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, Martine Ceberio, and Ricardo Alvarez

Department of Computer Science
University of Texas at El Paso, El Paso, Texas 79968, USA,
vladik@utep.edu, mceberio@utep.edu, ralvarezlo@miners.utep.edu

1. Need to Speed up Data Processing

- There have been fantastic advances in computer technologies.
- Computer processing now is several orders of magnitude faster than it was in the past.
- However, there are still many computational problems – in engineering and in other applications – for which:
  - solution takes too long a time,
  - even on the fastest high performance computers.

2. What Can We Do About It – Other Than Designing Faster Computers?

- The need to speed up data processing is well-recognized.
- It 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 of biomedical molecular interactions
  - 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.

3. What Can We Do About It (cont-d)

- 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.

4. Need to Check Which Inputs Are Relevant Is Well Understood

- This need to separate relevant from irrelevant inputs is well-recognized in physics:
  - physicists can 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 up with these equations at the same time as Einstein.
- Hilbert's paper describing these equations was submitted only two weeks later than Einstein's.
- However, equations is all Hilbert did.
- Einstein also provided observable consequences.
- He did it by considering only the most relevant inputs.

5. 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 provably difficult.
- Let's consider the simple case when:
  - the inputs are 1-bit variables $x_1, \ldots, x_n$, and
  - the data processing algorithm consists of applying boolean operations ("and", "or", and "not").
- The problem of checking whether these inputs are needed or the result is always false (0) is NP-hard.
- 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.

6. It's a Difficult Problem (cont-d)

- Not only this problem is difficult, it is the most difficult of all the problems.
- Indeed, 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.
- This means that algorithms for checking which inputs are relevant themselves require a lot of time.
- Thus, we need to speed up these algorithms as well.

7. How Can We Speed Up Checking Which Inputs Are Relevant

- In general:
  - if the *existing* technology does not enable us to compute something sufficiently fast,
  - a natural idea is to use *new* technology, new physical processes.
- Computers consist of many very small parts.
- For such very small objects, physical processes involve the use of quantum physics.
- The idea of using quantum effects in computing turned out to be very successful.
- E.g., quantum Deutch-Josza algorithm checks whether a given bit is relevant for computations.

8. Formulation of the Problem

- The Deutsch-Josza algorithm is efficient.
- However, it is not clear whether this is the only possible quantum algorithm for solving this problem.
- Maybe a better quantum algorithm is possible?
- In this talk:
  - for the simplest case of a 1-bit input,
  - we show that the Deutsch-Josza algorithm is the only possible quantum algorithm for this problem.

9. Towards Formulation of the Problem in Exact Terms

- In the 1-bit case, we are given a function $f(x)$ of one bit. In the non-quantum case, this means that:
  - we are given a black box that
  - transforms a 1-bit state x into a new 1-bit state $f(x)$.
- We want to check whether the input x is relevant.
- If $f(0) = f(1)$, then the result of applying the function $f(x)$ does not depend on the input, so, x is irrelevant.
- On the other hand, if $f(0) \neq f(1)$, the value $f(x)$ depends on the input, so x is relevant.
- So, checking whether the input is relevant means checking whether $f(0) \neq f(1)$.

10. How This Problem Can Be Solved in the Case of Non-Quantum Computing

- 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 $f(0) = f(1)$, we need to call the algorithm f *twice*: for $x = 0$ and for $x = 1$.
- $f(x)$ may be a very complex time-consuming algorithm.
- So, the need to run it twice requires too much time.
- In quantum computing, we can check the equality $f(0) = f(1)$ by calling the function f only once.

11. States in Quantum Mechanics

- Let us 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 $s = a_1 s_1 + \ldots + a_n s_n$.
- Here, a_i are complex numbers for which

$$|a_1|^2 + \ldots + |a_n|^2 = 1.$$

- If we apply, to the above 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.

12. The Notion of a Qubit

- 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.

13. States of Multi-Particle/Multi-Bit Systems

- In classical physics:
  - when the object of study consists of two independent parts,
  - then 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, in states

$$s = a_1 s_1 + \ldots + a_n s_n \text{ and } s' = a'_1 s'_1 + \ldots + a'_m s'_m,$$

  - 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 $s \otimes s'$.

14. Measuring 2-Qubit States

- 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 coefficients are divided by a constant so that the sum of the squares of absolute values remain 1.
- *Example:* we measure the 1st bit in the state
$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle.$$
- Then, the result is 0, then the remaining state is
$$\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle.$$
- We normalize it by multiplying it by $\sqrt{2}$, this leads to:
$$\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|01\rangle.$$

15. Transformations of Quantum States: General Idea

- We can also perform some linear transformations on the set of all possible quantum states.
- The main requirement is that these transformation preserve the equality $\sum_{i=1}^{n} |a_i|^2 = 1$.
- This is equivalent to requiring that orthogonal states $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ get transformed into orthogonal ones.
- Here, orthogonal means that $a \cdot b = \sum_{i=1}^{n} a_i \cdot b_i^* = 0$, where b_i^* means complex conjugate.

