Lecture 6: Deutsch and Deutsch-Jozsa algorithm

Rajat Mittal

IIT Kanpur

Deutsch's problem can be seen as one of the simplest examples that show a quantum advantage. We will cover the algorithm in detail providing inspiration from Mach-Zehnder interferometer. The last part of the section will generalize Deutsch's algorithm to Deutsch-Jozsa algorithm giving a more impressive quantum advantage.

1 Deutsch's algorithm

Our first target is to use Mach-Zehnder interferometer to design a computational task that is resource efficient on a quantum computer but not on a classical computer. This task will be called Deutsch's problem.

1.1 Idea: Mach-Zehnder to Deutsch

Remember the Mach-Zehnder interferometer. Using constructive and destructive interference we were able to show a counter-intuitive phenomenon (look at the first part of Figure 1). Probabilistically we would expect that both detectors light up with probability 1/2; instead all photons end up at Detector-1, and Detector-2 never lights up.

What happens if we put a phase-gate for one of the paths (for second part of Figure 1, we put a phase shift on the lower path)?

Exercise 1. Try to answer this intuitively.

After the first beam-splitter we would get $H|1\rangle = |-\rangle$, putting a phase gate on the lower part is equal to giving a phase of -1 to $|0\rangle$ part. So we get $|+\rangle$ state with a global phase after the mirrors (the global phase can be ignored). Remember that the action of second beam splitter was captured by K matrix.

$$K = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}.$$

You can easily check that $K|+\rangle = |1\rangle$. So, in the second part of Figure 1, only Detector-2 will light up and Detector-1 will never light up. Hence, these two scenarios of having a phase gate at the lower part or not can be perfectly distinguished.

Note 1. This is in contrast to blocking and not blocking the lower path. When detecting a photon at Detector-1 will not tell us the difference between the two scenarios for sure. Though if we find a photon at Detector-2 it would have shown that we were in the blocking case. Refer to earlier lecture notes for this.

Let us make it more interesting, we can potentially have phase gate at the upper path too. You can check (and should check), that if both paths have (or don't have) the phase gate, we will detect photon at Detector-1. If exactly one of them has a phase gate, we get the photon at Detector-2.

Let us capture putting (and not putting) phase gate as a function on $\{0, 1\}$ (lower path and upper path). We will say that $f(0) = 1$ ($f(1) = 1$) iff the phase gate is put at the lower path (upper path respectively). You notice that we can detect whether $f(0)$ is equal to $f(1)$ or not just by sending one photon. If $f(0) = f(1)$ then we will see the photon at Detector 1, if $f(0) \neq f(1)$ then we will see the photon at Detector-2. Again, to emphasize, we can *look at* the device just once and check if $f(0) = f(1)$ or not.

"Classically" if we wanted to check whether the functions values are equal, we would have to look at both paths independently and hence look twice at the device.

Exercise 2. We noticed in the previous lecture that both the beam splitters can be thought of as Hadamard gates and the destructive interference still happens. Show that if we assume both beam splitters to be Hadamard, we can still check if $f(0) = f(1)$ or not by sending just one photon.

Deutsch Algorithm does the same in the circuit model.

1.2 The problem

Suppose there is a function $f : \{0, 1\} \to \{0, 1\}$. We are given a gate (oracle, either XOR or phase) which can give us $f(b)$, given an input b. Deutsch's problem is to find whether $f(0) = f(1)$ and use minimum number of applications to this oracle computing f. For a classical computer, clearly we need two applications of the gate. In other words, it has to compute both $f(0)$ as well as $f(1)$ to answer Deutsch's question.

Exercise 3. Should we give the classical implementation of the function f to the quantum computer or the quantum implementation?

We should give the quantum implementation, but what is a quantum implementation?

Subroutine in a quantum computer:

Exercise 4. What should be the equivalent of a subroutine (say on input x we get $f(x)$ classically) in a quantum computer?

Linearity in the circuit allows us to apply an operation/function on many inputs simultaneously. Notice that we said apply and not compute. This distinction will be clear after the explanation below, but you should be very careful in drawing conclusions from the next paragraph.

