How to Use Quantum Computing to Check Which Inputs Are Relevant: A Proof That Deutsch-Jozsa Algorithm Is, In Effect, the Only Possibility

Vladik Kreinovich[1], Martine Ceberio[1], and Ricardo Alvarez[1]
[1]Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
vladik@utep.edu, mceberio@utep.edu, ralvarezlo@miners.utep.edu

Abstract

One of the main reasons why computations – in particular, engineering computations – take long is that, to be on the safe side, models take into account all possible affecting features, most of which turn out to be not really relevant for the corresponding physical problem. From this viewpoint, it is desirable to find out which inputs are relevant. In general, the problem of checking the input's relevancy is itself NP-hard, which means, crudely speaking, that no feasible algorithm can always solve it. Thus, it is desirable to speed up this checking as much as possible. One possible way to speed up such a checking is to use quantum computing, namely, to use the Deutsch-Josza algorithm. However, this algorithm is just *a* way to solve this problem, it is not clear whether a more efficient (or even different) quantum algorithm is possible for solving this problem. In this paper, we show that the Deutsch-Josza algorithm is, in effect, the only possible way to use quantum computing for checking which inputs are relevant.

1 Formulation of the Problem

Need to speed up data processing. In spite of the fantastic advances in computer technologies, in spite of the fact that computer processing now is several orders of magnitude faster than it was in the past, there are still many computational problems – in engineering and in other applications – whose solution takes too long a time, even on the fastest high performance computers.

What can we do about it – other than designing faster computers? The need to speed up data processing is a well-recognized need that motivates many efforts to design even faster computers.

But even for the existing computers, there is a usually a way to speed them up. This possibility comes from the fact that when we set up time-consuming simulations of real-life processes – be it simulations of atmospheric processes in meteorology or simulations of molecular interactions in biomedical applications – we do not a priori know which inputs are relevant and which are not. As a result, in our simulations, we use all the inputs that *may* be relevant. Because of this, a large number of these inputs are actually *not* relevant.

If we could figure out which inputs are relevant and which are not relevant, and limit ourselves only to relevant inputs, we would be able to save a lot of computation time which is now wasted on processing irrelevant inputs and thus, speed up the corresponding computations.

Need to check which inputs are relevant is well understood. This need to separate relevant from irrelevant inputs is well-recognized in physics (see, e.g., [2, 9]): it is an important skill of a physicist, enabling physicists to find solutions to complex physics-related equations much faster and much easier than mathematicians who do not have this skill. A classical example comes from General Relativity: the famous mathematician David Hilbert came with these equations practically at the same time as Einstein (his paper describing these equations was submitted only two weeks later than Einstein's), but equations is all Hilbert did, while Einstein, undeterred by the complexity of these nonlinear partial differential equations, also provided observable consequences – based on his ability to ignore what could be ignored in the corresponding physical situation and to consider only the most relevant inputs.

Checking which inputs are relevant is, in general, a difficult computational problem. In general, the problem of checking which inputs are relevant and which are not is a probably difficult computational problem. Even for the simple case when the inputs are 1-bit (boolean) variables $x_1, \ldots, x_n$, and the data processing algorithm consists of applying some propositional (boolean) operations ("and", "or", and "not") to these variables, the problem of checking whether these inputs are needed at all or the result is always false (0) is a known NP-hard problem (see, e.g., [7]): it is, in effect, the same problem as the known NP-hard problem of checking whether a given propositional formula $f(x_1, \ldots, x_n)$ can be satisfied, i.e., whether there exist values $x_1, \ldots, x_n$ that make it true.

Not only this problem of checking which inputs are relevant is difficult, it is the most difficult of all the problems: NP-hardness means that if we can solve this problem is feasible time, then we can solve *any* problem with easy-to-check solution in feasible time.

In practice, this means that algorithms for checking which inputs are relevant themselves require a lot of computation time. Thus, we need to speed up these algorithms as well.

How can we speed up checking of which inputs are relevant. In general, if the *existing* technology does not enable us to compute something sufficiently fast, a natural way to speed up computations it to use *new* technology, a technology that utilizes additional physical processes. Computers consist of many very small parts, and for such very small objects, many physical processes involve the use of quantum physics – the physics that describes micro-objects like molecules, atoms, elementary particles, etc. Indeed, the idea of using quantum effects in computing turned out to be very successful; see, e.g., [3, 4, 6, 8]. In particular, there exists an efficient quantum algorithm – Deutch-Josza algorithm – that checks whether a given bit is relevant for computations; see, e.g., [1, 5, 6].

