4-2016

How to Introduce Technical Details of Quantum Computing in a Theory of Computation Class: Using the Basic Case of the Deutsch-Jozsa Algorithm

Olga Kosheleva
University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-16-21
Published in *International Journal of Computing and Optimization*, 2016, Vol. 3, No. 1, pp. 83-91.

How to Introduce Technical Details of Quantum Computing in a Theory of Computation Class: Using the Basic Case of the Deutsch-Jozsa Algorithm

Olga Kosheleva[1] and Vladik Kreinovich[2]
[1]Department of Teacher Education
[2]Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
olgak@utep.edu, vladik@utep.edu

Abstract

Many students taking the theory of computation class have heard about quantum computing and are curious about it. However, the usual technical description of quantum computing requires a large amount of preliminary information, too much to fit into an already packed class. In this paper, we propose a way to introduce technical details of quantum computing that does not require much time – it can be described in less than an hour. As such an introduction, we use a simplified description of the basic case of one of the pioneering algorithms of quantum computing.

1 Formulation of the Pedagogical Problem

Quantum computing is very promising. It is known that quantum computing is a promising direction in computing; see, e.g., [4]. Let us just give two examples:

- a quantum computing algorithm proposed by Grover enables us to search in an unsorted array of size n in time $\sqrt{n}$ [2, 3];
- a quantum computing algorithm proposed by Shor enables us to factorize large integers in polynomial time [5].

Since the difficulty of eavesdropping on RSA-encrypted messages – which underlie most current encryption schemes – relies on the difficulty of factoring large integers, this will enable us to read all encrypted messages.

Comment. This does not mean, of course, that there will be no more secrets – there exist quantum encryption schemes that cannot be decoded in this way.

At present, quantum computing is mostly theoretical. While quantum computing has a lot of promise, at present, only rudimentary quantum computers are available. In other words, quantum computing is not yet practical, it is more of a exciting theory that will eventually lead to practical efficiency.

Students are naturally curious about quantum computing. Many students have hear about quantum computing, they are curious about it, and they would like to hear the basic technical details without having to take a special course in quantum computing (which some universities offer).

A natural place for quantum computing seem to be theory of computation. Since at present, quantum computing is mostly theoretical, the natural place for students to ask for this information is the theory of computation class, which most graduate computer science students are required to take.

It is not easy to squeeze quantum computing into the theory of computation class. For many students, theory of computation is the only theoretical class that they take. As a result, this class is packed with the material that is needed for students from all areas of computer science: Turing machine, computability, NP-hardness, etc.

Most existing descriptions of quantum computing require a large mount of preliminary information, and it is not easy to squeeze all this information into the already-packed class.

What we propose. In this paper, we propose a way to introduce technical details of quantum computing that does not require much time – it can be described in less than an hour.

As such an introduction, we use a simplified description of the basic case of one of the pioneering algorithms of quantum computing – Deutsch-Jozsa algorithm [1].

2 Deutsch-Jozas Algorithm – The Simplest (But Still Convincing) Quantum Computing Algorithm

2.1 The Problem for Which This Algorithm Was Invented

What problem does this algorithm solve. We are given an algorithm $f(x)$ that takes one bit x as an input and returns one bit $f(x)$ as the output. This algorithm is given as a black box: all we can do is input x and observe the result $f(x)$.

We need to check whether the result actually depends on the input or not, i.e., whether $f(0) = f(1)$.

Why this problem is important. Often, when we process data, we try to use as much information as possible. Later, it often turns out that some of the inputs are irrelevant. If we knew this from the very beginning, we would be able to decrease the amount of input data and thus, to speed up computations.

The above problem is the simplest possible case of such situation, when we have only one input bit and only one output bit.

Example: weather prediction. To predict tomorrow's weather in El Paso, we can use today's weather data from all over the world. However, in reality, only weather in the small vicinity of El Paso is relevant: e.g., whatever happens in Africa today will not be able tomorrow's weather in El Paso.