16. Example: Hadamard Transformation H

- H transforms $|0\rangle$ into $H|0\rangle = \dfrac{1}{\sqrt{2}} \cdot |0\rangle + \dfrac{1}{\sqrt{2}} \cdot |1\rangle$ and $|1\rangle$ into $H|1\rangle = \dfrac{1}{\sqrt{2}} \cdot |0\rangle - \dfrac{1}{\sqrt{2}} \cdot |1\rangle$.
- One can easily check that if we apply H twice, we get back the same state:

$$H(H|0\rangle) = |0\rangle \text{ and } H(H|1\rangle) = |1\rangle.$$

17. Functions in Quantum Computing

- 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, in quantum computing, we represent a bit-valued function $f(x_1,\ldots,x_n)$ by a reversible transformation:

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

- Here $a \oplus b$ is exclusive or $\equiv$ addition modulo 2.

18. Functions in Quantum Computing (cont-d)

- Such transformations are reversible: indeed, if we apply the same transformation again,
  - the first n bits do not change, while
  - the last bit becomes

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

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

  - indeed, for addition modulo 2, we always have

$$a \oplus a = 0.$$

- Now, we are ready the describe the Deutsch-Josza quantum algorithm for solving this problem.

19. Deutch-Josza Algorithm: Reminder

- 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 $f(0) \neq f(1)$.
- In non-quantum computing, we need at least two calls to f to check whether $f(0) = f(1)$.
- The quantum Deutsch-Josza algorithm can check whether $f(0) = f(1)$ in one call to f.

20. Preliminary Step

- We start with a state $|01\rangle = |0\rangle \otimes |1\rangle$ and we apply the Hadamard transformation to both bits.
- As a result, we get the following state:

$$H(|0\rangle) \otimes H(|1\rangle) =$$

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

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

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 21 of 34
Go Back
Full Screen
Close
Quit

21. The Remaining Step

- After the preliminary step, we do the following:
  - we apply f to the quantum state resulting from the preliminary step;
  - then, we again apply the Hadamard transformation to both bits;
  - after that, we measure the state of the first bit.
- Based on the result of this measurement, we inform the user where the given function f is a constant:
  - It the resulting state of the first bit is 0, we conclude that the function $f(x)$ is constant.
  - If the resulting state of the first bit is 1, we conclude that the function $f(x)$ is not constant.

22. Proof of Correctness: General Idea

- To prove the algorithm's correctness, let us consider all possible bit-to-bit functions $f(x)$.
- We have two possible values for $f(0)$, for each of which we have two possible values of $f(1)$.
- Thus, overall, we have four possible cases:
  - the case when $f(0) = 0$ and $f(1) = 0$;
  - the case when $f(0) = 0$ and $f(1) = 1$, i.e., when $f(x) = x$ for all x;
  - the case when $f(0) = 1$ and $f(1) = 0$, i.e., when $f(x) = \neg x$ for all x; and
  - the case when $f(0) = 1$ and $f(1) = 1$.
- Let us consider these four cases one by one.

23. Case When $f(0) = f(1) = 0$

- In this case, after applying the function f, we get

$$y' = y \oplus f(x) = y.$$

- So the state does not change.
- When we apply the Hadamard transform again, the state gets back to $|01\rangle$.
- So, the first bit is in 0 state.

24. Case When $f(0) = f(1) = 1$

- In this case, 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.

25. Case When $f(x) = x$

- In this case, 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.

26. 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.

27. Summarizing

- 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 – in just 1 call to f instead of 2.
- Let us now prove that no other quantum algorithm can solve this problem.

28. 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

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

29. General Scheme (cont-d)

- 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 for some normalizing coefficient C:

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

- Thus, we conclude that

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

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

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

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

30. General Scheme (cont-d)

- Here, the first bit needs to go into the 1 state, i.e.:

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

- Here also, the two states of the first bit get transformed into the same state $|1\rangle$.
- This means that these states are identical, i.e., that for some C, $a_{00}|0\rangle + a_{11}|1\rangle = C \cdot (a_{01}|0\rangle + a_{10}|1\rangle)$.
- Thus, $\dfrac{a_{01}}{a_{00}} = \dfrac{a_{10}}{a_{11}}$.
- Comparing the two equalities, 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

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

31. General Scheme (cont-d)

- So, $a_{10} \neq a_{11}$; thus, we must have $x = -1$.
- So, $a_{00}|0\rangle + a_{10}|1\rangle \to |0\rangle$ and $a_{00}|0\rangle - a_{10}|1\rangle \to |1\rangle$.
- The states $|0\rangle$ and $|1\rangle$ are orthogonal: $|0\rangle \perp |1\rangle$.
- So, by the properties of quantum transformations, the states $a_{00}|0\rangle + a_{10}|1\rangle \perp a_{00}|0\rangle - a_{10}|1\rangle$, i.e.:

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

- So, $|a_{10}| = |a_{00}|$, and $a_{10} = \exp(\mathrm{i} \cdot \varphi) \cdot a_{00}$ for some 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 with absolute value 1 are considered identical.
- Thus, we can safely assume that $a_{00} = 1/2$. Hence, $a_{01} = -a_{00} = -1/2$.

32. General Scheme (cont-d)

- 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 following 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.*
- Our result has been proven.

33. 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).