CSE 599d - Quantum Computing
The Recursive and Nonrecursive Bernstein-Vazirani Algorithm

Dave Bacon
Department of Computer Science & Engineering, University of Washington

We have seen in the Deutsch-Jozsa problem that a quantum computer could be used to solve a problem in a single query which required an exponential number of queries on a classical computer in an *exact* model where no probability of failure was allowed. However, when we allowed a bounded error in our algorithm, then this advantage disappeared. After Deutsch and Jozsa introduced this problem, Bernstein and Vazirani took a deep look at quantum computers and produced a problem which showed a superpolynomial separation between quantum and classical query complexity. While Deutsch and Jozsa's problem gave us tantalizing hints that there would be a difference between quantum and classical query complexity, Bernstein and Vazirani made this hint real and showed, for the first time, that quantum computers could do things that their classical brethren could not!

The Bernstein-Vazirani paper is also important for many other reasons, among them showing that the notion of accuracy we used in our discussions of universal quantum computer adds and showing that bounded error quantum computing is in the complexity class PSPACE. But we will focus on the query complexity results of this important paper.

I. NONRECURSIVE BERNSTEIN-VAZIRANI ALGORITHM

In the Bernstein-Vazirani problem, we are given a n bit function $f : \{0,1\}^n \to \{0,1\}$ which outputs a single bit. This function is guaranteed to be of the form $f_s(x) = x \cdot s$ where s is an unknown n bit string and $x \cdot s = x_1 s_1 + x_2 s_2 + \cdots + x_n s_n \bmod 2$. The goal of the Bernstein-Vazirani problem is to find the unknown string s.

What is the classical query complexity of this problem? First suppose we are working in the exact query complexity model. Then a single query to the function can retrieve at most one bit of information about s (because f outputs only one bit.) Thus the exact query complexity must be at least n. In fact we can easily see that there is an algorithm which does in exactly n queries by querying with bitstrings which have the ith 1 and the other digits 0: this reveals the values of s_i.

What about if we allow bounded error? Here again we can use the fact that the function only outputs a single bit. Suppose that we could query this function in a probabilistic fashion and learn all of the bits of s in less that n bits with some probability of failure that is bounded below $1/2$. Then we would be able to use this function to compress an n bit string into less than n bits and use this to communicate these n bits, with high probability, through a channel. If we use the rigorous definition of information, we then essentially obtain that a bounded error algorithm will also need $\Omega(n)$ queries. Another way to prove this is to use Yao principle. We'll talk about this approach later.

What about a quantum algorithm for the Bersntein-Vazirani problem. We implement the function in our traditional reversible logic way, i.e. we have a quantum oracle which implements the unitary gate

$$U_s = \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}} |x\rangle\langle x| \otimes |y \oplus (s \cdot x)\rangle\langle y|. \tag{1}$$

How do we use this oracle to find s with less than n queries of this gate? First note that using the phase kickback trick and inputing a superposition over all possible input bit strings (just like we did in the Deutsch-Jozsa algorithm), we can create the state

$$|\psi_s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f_s(x)}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot s}|x\rangle \tag{2}$$

What is special about these states? They are all orthogonal! Let's show this:

$$\begin{aligned}\langle\psi_s|\psi_t\rangle &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot s}\langle x| \sum_{y\in\{0,1\}^n} (-1)^{y\cdot t}|y\rangle = \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot s+y\cdot t}\langle x|y\rangle \\ &= \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot s+y\cdot t}\delta_{x,y} = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot s+x\cdot t}\end{aligned} \tag{3}$$

Now $(-1)^{x\cdot s+x\cdot t} = (-1)^{x\cdot s+x\cdot t \bmod 2}$ and $x\cdot s+x\cdot t \bmod 2 = x_1s_1+\cdots x_ns_n+x_1t_1+\cdots+x_nt_n \bmod 2 = x\cdot(s+t) \bmod 2$ where $(s+t)$ is the bitwise addition of the two strings modulo 2. Thus we find that

$$\langle\psi_s|\psi_t\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot(s+t)} \tag{4}$$

Now consider the sum for fixed k,

$$\begin{aligned}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot k} &= \sum_{x\in\{0,1\}^n}(-1)^{x_1k_1}(-1)^{x_2k_2}\cdots(-1)^{x_nk_n}\\ &= \left(\sum_{x_1\in\{0,1\}}(-1)^{x_1k_1}\right)\left(\sum_{x_2\in\{0,1\}}(-1)^{x_2k_2}\right)\cdots\left(\sum_{x_n\in\{0,1\}}(-1)^{x_nk_n}\right)\\ &= 2\delta_{k_1,0}2\delta_{k_2,0}\ldots 2\delta_{k_n,0} = 2^n\delta_{k,0}\end{aligned} \tag{5}$$

This implies that $\langle\psi_s|\psi_t\rangle = \delta_{(s+t),0}$, or, since this addition is done bitwise modulo 2,

$$\langle\psi_s|\psi_t\rangle = \delta_{s,t} \tag{6}$$

In other words these states are orthogonal.

