

The hope is that quantum computing can solve computational tasks in a more resource-efficient manner than classical computing. In particular, we focus on the amount of time it takes to solve computational tasks, but we also consider things like the energy and memory space used. A quantum advantage is achieved when a quantum algorithm exists that solves a problem faster than any possible classical algorithm. Quantum supremacy is achieved when a tractable quantum algorithm exists for a problem that would be infeasible on a classical computer. An exponential speedup would be an example of quantum supremacy. In order to prove such speedups, we use asymptotic analysis.

In order to achieve quantum advantage and quantum supremacy, we need to develop a theoretical model for quantum computation, develop algorithms within this model, and develop a physical realization of quantum computers. This course will present a formal model of quantum computation, and then study common quantum algorithmic techniques and their power.

2 Historical Highlights

2.1 1980s

In the 1980s, physicists Richard Feynman and Paul Benioff observed that simulating quantum systems on classical computers did not work well. They had the idea that a different kind of a computer, based on quantum principles, could perform these simulations better. Further, they suggested that such a quantum computer could speed up computation for other problems standardly solved using classical computers.

David Deutsch gave the first formal model of a quantum computer in 1985. After the development of the first formal model, researchers found input-output relational tasks for which they developed quantum algorithms obtaining polynomial and exponential speedups. However, these initial tasks were contrived and were not interesting problems in their own right. Even so, quantum computing's potential was demonstrated in cryptographic tasks. Secure cryptographic key distribution, where an eavesdropper cannot observe a message without irreversibly altering it, was shown to be possible in the quantum setting. This cannot be done securely in the classical setting.

2.2 1990s

In the 1990s, researchers began finding interesting and practical computational tasks with quantum speedup.

In 1995, Lov Grover showed how to search for an element in an unordered list of n elements using $O(\sqrt{n})$ queries. [1] In the classical and randomized settings, we require $\Theta(n)$ queries in the worst case. Grover's algorithm uses a technique called amplitude amplification. As in classical computation, where there are important algorithmic paradigms like divide-and-conquer and dynamic programming, there are important quantum algorithmic paradigms, and amplitude amplification is such an example. Applying Grover's algorithm to exhaustive search for a satisfying assignment to an n-variable Boolean formula yields a $\tilde{O}(2^{n/2})$ algorithm [2] for satisfiability, while we don't know of any such algorithms in the classical setting. However, finding a subexponential algorithm for satisfiability is an open problem even on quantum computers. This makes Grover's algorithm an example of quantum advantage rather than quantum supremacy; the quantum algorithm's runtime is still exponential, but it is a better exponential than the best classical algorithms' runtimes.

In 1994, Peter Shor used a technique called phase estimation to develop an $\tilde{O}(n^2)$ polynomial-time quantum algorithm for factoring integers and for finding discrete logarithms. Today's best known classical algorithms for these problems have exponential running times, taking $O(2^{\Theta(\sqrt[3]{n})})$ time. An important implication of this result is that public-key cryptosystems like RSA can theoretically be broken in polynomial time using a quantum computer, since breaking RSA only requires the prime factorization of a large integer. However, quantum computers breaking RSA is not an imminent threat since the largest integer that existing quantum computers can factored is 21 (due to difficulties in error correcting that will be touched on later this lecture).

2.3 2000s

In the 2000s, researchers developed a new paradigm for quantum computation called quantum walks, a variant of classical random walks. Quantum walks are closely related to the simulation of physical quantum systems, a problem known as Hamiltonian simulation

Much research in quantum algorithms in the 2000s focused on efficient algorithms for NP-intermediate problems, problems which are known to be in NP, but are not known to be NP-complete and are not known to be in P. Examples of NP-intermediate problems are factoring, the discrete logarithm problem, the approximate shortest vector problem in lattices, and graph

[1] Note that these queries are of a different variety than our classical conception of queries, requiring quantum access to the data. This will be discussed more in the future.

[2] $\tilde{O}(g(n))$ means $O(g(n)\log^k n)$ where k is an arbitrary constant.

isomorphism. Note that lattice problems underlie many cryptosystems. However, whether there are efficient quantum algorithms to solve lattice problems is still an open question, so these cryptosystems are not immediately threatened by quantum computers.

