Final Project: Quantum Computing and the Deutsch-Jozsa Algorithm

CS110: Computation: Solving Problems with Algorithms

Yoav Rabinovich

April 2018

---

1 Quantum Computing

Throughout the semester, we have learned how to think about algorithms and the data structures they define in terms of their structure and their operational complexity. We have limited ourselves, understandably, to algorithms involving that on a fundamental level operate on binary bits. However, a great deal of thought and effort was put into the growing field of quantum computing, which seeks to expand our idea of a bit and allow for more diverse operations.

Quantum computing has been proven to be able to use the properties of quantum states to offer substantial reductions in complexity for certain tasks. Most notably, Shor's algorithm can find the prime factors of any number in polynomial log N time (specifically *O((log N)2(log log N)(log log log N))*) while the best classical alternatives run in sub-exponential time. The problem of fast prime factorization is crucial for some cryptography applications, a single factor that drives much of the motivation behind quantum computing today. The Deutsch-Jozsa algorithm, among the first proof of substantial complexity reduction in quantum computing is a milestone in its own right, and a simpler example which we will explore in this paper.

2 Qubits and Quantum Operations

Quantum computing operates on **Qubits** - information structures represented by fundamental particles, that can exist either in defined orthogonal states analogous to the 1s and 0s of traditional binary computing, or in a **superposition** or such states, essentially having a set probability to be measured of in either state, rather than a definite configuration. This probability is embedded in a propertay called **probability amplitude** which can have negative or even complex values, and from it we infer a probability. This departure from strict probability lets our states interfere constructively or destructively. For example, we can represent a state with a simple weighted sum using Dirac notation:

$$|x\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle$$

State x is a superposition of states 0 and 1, with amplitudes $\frac{1}{\sqrt{2}}$ and $-\frac{i}{\sqrt{2}}$ respectively. To find the probability to measure each base state, we square the absolute values of the amplitudes and attain a uniform distribution with a probability of $\frac{1}{2}$ for each state to be measured, even though the amplitude for 1 was negative and imaginary.

States can also be thought of more abstractly as systems of several Qubits, of which we don't know the states but we know some relation between the states. Those are called **entangled states**. For example, we might know that two Qubits are in opposite states from one another. We could label the combined entangled state as:

$$|x\rangle \otimes |y\rangle = |x, y\rangle = \frac{1}{\sqrt{2}}(|0, 1\rangle + |1, 0\rangle)$$

Where the probability to measure any opposite state (01 or 10) is $\frac{1}{2}$ and 0 to measure any identical state (00 or 11).

Working with Qubits we can then incorporate additional basic operations, such as measurement operators, operators that manage the superposition configuration of a Qubit (rotation operators), operators that entangle bits and so on. Operators in quantum mechanics can be interpreted many ways from actions taken on particle to abstract linear algebra matrices acting on vector states, but in our context they are essentially arithmetic operations that affect Qubits. But inevitably, a measurement of the system must be made, and superposition states will collapse into a single measurement, erasing information of the previous superposition. Therefore, Quantum computing algorithms need to find ways to use the available operators to create states that collapse into a solution encoded in base states, which can be a challenge.

3 The Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm is a deterministic algorithm, and one of the first ones to be proven to be an exponential improvement in complexity over classical solutions. However, the problem it solves is very narrow and fairly useless.

The algorithm evaluates any function that takes an array of binary numbers, and returns an array of binary numbers that is guaranteed to be either:

- **Constant**: All 1s or all 0s; or
- **Balanced**: Where the amount of 1s in the array is equal to the amount of 0s.

The algorithm then determines whether the function returns constant or

balanced outputs.

With classical computing, the problem is solved by comparing the two strings, in the worst case having to evaluate the function $2^{n-1}+1$ times, resulting in $O(2^n)$ exponential complexity, if the first half of the output is constant before a change. A smart programmer might randomize the output to minimize that chance, and get a solution with constant evaluations of f with very high probability, but the Deutsch-Jozsa algorithm achieves this deterministically.