Suppose there is a circuit which evaluates $f(x)$ given an input x. The theory of quantum mechanics forces any operation to be reversible. So, a gate which outputs $f(x)$ on the input x is not a valid quantum gate (why?).

The same objective of computing $f(x)$ can be achieved in multiple ways, one of the possible implementations is given in the diagram below (Fig. 2). To compute the function value on a bit b, we set the control qubit to 0 and the data qubit to b (take a look at the diagram). This is called an *oracle/black-box for* f (in other words, quantum subroutine for f).

Exercise 5. What will happen if we feed the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ as the data qubit and $|0\rangle$ as the control qubit to the gate given above?

The output, by linearity, should be,

$$\frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle).$$

It seems that just by calling the gate once, the value of $f(0)$ and $f(1)$ can be computed. Though, this does not mean that we can get $f(0)$ and $f(1)$ simultaneously. As mentioned before, the only way to extract information from a quantum state is through a *measurement*. These measurements will not allow us to compute $f(0)$ and $f(1)$ together.

Exercise 6. Read about superposition on internet.

It might seem that the principle of superposition (allowing linear combinations of states) is not of much use, we can apply the function on multiple inputs but can only get the answer on one input.

Exercise 7. Show that by applying Hadamard to second qubit, oracle for f gets converted from $|x, y\rangle \to |x, y \oplus f(x)\rangle$ to $|x, y\rangle \to (-1)^{y \cdot f(x)}|x, y\rangle$.

Note 2. We have two possible oracles now to get the value of f, the first one computes value as XOR, the one in the exercise above computes the value in phase.

The trick to use superposition lies in *using* these hidden values of $f(0)$ and $f(1)$ for constructive/destructive interference.

Exercise 8. What will happen if we feed the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ as the data qubit and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ as the control qubit to the oracle given above?

1.3 The algorithm

Let us take the quantum implementation of the function discussed in the previous section.

Exercise 9. Before looking at the algorithm, think about the following questions.

1. What should be the input for data qubit?
2. Which oracle should we use, the one with XOR or one where f is computed in phase (as power of -1)?
3. How can the output help?

The algorithm comes out of the Mach-Zehnder interferometer inspiration almost directly.

There is an intuitive way to look at the algorithm. We can start in state $|+\rangle$ and compute the function over it. Since the values of $f(0)$ and $f(1)$ are needed to create interference, it might be better to compute the value of the function in phase. In other words, if $f(0) \neq f(1)$, there will be a relative phase between $|0\rangle$ and $|1\rangle$ at the data qubit. That means, we should have $|+\rangle$ state when $f(0) = f(1)$ and $|-\rangle$ otherwise.

Exercise 10. How can we distinguish between these two states?

The Figure 3 gives the idea, now look at Figure 4. Can you see why it is the implementation of the intuition?

Exercise 11. What is the purpose of the Hadamard gate to the control qubit?

Remember that the Hadamard gate on the control qubit (below one) is only meant to convert the XOR oracle into phase oracle. We can assume that the oracle is a phase oracle and remove the Hadamard gate to the control qubit. Then, our circuit is just Hadamard on the data (first) qubit, then oracle, then Hadamard again on the data qubit.

First Hadamard creates an equal superposition of two *waves*, one in state $|0\rangle$ and other in state $|1\rangle$. If $f(0) = f(1)$, then these two waves stay *in phase* even after the application of the oracle. If $f(0) \neq f(1)$, then these two waves go *out of phase* after we apply the oracle. The action of next Hadamard (on data qubit) is similar to having an interference of these waves. If they are in phase, then there is *constructive interference* and the amplitude of $|0\rangle$ is 1. If the waves are out of phase, there is *destructive interference* and the amplitude is 0 at $|0\rangle$. Given that Deutsch was a physicist, he might have thought of his algorithm in this way.

Let us look at the computation done by circuit in Figure 4 formally.

