QUANTUM INFORMATION & COMPUTATION

Nilanjana Datta, DAMTP Cambridge

1 The Deutsch-Jozsa (DJ) algorithm

Our first (and historically the first, from 1992) example of an exponential benefit of quantum over classical computation is a quantum algorithm (the so-called DJ algorithm) for the "balanced vs. constant" black-box promise problem, which we reiterate here:
The "balanced versus constant" problem
Input: a black box for a Boolean function $f : B_n \to B$ (one bit output).
Promise: f is either (a) a constant function ($f(x) = 0$ for all x or $f(x) = 1$ for all x) or (b) a "balanced" function in the sense that $f(x) = 0$ resp. 1 for exactly half of the 2^n inputs x.
Problem: Determine (with certainty) whether f is balanced or constant.
(At the end we will discuss the bounded error version of the problem i.e. requiring the correct solution only with some high probability, 0.999 say).

A little thought shows that classically $2^n/2 + 1$ queries (i.e. exponentially many) are necessary and sufficient to solve the problem with certainty in the worst case. Sufficiency is clear (why?). For necessity, suppose we have a deterministic classical algorithm that claims to solve this problem with certainty in every case, for any f satisfying the promise, while making $K \leq 2^n/2$ queries. Here the choice of query may even adaptively depend in any way on the results of previous queries too.
A devious adversary (having a function f) can force this algorithm to fail as follows (thus showing necessity): when the algorithm is applied to him, he actually has not a priori chosen his function f yet but simply answers 0 for all queries. At the end his function has been fixed on K inputs, but if $K \leq 2^n/2$ he is still free to complete the definition of his function to be either constant or balanced, and have it contradict whatever conclusion the algorithm reached.
Similarly for any *probabilistic* classical algorithm whose final output is required still to be correct with certainty (although probabilistic choices may be used along the way), each of the probabilistic branches of the algorithm must work with certainty themselves and the above argument applies to them, showing again that the number of queries (on any probabilistic branch) must be at least $2^n/2 + 1$.

We now show that in the quantum scenario, just *one* query suffices (with $O(n)$ extra processing steps) in every case! Our quantum black box is

$$U_f : |x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle$$

where the input register $|x\rangle$ comprises n qubits and the output register $|y\rangle$ comprises a single qubit. We assume that initially all $(n+1)$ qubits are in standard state $|0\rangle$. We begin by constructing an equal superposition of all n-bit strings in the input register (as described above) and (surprisingly!) set the output register to the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The latter is achieved by applying X and then H to the output qubit, initially in state $|0\rangle$. Thus we have the $(n+1)$-qubit state

$$\left(\frac{1}{\sqrt{2^n}} \sum_{x \in B_n} |x\rangle \right) |-\rangle .$$

Next we run the oracle U_f on this state. To see the effect of this, consider each x-term separately. We have (omitting the $\sqrt{2}$ normalisation factors)

$$\begin{aligned} U_f : |x\rangle (|0\rangle - |1\rangle) &\longrightarrow |x\rangle (|f(x)\rangle - |f(x) \oplus 1\rangle) \\ &= \begin{cases} \;\; |x\rangle (|0\rangle - |1\rangle) & \text{if } f(x) = 0 \\ -|x\rangle (|0\rangle - |1\rangle) & \text{if } f(x) = 1 \end{cases} \qquad (1) \\ &= (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

i.e. we just get a minus sign on $|x\rangle$ if $f(x) = 1$ and no change if $f(x) = 0$. This process of obtaining the $(-1)^{f(x)}$ sign is sometimes called "phase kickback". Hence on the full superposition we get

$$U_f : \left(\frac{1}{\sqrt{2^n}} \sum_{x \in B_n} |x\rangle \right) |-\rangle \longrightarrow \left(\frac{1}{\sqrt{2^n}} \sum_{x \in B_n} (-1)^{f(x)} |x\rangle \right) |-\rangle .$$

This is a product state of the n-qubit input and single qubit output registers. Discarding the last (output) qubit we get the n-qubit state

$$|f\rangle \equiv \frac{1}{\sqrt{2^n}} \sum_{x \in B_n} (-1)^{f(x)} |x\rangle . \qquad (2)$$

What does this state look like when f is constant or balanced?