Formulation of the problem. The challenge is that while the Deutsch-Josza algorithm is efficient, it is not clear whether this is the only possible quantum algorithm for solving this problem – maybe a better quantum algorithm is possible?

What we do in this paper. In this paper, for the simplest case of a 1-bit input, we show that the Deutsch-Josza algorithm is, in effect, the only possible quantum algorithm for solving this problem.

2 Deutsch-Jozsa Algorithm: Reminder

Quantum computing: a brief reminder. Let us first recall the main ideas behind quantum computing. In quantum physics, in addition to the usual states $s_1, \ldots, s_n$, it is also possible to form a *superposition*, i.e., a new state

$$s = a_1 s_1 + \ldots + a_n s_n,$$

where a_i are complex numbers for which $|a_1|^2 + \ldots + |a_n|^2 = 1$. If we apply, to this superposition state, the usual measurement procedure that checks which of the states s_i we are in, we will get the state s_i with probability $|a_i|^2$. These probabilities should add up to one – which explains the above restriction on the coefficients a_i.

In particular, for a usual 1-bit state (0 or 1), in addition to the traditional states – which in quantum physics are denoted by $|0\rangle$ and $|1\rangle$ – we can also have superposition states $a_0|0\rangle + a_1|1\rangle$. The corresponding quantum analogue of a bit is known as a *quantum bit*, or *qubit*, for short.

When the object of study consists of two independent parts, then in classical physics, the state of the object can be described by describing the states s_i and s'_j of each part. Similarly, in quantum physics, if we have two independent parts, one in a state $s = a_1 s_1 + \ldots + a_n s_n$ and another in a state $s' = a'_1 s'_1 + \ldots + a'_m s'_m$, then the whole object is in the state

$$(a_1 \cdot a'_1)|s_1 s'_1\rangle + (a_1 \cdot a'_2)|s_1 s'_2\rangle + \ldots + (a_n \cdot a'_m)|s_n s'_m\rangle.$$

This state is called a *tensor product* of the states s and s' and is usually denoted by $s \otimes s'$. In this paper, we will use the tensor product of two qubits.

For a 2-qubit state, if we measure the state of one of the qubits – e.g., the first one – then the state of the first bit changes to 0 or 1, and the resulting state of the 2-bit system is *normalized*: the corresponding coefficients are divided by a constant so that the sum of the squares of absolute values remain to be 1. For example, if we measure the value of the first bit in the state

$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle,$$

and the result of this measurement is 0, then we take the remaining state

$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle$$

and normalize it by multiplying it by $\sqrt{2}$; the resulting state is

$$\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|01\rangle.$$

We can also perform some linear transformations on the set of all possible quantum states – provided that these transformation preserve the equality $\sum\limits_{i=1}^{n} |a_i|^2 = 1$. This is equivalent to requiring that the states corresponding to orthogonal vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ (i.e., vectors for which the dot product $a \cdot b = \sum\limits_{i=1}^{n} a_i \cdot b_i^* = 0$, where b_i^* means complex conjugate) get transformed into orthogonal ones.

A typical example of such a transformation is a so-called *Hadamard transformation* H that transforms $|0\rangle$ into

$$H|0\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle$$

and $|1\rangle$ into

$$H|1\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle.$$

One can easily check that if we apply H twice, we get back the same state: $H(H|0\rangle) = |0\rangle$ and $H(H|1\rangle) = |1\rangle$.

The last thing we need to describe quantum computing is how functions are represented. The problem is that many useful functions are not reversible: e.g., if we know that $f(a, b) = a \,\&\, b$ is false, we cannot uniquely determine the values of a and b, they can be both false, or one of them can be false and another true. On the other hand, on the microlevel of quantum physics, all operations are reversible. So, to represent a function $f(x_1, \ldots, x_n)$ from bits to bits in quantum physics, we represent is a reversible transformation

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

where $a \oplus b$ is exclusive or, which is the same as addition modulo 2. Such transformations are reversible: indeed, if we apply the same transformation again, the first n bits do not change, while the last bit becomes

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

since for addition modulo 2, we always have $a \oplus a = 0$.

Now, we are ready the describe the Deutsch-Josza algorithm.