We will set the data qubit to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and control qubit to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. To ease the notation, we will use subscripts, data qubit by $|\rangle_D$ and control by $|\rangle_C$. The initial state can be written as,

$$\frac{1}{\sqrt{2}}(|0\rangle_D + |1\rangle_D)\frac{1}{\sqrt{2}}(|0\rangle_C - |1\rangle_C) = \frac{1}{2}(|0\rangle_D(|0\rangle_C - |1\rangle_C) + |1\rangle_D(|0\rangle_C - |1\rangle_C)).$$

After applying the function,

$$\frac{1}{2}(|0\rangle_D(|f(0)\rangle_C - |1 \oplus f(0)\rangle_C) + |1\rangle_D(|f(1)\rangle_C - |1 \oplus f(1)\rangle_C)).$$

If $f(0) = f(1)$ then the first qubit (data) will be in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, otherwise it will be in state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Observe that these states are the output of the Hadamard gate.

Remember that a quantum gate is always reversible. So, there exists an inverse of Hadamard gate, which is Hadamard itself. We will apply this Hadamard gate to the first qubit. We get $|0\rangle$ if $f(0) = f(1)$ and $|1\rangle$ otherwise.

By measuring the first register, we have solved the problem using just one query to the function. This shows that a quantum computer is faster than a classical computer for the Deutsch's problem.

The output of the function f has four possible values, $(00, 01, 10, 11)$. Classically, it was easy to distinguish whether f is $(00, 01)$ or $(01, 10)$. Using quantum mechanics, we were able to distinguish between $(00, 11)$ and $(01, 10)$. The amount of information gathered using one query to the black-box is only 1 bit. Surprisingly, that information (property of f) is not accessible by a classical computer.

You might complain that the problem seems very simple and the speed-up of 2 is not that impressive. This was just a toy example to illustrate how quantum property of superposition can be helpful. Throughout the course, there will be much more interesting problems and more impressive solutions.

Exercise 12. What will happen if we start from $|00\rangle$ state in Deutsch's algorithm?

2 Query model

Before we begin to take a look at a generalization of Deutsch's algorithm which gives a better speed-up, let us take a look at the *query model*. It is very similar to the quantum subroutine we described earlier. Instead of a function being the input/oracle, we will say that our input x is the oracle.

Exercise 13. What is the input for the Deutsch's algorithm?

You can always view the truth table of a function as the input, and these two notions coincide.

Most of the algorithms in the quantum world are query algorithms, or are described as query algorithms. The two best known algorithms, Shor's (the critical subroutine for order-finding) and Grover's, can both be put in this format.

The query format is different from the standard format because the input is not directly given to the algorithm. Suppose the input to the algorithm is an n bit string, $x \in \{0,1\}^n$. The algorithm has access to a black-box which on an input $i \in [n]$ outputs the bit x_i. This is described by an oracle O_x,

$$O_x|i, b\rangle \to |i, b \oplus x_i\rangle.$$

Exercise 14. Prove that the oracle O_x is unitary.

Note 3. This is in the same spirit as the transformation $(x, y) \to (x, y \oplus f(x))$.

Specifically, to obtain the i^{th} bit x_i, we input $|i, 0\rangle$ into the oracle and get the output on the second part of the register. The first part of the register is the *address* and the second part is called the *target*. Since our alphabet is $\{0, 1\}$, the target will be one qubit.

The dimension of the address part should be enough to specify the index i. Most of the time we will choose $n = 2^k$, so the index i will be a k bit string and hence the address part will be k qubits.

Note 4. Since there is a difference between query and standard model, the exponential separation between quantum and classical model do not transfer to exponential separation in the standard model.

Sometimes a different oracle is used for querying the input instead of the one mentioned above. It is called the *phase oracle* (as it puts the phase depending upon the input bit x_i),

$$O'_x|i, b\rangle = (-1)^{bx_i}|i, b\rangle.$$

You will show that the query oracle and phase oracle are equivalent and only differ by an application of Hadamard to the target part of the register.

Exercise 15. Show that,

$$O_x|i, -\rangle = (-1)^{x_i}|i, -\rangle.$$

Exercise 16. Why is this global phase not useless?