Another direction of research was post-quantum cryptography. Crucially, there are some classical cryptosystems that remain secure in quantum settings. This means that the cryptosystems can be run using classical computers but would still be safe against quantum adversaries.

2.4 2010s

Several important advancements in quantum algorithms came out of the 2010s. Using Hamiltonian simulation, Harrow, Hassidim, and Lloyd developed the HHL algorithm, an algorithm for solving well-conditioned, sparse systems of linear equations. Sparsity means that every row in the system of equations contains only a small number of non-zero coefficients. This is realistic in many engineering settings. Well-conditioned means that if you perturb the input system of equations slightly, the exact solution does not change significantly. The best known classical algorithm for solving a well-conditioned $N \times N$ system of equations with at most s non-zero entities per row takes $O(N^2 s)$. The HHL algorithm takes only $\tilde{O}((\log N)s)$ time. At first glance this seems impossible; if we can solve a system of equations in less than N time, then we have not even looked at all of the input coefficients. However, this seeming paradox is explained because – like we saw with Grover's algorithm – the HHL algorithm uses quantum queries rather than classical ones. When solving the equation $Ax = b$, this algorithm requires b to be in superposition and requires "quantum access" to A. Similarly, the output only provides "quantum access" to the solution x. The idea of "quantum access" will be formalized at a later point.

The area of quantum approximate optimization algorithms (QAOA) was inspired by the HHL algorithm. QAOAs are subject to the same caveats about quantum access to inputs and outputs. QAOAs have applications in machine learning. Several of the quantum machine learning algorithms have subsequently been dequantized, meaning that people came up with classical algorithms with comparable efficiencies.

Another area of interest is sampling problems, in which we want to sample an object from a specific distribution or a class of distributions. A simple example would be sampling a binary string from the uniform distribution over strings of length n. It's thought that it may be easier to establish quantum supremacy by looking at sampling problems because there may be very simple quantum algorithms without the need for heavy error correction that can generate distributions which are hard to generate classically. Indeed, Google's 2019 claim about quantum supremacy was based on random quantum circuit sampling, a problem which is likely classically hard.

3 Physical Realizations

Although this is not the focus of the course, we touch on general concepts of the physical realization of quantum computers. To observe quantum effects, we need to either go to microscopic systems or to extremely low temperatures.

In microscopic quantum computation systems, quantum bits are individual elementary particles: ions, atoms, electrons, and photons. Research at UW-Madison focuses on using atoms and electrons. In an atom-based quantum system, the atoms are trapped in a lattice created by lasers. The main problem with this system is scalability. Because of the interaction between atoms, large lattice systems would take too much laser power. Using electrons as quantum bits avoids issues of scalability since the electron systems use silicon technology from classical computers. This means that once we get individual gates to work, we can use existing scaling techniques from classical computers. The current difficulty in electron systems is ensuring that neighboring electrons interact reliably.

Low temperature quantum computation systems rely on superconducting circuits. Industry research in quantum computing uses low temperature systems, and this approach seems more promising than the microscopic quantum computation systems. The approach is also developed at UW-Madison.

A major issue in both types of quantum computation systems is error. Quantum behavior of algorithms is negatively influenced by *decoherence*, which is a term for the interaction between the quantum system and the outside environment. We would like to design error-correcting algorithms that cope with decoherence. Unfortunately, even the error-correcting algorithms themselves may suffer from decoherence. Fault-tolerant computing solves the latter problem, provided the amount of interaction between the error-correcting procedure and the environment stays below a certain threshold. Error-correction is harder in the quantum setting. Because of this, if we want to implement one logical quantum bit, we need many physical quantum bits in order to catch the errors. This issue of scaling is why the largest number quantum computers can currently factor is only 21.

4 Topics Covered in This Course

This course covers topics listed below.

1. Model of quantum computation
2. Paradigms for efficient quantum algorithms
   (a) Amplitude amplification
   (b) Phase estimation
   (c) Quantum walks
3. Lower bounds