We describe the algorithm first by considering the simplest case of $n = 2$. We don't know the effect of our function f on a bit, but we can define a function to test it in the quantum context as:

$$G(|a, b\rangle) = |a, b + f(a)\rangle$$

The algorithm starts with arranging two Qubits into the states:

$$|0\rangle \otimes |1\rangle = |0, 1\rangle$$

For the second step, we then apply a simple rotation operator, named the Hadamard operator (or Hadamard gate), which is defined simply as:

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

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

This basically takes pure states and puts them into superpositions with an equal probability to be measured as each base state. However, notice that the resulting states are not identical, since the amplitudes differ in sign. There is still information left about the original state. Our system is

now in a transformed state:

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

Or in the entangled state representation:

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

The third step is applying our function G to the state, inserting into the system information about our f function, using modulus 2, since we are dealing with two base vectors. We do not know the affect of f but we can see for example that if $f(0) = 1$, our term would gain a negative sign, and if $f(0) = 0$ nothing would change. We can write the general solution as:

$$G(\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)) = \left[(-1)^{f(0)}\,|0\rangle\,(|0\rangle - |1\rangle) + (-1)^{f(1)}\,|0\rangle\,(|0\rangle - |1\rangle)\right] \tag{3}$$

$$= (\frac{1}{2}\,|0\rangle + (-1)^{f(0)+f(1)}\,|1\rangle)(|0\rangle - |1\rangle) \tag{4}$$

We can now discard the second Qubit, represented in the second brackets, since we have grouped all the information required in one Qubit. The fourth and penultimate step is applying the Hadamard operator to our state one final time, achieving an interesting superposition state:

$$H(\frac{1}{2}\,|0\rangle + (-1)^{f(0)+f(1)}\,|1\rangle) = \frac{1}{2}\left[(1+(-1)^{f(0)+f(1)})\,|0\rangle + (1-(-1)^{f(0)+f(1)})\,|1\rangle\,\right]$$

This state is interesting becasue each of the probability amplitudes for measuring base states is reliant on the affect of f on Qubits in both base states, which allows us to make use of destructive and constructive interfer-

ence: We know a constant function would have $f(0) = f(1)$, and so when measuring the state with such a function, the $(1-(-1)^{f(0)+f(1)})$ term would disappear, ensuring that we always measure $|0\rangle$. Otherwise, for a balanced function $f(0) \neq f(1)$, the $(1-(-1)^{f(0)+f(1)})$ term would cancel, ensuring we measure $|1\rangle$.

We have successfully encoded the property we were looking for onto the two state basis, achieving a deterministic result. This algorithm can handle n of any size, simply by considering states in the form of:

$$H(|0\rangle^{\otimes n}|1\rangle) = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle - |1\rangle)$$

In both cases, the function needs to be evaluated once to reach a definite solution, an exponential improvement on classical equivalents.

4 Python Implementation

An implementation of a quantum computing algorithm in python wouldn't make much sense, for one because the complexity gain is by definition impossible to achieve in classical computing systems, and secondly because qubits and their most basic operations are not defined in the context of python. If I were required to do simulate the system from scratch, I would use the linear algebraic approach to quantum mechanics: I'd represent Qubits as vectors comprised of their amplitudes in the 0-1 space of states, and operations as transformation matrices. However, I have found a quantum mechanics module in the SymPy library, with a simulation implementation of Qubits and the Hadamard gate. A simple implementation of the Deutsch-Jozsa algorithm using SymPy follows:

```
import sympy.physics.quantum
#This function takes a function that itself takes a binary
    array of length n and outputs either a constant or a
    balanced binary array, determines which one it is and
    returns 0 for constant and 1 for balanced.
def DJA(n,func):
  #step one: prepare Qubit array
  placeholder=[0 for i in range(n-1)]
    placeholer.append(1)
    q=Qubit(placeholder)
    #step two: Hadamard gate
    q=gate.HadamardGate(0)*HadamardGate(1)*q
    #step three: G function
    for bit in q:
      bit[1]=bit[1]+func(bit[0])
    #step four: second Hadamard gate
    q=gate.HadamardGate(0)*HadamardGate(1)*q
    #step five: measurement
    return qubit.measure_all_oneshot(q)
```