Note 5. Since these are quantum oracles, we assume that we can query in superposition. In other words, the state of the address part (as well as the target part) can be in superposition.

You can also check that,

$$O_x|i,+\rangle = |i,+\rangle.$$

Exercise 17. Using the facts above, show that the oracle O_x and O'_x are *equivalent*, in the sense that one can be converted into another.

There is another possible definition of phase query oracle, instead of $i \in [n]$, consider $i \in \{0, 1, 2, \cdots, n\}$. Then,

$$O''_x|i\rangle = (-1)^{x_i}|i\rangle,$$

Where we assume $O''_x|0\rangle = |0\rangle$.

Exercise 18. Show that if we only take $i \in [n]$ for the above oracle (don't have 0 as the address state), then we can't distinguish between x and its complement.

We will use either of the three mentioned oracles depending upon the application, remember that all three of them are equivalent.

As mentioned before, most of the time, we will choose $n = 2^k$. The input $x \in \{0,1\}^n$ is instead interpreted as a function $f : \{0,1\}^k \to \{0,1\}$ (like in Deutsch's algorithm, $k = 1$). This means, every index $i \in [n]$ is interpreted as a k bit binary string with x_i being considered as the function value $f(i)$.

This interpretation shows another reason why query model is important. Suppose there is a subroutine for f and that subroutine is very expensive. Then, query model translates to the task of computing *another function* g through this subroutine, where we want to minimize the applications of this subroutine.

Exercise 19. Convince yourself that this is just the relabelling of the input.

One example, g could be the sorting function on a list. The oracle for comparing two inputs (f) from the list can be given as an input.

3 Deutsch-Jozsa algorithm

This is going to be an extension of Deutsch's algorithm, the small 1 bit problem we saw before.

Exercise 20. If you don't remember, look at the notes for Deutsch's problem.

The Deutsch-Jozsa problem is a simple generalization of Deutsch's problem, given a string $x \in \{0,1\}^n$ (where $n = 2^k$), find whether

- all x_i's are same,
- or string x is balanced, i.e., exactly half the x_i's are 1 and other half of them are 0.

You are promised that the input falls in one of the cases mentioned above. From the discussion about viewing the input as being the output of a function f, this problem is equivalent to determining whether the corresponding function to the input is balanced or constant.

Exercise 21. Show that the task of determining $f(0) = f(1)$ is same as the above problem when $n = 2$.

We are given the query oracle,

$$O_x|i,b\rangle = (-1)^{bx_i}|i,b\rangle,$$

where i is a k length binary string. So, the state space of the oracle is $k+1$ qubits.

We can rephrase the Deutsch-Jozsa problem in terms of phases,

– All x_i are same, i.e., $|\sum_i (-1)^{x_i}| = 2^k$.
– String x is balanced, i.e., $|\sum_i (-1)^{x_i}| = 0$.

Before we describe the algorithm, let us take a look at the action of Hadamard on $|i\rangle$. As an assignment, show that,

$$H^{\otimes k}|i\rangle = \frac{1}{\sqrt{2^k}} \sum_j (-1)^{i.j} |j\rangle,$$

where $i.j$ is the bitwise inner product between two binary strings.

Exercise 22. What is the inverse of $H^{\otimes k}$?

Abusing the notation $|0\rangle = |00\cdots0\rangle$, then

$$H^{\otimes k}|0\rangle = \frac{1}{\sqrt{2^k}} \sum_j |j\rangle.$$

In other words, Hadamard spreads the amplitude to every state when applied to $|0\rangle$.
This also means: if $|\psi\rangle = \sum_j \alpha_j |j\rangle$ then $\langle\psi|H^{\otimes k}|0\rangle = \frac{1}{\sqrt{2^k}} \sum_j \alpha_j$.
That means, it also collects the amplitude to state $|0\rangle$ when applied to a state $|\psi\rangle$ (Hadamard is its own inverse).

Exercise 23. Before looking at the circuit below, can you guess the Deutsch-Jozsa algorithm?

