C191 - Lecture 7 - Quantum Advantages and Computational Complexity

I. SUPERDENSE CODING

We've seen the teleportation protocol, whereby a quantum state can be sent using a shared entangled state and two classical bits. Now we'll take a look at the superdense coding protocol, where one can send two classical bits with by sending only one quantum bit. Let's consider the following circuit, The initial state of qubits A and B is the

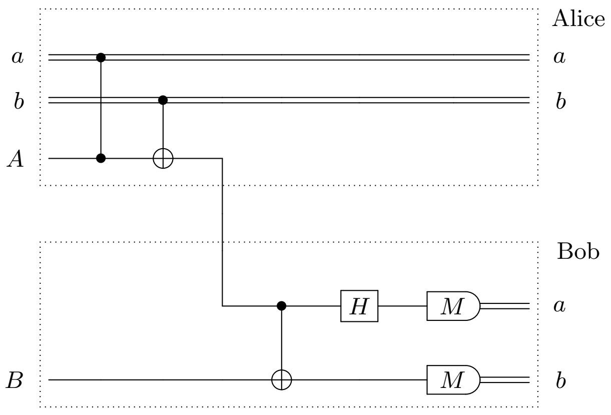


FIG. 1. Superdense coding. The initial state of qubits A and B is the entangled Bell state, $(|00\rangle + |11\rangle)/\sqrt{2}$.

entangled Bell state, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, while a,b are classical bits. After the first controlled phase gate, the state of the system becomes

$$\frac{1}{\sqrt{2}}\left(|00\rangle + (-1)^a |11\rangle\right)$$

while the state after the controlled not is

$$\frac{1}{\sqrt{2}}\left(|b0\rangle + (-1)^a |\bar{b}1\rangle\right).$$

At this point, Alice sends qubit A to Bob. Bob is now in possession of both qubits and can perform the controlled not gate, leading to the state,

$$\frac{1}{\sqrt{2}}\left(|bb\rangle + (-1)^a |\bar{b}b\rangle\right).$$

Now, a Hadamard gate sends the state $|x\rangle$ to the state $(|0\rangle + (-1)^x |1\rangle)/\sqrt{2}$, so the state after a Hadamard on the first qubit is

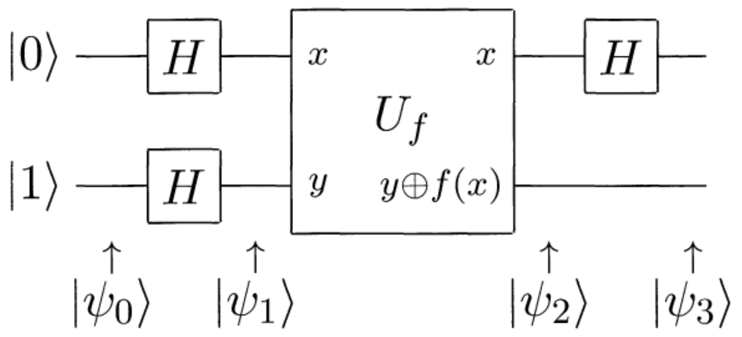
$$\begin{aligned}
&\frac{1}{2}\Big(|0b\rangle + (-1)^b |1b\rangle + (-1)^a |0b\rangle + (-1)^{a+\bar{b}} |1b\rangle)\Big) \\
&= \frac{1}{2}\Big((1 + (-1)^a) |0\rangle + ((-1)^b + (-1)^{a+\bar{b}}) |1\rangle\Big) |b\rangle \\
&= (-1)^{ab} |a\rangle |b\rangle
\end{aligned}$$

You should verify the last step above! Measurement of these two qubits will yield the classical bits a and b with 100% probability. Thus by initially sharing some entanglement, we can send *two* bits of information by sending only a single qubit. When we talk about quantum cryptography, we will see that shared entanglement is an incredibly powerful resource.

II. THE DEUTSCH-JOZSA PROBLEM

The Deutsch-Jozsa problem is what is known as a *promise problem.* In these problems we are asked to answer a question, but we are given a promise that the answer lies in some subset of all the otherwise possible answers. For instance, find a number which satisfies some property, where we are promised that the number is an integer. In the Deutsch-Jozsa problem, we are given a function on n-bit strings $f : \{0,1\}^n \to 0, 1$, which we are promised is either *balanced* or *uniform.* If f is balanced, then it returns 0 for exactly half of all possible bit strings, and 1 on the other half. If f is uniform, all inputs return the same value, though whether it is 0 or 1 is not specified. How many queries to the function must we make to determine whether f is balanced or uniform? If uniform, we would keep drawing the same bit. But if it is balanced, we could get incredibly unlucky and keep drawing inputs that yield the same output bit. In the worst case, we would have to completely exhaust that set of inputs before we saw a different output bit. This would require $2^n/2 + 1$ draws to determine whether f is balanced or uniform.