Deutsch-Josza algorithm: description. We are given a function $f(x)$ of one bit, i.e., a black box that transform a 2-bit state $|x, y\rangle$ into a new state $|x, y \oplus f(x)\rangle$. We want to check whether the input x is relevant, i.e., whether $f(0) \neq f(1)$. (Indeed, if $f(0) = f(1)$, then the result of applying the function $f(x)$ does not depend on the input, and the input is thus irrelevant.)

In non-quantum computing, the only way to use the black box for computing f is to apply this box either to 0 or to 1. Thus, to check whether $f(0) = f(10$, we need to call the algorithm f twice: once for $x = 0$ and the second time for $x = 1$. Since, as we have mentioned earlier, $f(x)$ may be a very complex time-consuming algorithm, the need to run it twice requires too much time. In turns out that in quantum computing, we can check the equality $f(0) = f(1)$ by calling the function f only once.

Here is how. First, a preliminary step: we start with a state $|01\rangle = |0\rangle \otimes |1\rangle$ and apply the Hadamard transformation to both bits. As a result, we get the following state:

$$H(|0\rangle) \otimes H(|1\rangle) = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)$$
$$= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle - \frac{1}{2}|11\rangle.$$

Then, we apply f to the resulting state, and then again apply the Hadamard transformation to both bits. After that, we measure the state of the first bit.

- It the resulting state of the first bit is 0, we conclude that the function $f(x)$ is constant, i.e., that the input is *not* relevant.
- If the resulting state of the first bit is 1, we conclude that the function $f(x)$ is not constant, i.e., that the input *is* relevant.

Deutsch-Josza algorithm: proof of correctness. To prove the algorithm's correctness, let us consider all four possible bit-to-bit functions $f(x)$.

When $f(0) = f(1) = 0$, then after applying the function f, we get $y' = y \oplus f(x) = y$. So the state does not change, and when we apply the Hadamard transform again, the state gets back to $|01\rangle$. So, the first bit is in 0 state.

When $f(0) = f(1) = 1$, then we get $y' = y \oplus f(x) = y \oplus 1$. Here 0 is changed to 1 and 1 is changed to 0. As a result the state of the 2-bit system changes to

$$\frac{1}{2}|01\rangle - \frac{1}{2}|00\rangle + \frac{1}{2}|11\rangle - \frac{1}{2}|10\rangle.$$

One can check that when we apply the Hadamard transform again, the state of the system changes to $-|01\rangle$. Here also, the first bit is in the 0 state.

When $f(x) = x$, the application of f leads to:

$$f(|00\rangle) = |00\rangle, f(|01\rangle) = |01\rangle, f(|10\rangle) = |11\rangle, \text{ and } f(|11\rangle) = |10\rangle.$$

Thus, the superposition state changes to

$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle - \frac{1}{2}|10\rangle.$$

When we apply the Hadamard transformation to this state, we get $|11\rangle$. Here, the first bit is in the 1 state.

Finally, let us consider the case when $f(x) = \neg x$, In this case,

$$f(|00\rangle) = |01\rangle, f(|01\rangle) = |00\rangle, f(|10\rangle) = |10\rangle, \text{ and } f(|11\rangle) = |11\rangle,$$

so the superposition changes to the following state:

$$\frac{1}{2}|01\rangle - \frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle.$$

When we apply the Hadamard transformation to this state, we get $-|11\rangle$, so the first bit is 1.

In all four cases, when the function is constant, the algorithm returns 0, and when the function is not constant, the algorithm returns 1. Thus, Deutsch-Josza algorithm indeed solves our original problem – and solves it in just one call to f instead of two.

3 Proof That It Is the Only Possibility

General scheme. We want an algorithm that calls f only once. Thus, we first perform some transformations, resulting in some 2-qubit state

$$a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle.$$

To this state, we apply the function f, and then we again apply some transformations to each bit. After all this, we expect to get 0 if the input is irrelevant and 1 if it is relevant.