From the observations about the Hadamard gate, the Deutsch-Jozsa algorithm can be inferred. We start with $|0, 1\rangle$, where the first part of the register is a k bit string. Using Hadamard, we can transfer it to equal superposition of all index states,

$$\frac{1}{\sqrt{2^k}} \sum_j |j, 1\rangle.$$

Then we apply the phase query oracle,

$$\frac{1}{\sqrt{2^k}} \sum_j (-1)^{x_j} |j, 1\rangle.$$

Again, using Hadamard, we can collect the amplitudes on state $|0\rangle$.

Exercise 24. Show, if the input is constant then we get state $|0\rangle$, else if the input is balanced then the amplitude on the state $|0\rangle$ is zero.

Measuring in the standard basis, if we get state $|0\rangle$ then algorithm will answer *constant* else it will answer *balanced*.

Exercise 25. Convince yourself that the circuit given in Fig. 5 works.

Notice that this algorithm is without error. Any classical deterministic algorithm will require at least $O(n)$ queries of the classical oracle to determine the correct answer, while this algorithm takes only one query to the quantum oracle.

Exercise 26. Why will any classical deterministic algorithm require $O(n)$ queries?

4 Assignment

Exercise 27. Create a circuit to convert $|00\rangle$ to equal superposition of all 4 standard basis states. Can you extend this argument to creating equal superposition over all 2^n standard states using appropriate input.

Exercise 28. Consider the oracle/blackbox for a function as defined in the text. Suppose we apply Hadamard on the input qubit x (required tensor product if the input is multiple qubits) with the second qubit $|0\rangle$. What is the output of the blackbox?

Exercise 29. Create a circuit using Hadamard gate and CNOT gate which creates $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ from $|00\rangle$ state.

References

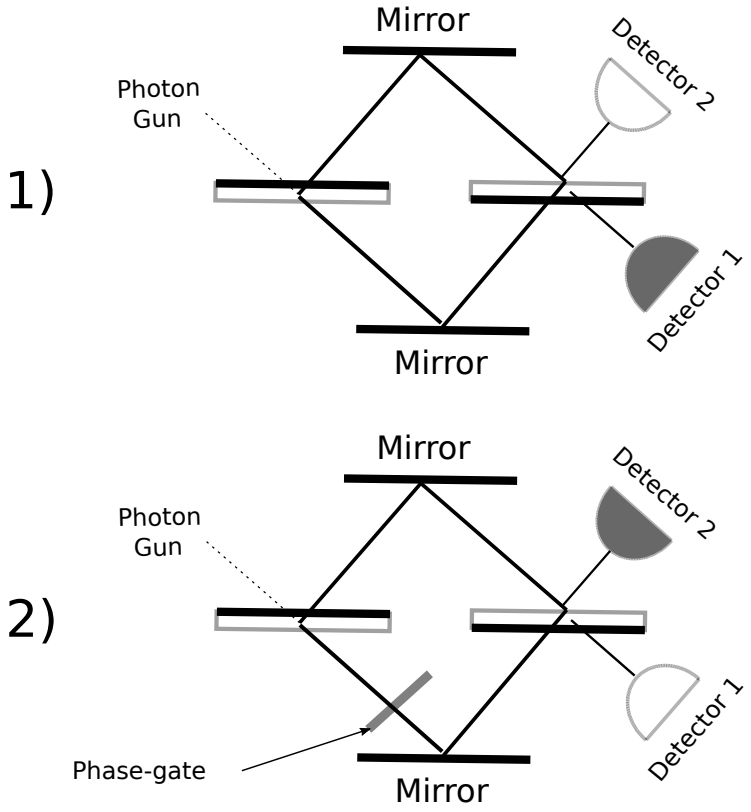

Fig. 1. Mach-Zehnder interferometer: Blocking a path allows us to see more photons at Detector-2