Number of calls to f is important. Weather prediction is a good example of a program f that requires a lot of computations. It is therefore important to solve the original problem with as few calls to a (possibly time-consuming) program f as possible.

In classical (non-quantum) computations, we need two calls to f. In the usual computations, we can either apply f to input 0 or to input 1. Based on one of these result, we cannot tell whether $f(0) = f(1)$. To tell this, we need to apply f to both 0 and 1 and compare the results. Thus, we need at least two calls to the program f.

Quantum computing can help. Deutsch and Jozsa showed that by using quantum computing, we can solve this problem by using only one call to f. The goal of this paper is to explain how this can be done.

For this, we first need to describe the main features of quantum mechanics.

2.2 Basic Features of Quantum Mechanics and Quantum Computing: A Very Brief Introduction

States in quantum mechanics. In classical physics, a bit can be in two possible states: 0 and 1; let us denote these states $|0\rangle$ and $|1\rangle$. In quantum mechanics, in addition to these two "pure" states, we can also have linear combinations of such states, i.e., states of the type $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers for which $|\alpha|^2 + |\beta|^2 = 1$. Such linear combinations are called *superpositions*.

The physical meaning of such a state is that if we measure the bit in this state, we will get 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$. These probabilities should add up to 1, which explains the requirement that $|\alpha|^2 + |\beta|^2 = 1$.

States of two-bit systems. If the first bit is in state $|0\rangle$ and the second bit is in state $|1\rangle$, then we say that the whole 2-bit system is in the state $|0\rangle \otimes |1\rangle$. This state-combining operation $\otimes$ is known as the *tensor product*. The tensor product is often described, for simplicity, as $|0, 1\rangle$.

In quantum mechanics, all operations are linear. In quantum physics, all operations are linear, including the tensor product. For example, a tensor

product of the states $\alpha|0\rangle + \beta|1\rangle$ and $\alpha'|0\rangle + \beta'|1\rangle$ has the form

$$(\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha'|0\rangle + \beta'|1\rangle) = \alpha\alpha'|0,0\rangle + \alpha\beta'|0,1\rangle + \beta\alpha'|1,0\rangle + \beta\beta'|1,1\rangle.$$

In quantum mechanics, all processes are reversible. In macrophysics, many processes are irreversible: if we break a glass, we cannot make it whole again. However, on the microscopic level, all the processes are reversible. Indeed, Newton's equations, Maxwell's equations – all of them are reversible. Quantum mechanics describes microscopic effects and is, thus, reversible.

Reversibility means that we need a different representation for functions. Because of reversibility, we cannot have a constant function $f(x)$ (for which $f(0) = f(1)$) represented simply as a transformation of one bit into one bit: otherwise, based on the resulting bit, we would not be able to tell whether the original bit was 0 or 1 (i.e., this operation would not be reversible).

How functions are represented in quantum mechanics. In quantum mechanics, a generic function $f(x_1, \ldots, x_n)$ is represented as follows:

$$f : |x_1, \ldots, x_n, y\rangle \to |x_1, \ldots, x_n, y \oplus f(x_1, \ldots, x_n)\rangle,$$

where $\oplus$ is exclusive "or" (which is the same as addition modulo 2).

In particular, for functions of one variable ($n = 1$), we have $f(|x, y\rangle) = |x, y \oplus f(x)\rangle$.

This representation is indeed reversible. The above representation is reversible: specifically, we can apply the same operation f again and get back the original state $|x_1, \ldots, x_n, y\rangle$.