In the case of $f(x) = 0$, the application of f does not change the state, i.e., we get the same state

$$(a_{00}|0\rangle + a_{10}|1\rangle) \otimes |0\rangle) + (a_{01}|0\rangle + a_{11}|1\rangle) \otimes |1\rangle).$$

No matter what we do with the second bit, the first bit needs to get into the 0 state, so we have

$$a_{00}|0\rangle + a_{10}|1\rangle \to |0\rangle \text{ and } a_{01}|0\rangle + a_{11}|1\rangle \to |0\rangle.$$

All quantum transformations are reversible. Thus, the fact that these two states of the first bit get transformed into the same state $|0\rangle$ means that these states are identical, i.e., that

$$a_{00}|0\rangle + a_{10}|1\rangle = C \cdot (a_{01}|0\rangle + a_{11}|1\rangle)$$

for some normalizing coefficient C. Thus, we conclude that

$$\frac{a_{01}}{a_{00}} = \frac{a_{11}}{a_{10}}. \tag{1}$$

For $f(x) = x$, applying the function f leads to the state

$$a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|11\rangle + a_{11}|10\rangle =$$

$$(a_{00}|0\rangle + a_{11}|1\rangle) \otimes |0\rangle) + (a_{01}|0\rangle + a_{10}|1\rangle) \otimes |1\rangle).$$

Here, the first bit needs to go into the 1 state, i.e., we have

$$a_{00}|0\rangle + a_{11}|1\rangle \to |1\rangle \text{ and } a_{01}|0\rangle + a_{10}|1\rangle \to |1\rangle.$$

Here also, the fact that these two states of the first bit get transformed into the same state $|1\rangle$ means that these states are identical, i.e., that

$$a_{00}|0\rangle + a_{11}|1\rangle = C \cdot (a_{01}|0\rangle + a_{10}|1\rangle)$$

for some normalizing coefficient C. Thus, we conclude that

$$\frac{a_{01}}{a_{00}} = \frac{a_{10}}{a_{11}}. \tag{2}$$

Comparing equalities (1) and (2), we conclude that for $x \stackrel{\text{def}}{=} \dfrac{a_{11}}{a_{10}}$, we have $x = 1/x$, hence $x^2 = 1$ and so, $x = \pm 1$.

The ratio x cannot be equal to 1, since then we would have $a_{10} = a_{11}$, but we have

$$a_{00}|0\rangle + a_{10}|1\rangle \to |0\rangle \text{ and } a_{00}|0\rangle + a_{11}|1\rangle \to |1\rangle,$$

so $a_{10} \neq a_{11}$. Thus, we must have $x = -1$.

So, we have

$$a_{00}|0\rangle + a_{10}|1\rangle \to |0\rangle \text{ and } a_{00}|0\rangle - a_{10}|1\rangle \to |1\rangle.$$

The states $|0\rangle$ and $|1\rangle$ are orthogonal. So, by the properties of quantum transformations, the states $a_{00}|0\rangle + a_{10}|1\rangle$ and $a_{00}|0\rangle - a_{10}|1\rangle$ must also be orthogonal, i.e., we must have

$$a_{00} \cdot a_{00}^* - a_{10} \cdot a_{01}^* = |a_{00}|^2 - |a_{10}|^2 = 0.$$

Thus, $|a_{10}| = |a_{00}|$, and $a_{10} = \exp(\mathrm{i} \cdot \varphi) \cdot a_{00}$ for some angle φ.

From the fact that the probabilities should add up to 1, we conclude that $4|a_{00}|^2 = 1$, hence $|a_{00}| = 1/2$. In quantum physics, states differing only by

a complex factor whose absolute value is 1 are considered identical. Thus, we can safely assume that $a_{00} = 1/2$. Hence, $a_{01} = -a_{00} = -1/2$. Then, $a_{10} = \exp(\mathrm{i} \cdot \varphi) \cdot (1/2)$ and $a_{11} = -a_{10}$. So, for the original 2-bit state, we get the expression

$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{\exp(\mathrm{i} \cdot \varphi)}{\sqrt{2}}|10\rangle - \frac{\exp(\mathrm{i} \cdot \varphi)}{2}|11\rangle.$$

This is almost the same as for the original Detsch-Josza algorithm: the only minor difference is the factor $\exp(\mathrm{i} \cdot \varphi)$ that does not affect any probabilities.

Modulo this minor difference, the Deutsch-Josza algorithm is indeed, the only possible algorithm for solving the problem. Our result has been proven.

Acknowledgments

This work was supported in part by the National Science Foundation via grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science) and HRD-1242122 (Cyber-ShARE Center of Excellence).

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] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.

[3] 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.

[4] 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.

[5] O. Kosheleva and V. Kreinovich, "How to introduce technical details of quantum computing in a theory of computation class: using the basic case of the Deutsch-Jozsa algorithm", *International Journal of Computing and Optimization*, 2016, Vol. 3, No. 1, pp. 83–91.

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

[7] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[8] 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.

[9] K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2017.