If f is constant then $|f\rangle = \pm \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$. If we apply $H_n = H \otimes \ldots \otimes H$ we get $\pm |0 \ldots 0\rangle$ because H is self inverse and recall that $H_n |0 \ldots 0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \equiv |\varphi_n\rangle$.
If f is balanced then the sum in eq. (2) contains an equal number of plus and minus terms, with minus signs sprinkled in some unknown locations along the 2^n terms. But if we take the inner product of $|f\rangle$ with $|\varphi_n\rangle$ we simply add up all the coefficients in $|f\rangle$ (as $\langle x|y\rangle = 0$ if $x \neq y$ and $= 1$ if $x = y$) and wherever the minus signs occur, the total sum is always zero i.e. if f is balanced then $|f\rangle$ is orthogonal to $|\varphi_n\rangle$. Hence if we apply the unitary operation H_n (which preserves inner products), $H_n |f\rangle$ will be orthogonal to $H_n |\varphi_n\rangle = H_n(\frac{1}{\sqrt{2^n}} \sum_x |x\rangle) = |0 \ldots 0\rangle$. Hence $H_n |f\rangle$ must have the form $\sum_{x \neq 0 \ldots 0} a_x |x\rangle$ *having the all-zero term absent.* Generally $H_n |f\rangle$ will be a superposition of many basis states but for some special balanced functions f it can have the form $|x\rangle \neq |0 \ldots 0\rangle$ comprising a single basis state. See Exercise Sheet 3 (the Bernstein-Vazirani problem) for more details.

In view of the above discussion, having constructed $|f\rangle$ (for our given black box) we apply H_n and measure the n qubits in the computational basis. If the result is $0 \ldots 0$

then f was certainly constant and if the result is any non-zero string $x_1 \dots x_n \neq 0 \dots 0$ then f was certainly balanced. Hence we have solved the problem with one query to f and $(3n+2)$ further operations: $(n+1)$ H's and one X to make the input state for U_f, n H's on $|f\rangle$ and n single qubit measurements to get the classical output string.

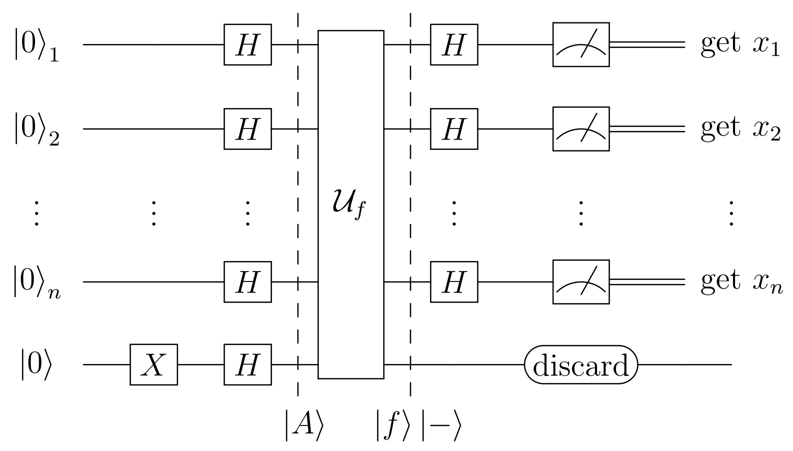

Figure. Circuit diagram for the DJ algorithm. The state $|A\rangle$ just before U_f is the equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{x \in B_n} |x\rangle$ in the first n qubits and $|-\rangle$ in the last qubit. The action of U_f then produces the state $|f\rangle$ as in the text above.

The balanced versus constant problem with bounded error

Suppose we tolerate some error i.e. require our algorithm to correctly distinguish balanced versus constant functions only with probability $> 1 - \epsilon$ for some $\epsilon > 0$. Then the above (single query) algorithm still works (as it has $\epsilon = 0$) but there is now a classical (randomised) algorithm that solves the problem with only a constant number of queries (depending on ϵ as $O(1/\log \epsilon)$ for any n and for any fixed $\epsilon > 0$). Thus we lose the all-interesting exponential gap between classical and quantum query complexities in this bounded error scenario. The classical algorithm is the following: we pick K x values, each chosen independently uniformly at random and evaluate the corresponding f values. If they are all 0 or all 1, output "f is constant". If we get at least one instance of each of 0 and 1, output "f is balanced". Clearly the second output must always be correct (as a constant function can never output both values). But the first output ("f is constant") can be erroneous. Suppose f is a balanced function. Then each random value $f(x)$ has probability half to be 0 or 1. So the probability that K random values are all 0 or all 1 is $2/2^K = 1/2^{K-1}$. This is $< \epsilon$ if $1/2^{K-1} < \epsilon$ i.e. $K > \log 1/\epsilon$ i.e. $K = O(\log 1/\epsilon)$ suffices to guarantee error probability $< \epsilon$ in every case, for all n. $\square$