Indeed, once we have

$$f(|x_1, \ldots, x_n, y\rangle) = |x_1, \ldots, x_n, y'\rangle,$$

where $y' = y \oplus f(x_1, \ldots, x_n)$, then

$$f(f(|x_1, \ldots, x_n, y\rangle)) = f(|x_1, \ldots, x_n, y'\rangle) = |x_1, \ldots, x_n, y' \oplus f(x_1, \ldots, x_n)\rangle.$$

Since $y' = y \oplus f(x_1, \ldots, x_n)$, we have

$$y' = (y \oplus f(x_1, \ldots, x_n)) \oplus f(x_1, \ldots, x_n) = y \oplus (f(x_1, \ldots, x_n) \oplus f(x_1, \ldots, x_n)).$$

Once can easily check that $a \oplus a = 0$ for both $a = 0$ and $a = 1$, thus, indeed,

$$f(f(|x_1, \ldots, x_n, y\rangle)) = |x_1, \ldots, x_n, y\rangle,$$

and the transformation f is indeed reversible.

Hadamard transformations. One way to go from the pure states to superpositions is to use rotations in the 0,1-coordinate system. One of such rotations is *Hadamard transformation* H:

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle; \quad H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

One can easily see that, due to linearity, if we apply the Hadaramd transformation twice, we get the same state back:

$$H(H(|0\rangle)) = H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}H(|0\rangle) + \frac{1}{\sqrt{2}}H(|1\rangle) =$$

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|0\rangle - \frac{1}{2}|1\rangle = |0\rangle$$

and similarly,

$$H(H(|1\rangle)) = H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}H(|0\rangle) - \frac{1}{\sqrt{2}}H(|1\rangle) =$$

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle - \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle = |1\rangle.$$

Now, we are ready to start describing the Deutsch-Jozsa algorithm.

2.3 Deutsch-Jozsa Algorithm: Description and Justification

First part of the Deutsch-Jozsa algorithm:

- we start with the state $|0,1\rangle = |0\rangle \otimes |1\rangle$;
- we apply the Hadamard transformation H to both bits;
- then, we apply f;
- after that, we again apply the Hadamard transformation to both bits.

What happens when we follow these steps? We start with the state $|0,1\rangle = |0\rangle \otimes |1\rangle$, in which the first bit is in state $|0\rangle$ and the second bit is in state $|1\rangle$. When we apply the Hadamard transformation to both bits, we get

$$H(|0\rangle) \otimes H(|1\rangle) = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).$$

Due to linearity, we get

$$H(|0\rangle) \otimes H(|1\rangle) = \frac{1}{2}|0,0\rangle - \frac{1}{2}|0,1\rangle + \frac{1}{2}|1,0\rangle - \frac{1}{2}|1,1\rangle.$$

What happens next depends on the function f. Let us see what happens as a result for all four possible functions f:

- $f(0) = f(1) = 0$;
- $f(0) = f(1) = 1$;

- $f(0) = 0$ and $f(1) = 1$;
- $f(0) = 1$ and $f(1) = 0$.

Case of $f(0) = f(1) = 0$. In this case, $y \oplus f(x) = y$ for all x and y, so f does not change the state at all. Thus, when we apply again the Hadamard transformation, we simply get the original state $|0,1\rangle$ back.

Case of $f(0) = f(1) = 1$. In this case, $y \oplus f(x) = y \oplus 1$ changes 0 to 1 and 1 to 0. Thus,

$$f(|0,0\rangle) = |0,1\rangle, \quad f(|0,1\rangle) = |0,0\rangle, \quad f(|1,0\rangle) = |1,1\rangle, \quad f(|1,1\rangle) = |1,0\rangle.$$

Due to linearity, we thus have

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \frac{1}{2}|0,1\rangle - \frac{1}{2}|0,0\rangle + \frac{1}{2}|1,1\rangle - \frac{1}{2}|1,0\rangle.$$

One can easily see that this is exactly minus $H(|0\rangle) \otimes H(|1\rangle)$:

$$f(H(|0\rangle) \otimes H(|1\rangle)) = -H(|0\rangle) \otimes H(|1\rangle).$$

Thus, due to linearity, when we apply the Hadamard transformation once again, we get minus what we would get if we apply the Hadamard transformation to $H(|0\rangle) \otimes H(|1\rangle)$, i.e., $-|0,1\rangle$.