But now consider the following quantum circuit,



In the diagram the state $|0\rangle$ represents n qubits all in the zero state. The initial state of the system then is

$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$$

After the Hadamard, the state of the system is

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

The large unitary operator in the middle implements the function evaluation, leading to the state

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \otimes \frac{1}{\sqrt{2}} \left(|f(x)\rangle - \left|\overline{f(x)}\right\rangle\right)$$

Now, if $f(x) = 0$, then

$$\left(|f(x)\rangle - \left|\overline{f(x)}\right\rangle\right) = (|0\rangle - |1\rangle),$$

while if $f(x) = 1$, then

$$\left(|f(x)\rangle - \left|\overline{f(x)}\right\rangle\right) = -(|0\rangle - |1\rangle).$$

So, for any value of $f(x)$ we can write

$$\left(|f(x)\rangle - \left|\overline{f(x)}\right\rangle\right) = (-1)^{f(x)} (|0\rangle - |1\rangle).$$

Using this, we have

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

Now we take a Hadamard transformation. The effect of the Hadamard operator on a computational basis state $|x\rangle$ is,

$$H|x\rangle = \sum_y (-1)^{x\cdot y}|y\rangle ,$$

where x and y are n bit strings and $x \cdot y = \sum_i x_i y_i \mod 2$. This means,

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_y \sum_x (-1)^{f(x)+x\cdot y}|y\rangle \otimes \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$$

Now, if the function is uniform, $f(x)$ is constant. After summing over x, the only state that survives is $|00\ldots0\rangle$. But if f is balanced, the amplitude of the state $|00\ldots0\rangle$ is 0. This means that when we measure the the first n qubits, if we see all zeros, then we know the function is uniform. But if we see any other state, we know that it is balanced.

Thus we have, with a single run of the machine, determined whether f is balanced or uniform - a task that would have taken us $2^n/2+1$ runs in the worst case classically. That's a remarkable improvement!

III. COMPUTATIONAL COMPLEXITY

Computational complexity is the study of how many resources are required to solve various problems. Roughly, we tend to divide problems up into *easy* and *hard.* Easy problems are those for which the resources needed to solve them grow only like some polynomial in the input size. For example, multiplication is easy. There are many multiplication algorithms, but naively, multiplying two n-bit integers requires n shifts and roughly $n*(2n)$ bitwise additions, which we say is roughly n^2, or $\mathcal{O}\left(n^2\right)$. Now consider factoring. For some n-bit number, x, the simplest algorithm is to check all the numbers up to $\sqrt{x}$ to see if any evenly divides x. This requires us to check $2^{n/2}$ numbers, though, an exponentially difficult task - so we say that factoring is hard. Solving hard problems generally requires resources that grow exponentially in the input size - something that quickly becomes infeasible.

In fact, there are much finer divisions that we can make, organizing problems into what are known as *complexity classes.*

1. P - Polynomial time - The class of decision problems which we can solve in polynomial time (and with polynomial memory). These problems are *easy.*

2. NP - Nondeterministic polynomial - The class of decision problems which we may not be able to solve, but we can verify a solution in polynomial time (and with polynomial memory) if it is given to us. Problems in P are also in NP (note that the reverse is not necessarily true).

3. NP-Hard - Problems which are, in some sense, at least as hard as anything in NP. That is, if someone were to give you a box that solved an NP Hard problem, you could use it to solve any problem in NP in polynomial time.

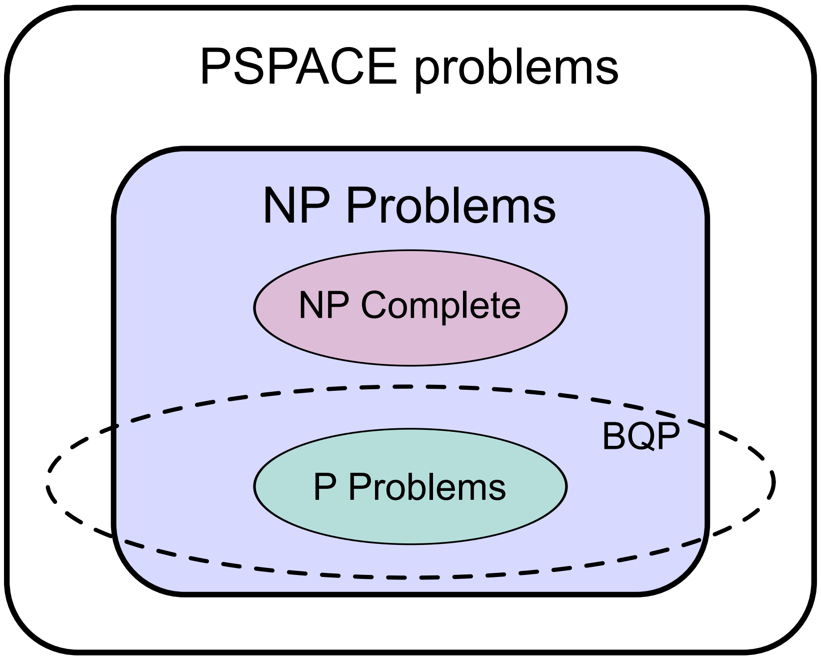
4. NP-Complete - Problems which are both NP Hard *and* are themselves in NP.

5. NP-Intermediate (NPI) - Problems which are in NP, but are not in P or in NP-Complete. This class includes factoring, graph isomorphism, discrete log, among others. This class contains problems which are potentially ripe for quantum algorithms - see BQP.

6. BQP - Bounded-error Quantum Polynomial - The class of decision problems solvable in polynomial time on a quantum computer with a probability of error less than 1/3. Especially interesting is that BQP seems to overlap somewhat with NP-Intermediate. For instance - Shor's algorithm enables a quantum computer to solve the factoring problem in polynomial time!

7. PSPACE - Polynomial space - The class of problems solvable using polynomial memory, but perhaps requiring exponential time.

The relationships between some of these classes is summarized in the following diagram:



Quantum computing has thrown a (very interesting!) wrench into classical computer science. A major tenant of classical CS has been the Church-Turing thesis. The strong version of this thesis says that

> Any model of computing can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations required.

The thesis mentions Turing machines, which can be thought of as precisely-defined, very-simple, classical computer. Few people believe that quantum computers can be efficiently simulated classically, so it is unlikely that the strong Church-Turing thesis is correct.

For more information on this topic, I encourage you to read the relevant sections in Nielsen and Chuang.