So does the above prove conclusively that quantum computation can be exponentially more powerful (in terms of time complexity) than classical computation? We point out two important shortcomings in this claim.
(i) The first weakness is that if we allow any level of error in the result, however small, we lose the exponential separation between classical and quantum algorithm running times (as described in the previous paragraph). But the *exactly*-zero error scenario in computation is an unrealistic idealisation and for *realistic* computation we should always accept some (suitably small) level of error – physical computers never work perfectly and in the quantum case for example, gates depend on continuous parameters that cannot

be physically set with infinite precision. However this weakness can be fully addressed: there exist other black box promise problems for which a provable exponential separation exists between classical and quantum query complexity even in the presence of error. An example is the so-called Simon's quantum algorithm, which we will outline below.
(ii) A second (more serious) issue is the fact that the DJ problem is only a black box problem (with the black box's interior workings being inaccessible to us) rather than a straightforward "standard" computational task with a bit string as input, and no "hidden" ingredients. To convert it to a standard task we would want a class of Boolean functions $f_n : B_n \to B$ such that the balanced/constant decision is hard classically (e.g. takes exponential time in n) even if we have full access to a description of the function e.g. a formula for it or a circuit C_n that computes f_n. Note that even a constant function can be presented to us in such a perversely complicated way that its trivial action is hard to recognise! Alas, no such (*provably* hard) class of Boolean function descriptions is known.

So, are there any "standard" computational tasks for which we can prove the existence of an exponential speed-up for quantum versus classical computation? No such absolute proofs are known but the difficulty seems to be largely within the classical theory: even though many problems have only exponential-time *known* classical algorithms, they cannot be *proven* to be hard classically i.e. we cannot prove that no poly-time algorithm exists (that we have not yet discovered!) – a glaring instance of this classical theory shortcoming is the notorious fact that it is unproven that the class **NP** (cf later for more about this class) or even **PSPACE**, is strictly larger than **P**. (**PSPACE** intuitively is the class of languages that can be decided with an algorithm that uses a polynomial amount of space or memory, and can thus generally run for exponential time). However there are problems which are *believed* to be hard for classical computation (i.e. no classical poly-time algorithm, even with bounded error, is known despite much effort) for which poly-time quantum algorithms *do* exist. A centrally important such problem is *integer factorisation.* In a later lecture will describe *Shor's polynomial time quantum algorithm for factorisation* after we introduce the fundamentally important construct of the *quantum Fourier transform*, which is at the heart of the workings of Shor's algorithm.

1.1 Simon's algorithm

Consider the following black box promise problem.
Simon's problem
Input: a black box for a Boolean function $f : B_n \to B_n$ (note: n- bit output!).
Promise: f is either (a) a one-to-one function or (b) a two-to-one function of the following form – there is an n-bit string $\xi \neq 00\ldots0$ such that

$$f(x) = f(y) \quad \text{iff} \quad y = x \oplus \xi \tag{3}$$

(where $\oplus$ denotes bitwise addition modulo 2).

Problem: Determine (with bounded error probability) whether f is (a) or (b).

Remark: in the case (b) we can further ask for a determination of the string ξ. Note that for any string ξ we have $\xi \oplus \xi = 00\ldots0$ so we can say in (b) that f has *period* ξ (relative to addition of n-bit strings). $\square$

Simon's quantum algorithm (see Exercise Sheet 3 for details of how it works!) solves this problem with only $O(n)$ queries to f. On the other hand we can argue that the problem is classically *hard*, requiring an exponential number of queries so (in contrast to DJ) we get here a provable exponential separation of quantum vs. classical query complexity for *bounded error* computation.

To appreciate intuitively why the problem is classically hard, suppose that (b) actually holds. Then we will argue that we need to query f an exponential number of times to have a reasonable probability of noticing that f is not one-to-one. Indeed we obtain no information until we are lucky enough to choose two queries x and y with $f(x) = f(y)$ i.e. $x \oplus y = \xi$. Suppose for example that we choose $2^{n/4}$ queries (independently and uniformly at random). Then the number of pairs of queries is less than $(2^{n/4})^2$ and for each pair the probability that $x \oplus y = \xi$ is 2^{-n}. Thus the probability of successfully seeing the existence of ξ is less than $2^{-n}(2^{n/4})^2 = 2^{-n/2}$ i.e. even as many as $2^{n/4}$ queries cannot notice the difference between between (a) and (b) with better than an exponentially small probability and hence cannot form the basis of any *bounded* error algorithm. This argument can be made rigorous e.g. allowing arbitrary strategies for choices for queries, but we omit the technicalities here. For more details see D. Simon "On the power of quantum computation", S.I.A.M. Journal on Computing, **28**, p1474-1483 (1997) and J. Gruska "Quantum computing", chapter 3, p109-111. McGraw-Hill Publishing Company (1999).