Case of $f(0) = 0$ and $f(1) = 1$. In this case,

$$f(|0,0\rangle) = |0,0\rangle, \quad f(|0,1\rangle) = |0,1\rangle, \quad f(|1,0\rangle) = |1,1\rangle, \quad f(|1,1\rangle) = |1,0\rangle,$$

and, thus,

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \frac{1}{2}|0,0\rangle - \frac{1}{2}|0,1\rangle + \frac{1}{2}|1,1\rangle - \frac{1}{2}|1,0\rangle =$$

$$\frac{1}{2}|0\rangle \otimes |0\rangle - \frac{1}{2}|0\rangle \otimes |1\rangle + \frac{1}{2}|1\rangle \otimes |1\rangle - \frac{1}{2}|1\rangle \otimes |0\rangle.$$

The first two terms have a common factor $|0\rangle$, the third and the fourth one have a common vector $|1\rangle$, so we have

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \frac{1}{\sqrt{2}}|0\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}|1\rangle \otimes \left(\frac{1}{\sqrt{2}}|1\rangle - \frac{0}{\sqrt{2}}|0\rangle\right).$$

This expression can be equivalently reformulated as

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \frac{1}{\sqrt{2}}|0\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) - \frac{1}{\sqrt{2}}|1\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{0}{\sqrt{2}}|1\rangle\right),$$

and thus, as

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right).$$

For both bits, what we have is $H|1\rangle$. Thus, when we apply the Hadamard transformation once again, we get $|1,1\rangle$.

Case of $f(0) = 1$ and $f(1) = 0$. In this case,

$$f(|0,0\rangle) = |0,1\rangle, \quad f(|0,1\rangle) = |0,0\rangle, \quad f(|1,0\rangle) = |1,0\rangle, \quad f(|1,1\rangle) = |1,1\rangle,$$

and, thus,

$$f(H(|0\rangle) \otimes H(|1\rangle)) = \frac{1}{2}|0,1\rangle - \frac{1}{2}|0,0\rangle + \frac{1}{2}|1,0\rangle - \frac{1}{2}|1,1\rangle.$$

We can see that this exactly minus what we got for the previous function, so when we apply the Hadamard transformation again, we will get minus what we got there, i.e., $-|1,1\rangle$.

Summarizing.

- When the function f is constant, we get either $|0,1\rangle$ or $-|0,1\rangle$. For both functions, if we measure the first bit, the result will be 0 with probability 1.
- When the function f is not constant, we get either $|1,1\rangle$ or $-|1,1\rangle$. For both functions, if we measure the first bit, the result will be 1 with probability 1.

So, by measuring the first bit, we can tell whether the function is constant or not. Thus, we arrive at the following algorithm:

Deutsch-Jozsa algorithm:

- we start with the state $|0,1\rangle = |0\rangle \otimes |1\rangle$;
- we apply the Hadamard transformation H to both bits;
- then, we apply f;
- after that, we again apply the Hadamard transformation to both bits;
- finally, we measure the first bit of the resulting 2-bit state:
  - if the first bit is 0, we conclude that the function f is constant;
  - if the first bit is 1, we conclude that the function f is not constant.

The above analysis shows that can thus indeed check whether the given function $f(x)$ is constant or not – and we can do it by applying the function f only once.

Discussion. We can do it since in non-quantum case, we could only apply f to 0 or 1, now we can apply it to a superposition of 0 and 1.

Of course, the above problem is a toy example, but – as we mentioned in the beginning of this paper – quantum computing is potentially efficient in more practical problems as well.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, and by an award "UTEP and Prudential Actuarial Science Academy and Pipeline Initiative" from Prudential Foundation.

References

[1] D. Deutsch and R. Jozsa, "Rapid solutions of problems by quantum computation", *Proceedings of the Royal Society of London, Ser. A*, 1992, Vol. 439, pp. 553–558.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search", *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996, pp. 212–219.

[3] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", *Physical Reviews Letters*, 1997, Vol. 79, No. 2, pp. 325–328.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, U.K., 2000.

[5] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994, pp. 124–134.