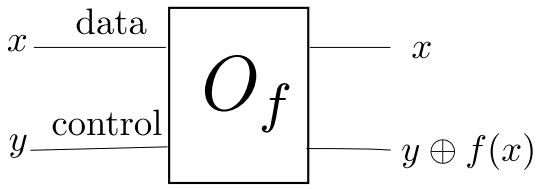

Fig. 2. Quantum implementation of computing a function, oracle for f

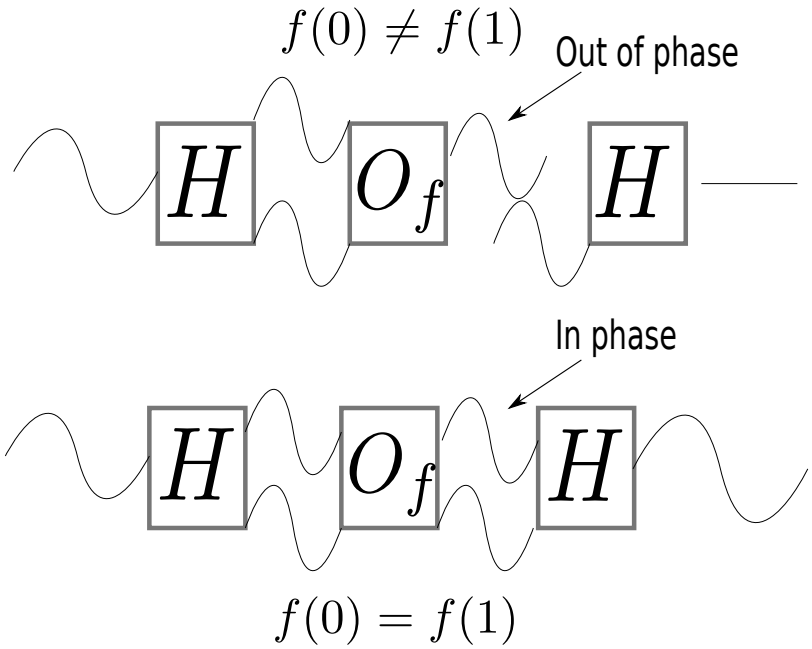

Fig. 3. Intuition for Deutsch's algorithm

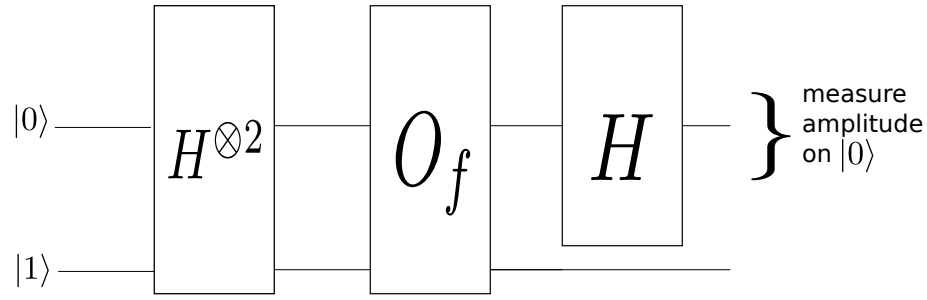

Fig. 4. Circuit for Deutsch's algorithm

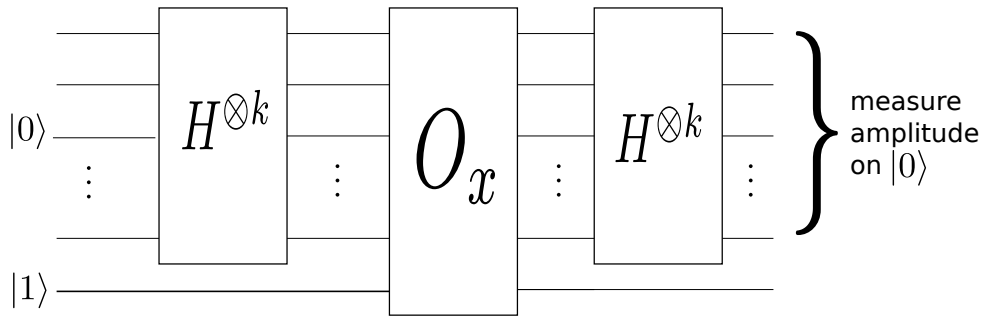

Fig. 5. Deutsch's Algorithm