Now what does this mean for the possibility of a quantum algorithm? Well an orthonormal set of vectors forms a basis and we can "measure in this basis." This amounts to performing a unitary and then measuring in the computational basis, from which we should be able to extract the string s. What is this transform? Well you've already seen it. It is the n qubit Hadamard transform:

$$H^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x,y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle\langle x| \tag{7}$$

Notice that we can express this as

$$H^{\otimes n} = \sum_{y\in\{0,1\}^n}|y\rangle\langle\psi_y| \tag{8}$$

Lets apply $H^{\otimes n}$ to $|\psi_s\rangle$:

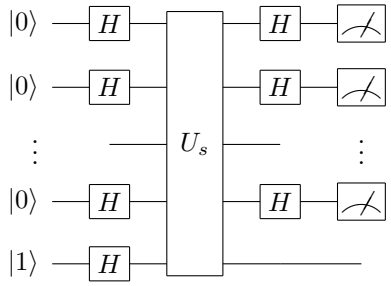
$$H^{\otimes n}|\psi_s\rangle = \sum_{y\in\{0,1\}^n}|y\rangle\langle\psi_y|\psi_s\rangle \tag{9}$$

which is equal to

$$H^{\otimes n}|\psi_s\rangle = |s\rangle \tag{10}$$

by using the orthogonality of the $|\psi_s\rangle$ states.

Thus we have found an algorithm for identifying s using a *single* quantum query. The circuit to do this is exactly the same circuit as in the Deutsch-Jozsa problem.



(11)

With certainty the outcome of the measurements will be the bits of the string s.

II. RECURSIVE BERNSTEIN-VAZIRANI ALGORITHM

In the Bernstein-Vazirani (nonrecursive version) we have a separation between the bounded error classical model, which requires $\Omega(n)$ queries, and the quantum model which can be solved using only one query (and is exact.) Now this is nice, but the problem Bernstein and Vazirani consider is *classically tractable*. By this we mean that the hardness of the classical problem scales only polynomial in the problem size. What we'd really like to see is a separation between a problem which is classically intractable, but which can be solved on a quantum computer using polynomial resources. The interest in the Bernstein-Vazirani algorithm we described above (the nonrecursive version) is that Bernstein and Vazirani where able to bootstrap it into a new problem, the recursive Bernstein-Vazirani problem where just such a separation is possible. Here we will discuss this algorithm.

In the nonrecursive Bernstein-Vazirani problem, we were given access to a function $f_s(x) = s \cdot x$ and our goal was to find s. The first thing we do in constructing the recursive Bernstein-Vazirani problem is to modify this problem slightly. Instead of requiring that we find the unknown s, we instead require that we find some function $g : \{0,1\}^n \to \{0,1\}$ on s, $g(s)$. This function should be easy to compute, but also represent a "hard-core" bit of s. An example and one which we will use throughout our discussion is $g_{mod3}(x)$ which is equal to 0 if $|x| \bmod 3 = 0$ and is 1 otherwise and $|x|$ is the Hamming weight of x (the number of ones in the bitstring.) Notice that at this point we are still querying the function $f_s(x) = x \cdot s$, but we now require that instead of answering s, we answer $g(s)$.

Now let's show how to perform the first level of recursion for the Bernstein-Vazirani problem. For me it is easiest to describe the oracle we are querying first (I'm a hands on type of person.) The oracle is now two n strings, lets call them $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$. The function we are querying takes this input and outputs $s_x \cdot y$, i.e. the function is $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and is given by $f(x,y) = s_x \cdot y$. Now what is s_x here? Well the s_x are just 2^n different bit strings, labelled by $x \in \{0,1\}^n$. But they have one important property. When we compute $g(s_x)$ then we require that these 2^n bits satisfy $g(s_x) = x \cdot s$ for some unknown s. The two level Bernstein-Vazirani problem is then to identify $g(s)$.

Lets try to think about why this problem is harder than the nonrecursive Bernstein-Vazirani problem. When we query $f(x,y)$ we get output $s_x \cdot y$. Now we already know that it will take some work to identify some s_x from this input for a fixed x (because this is the original nonrecursive problem.) But then even when we find s_x for some fixed x, we can use it to calculate $g(s_x)$. But now these $g(s_x)$s themselves for different values of x are part of a nonrecursive Bernstein-Vazirani problem! So you can see that by recursing this way the problem seems to get harder.

We can recurse the Bernstein-Vazirani problem in this manner k times. It is again useful to have in mind exactly what the function we are querying computes. At the kth level the function takes as input k n bit strings, x_i and computes $f(x_1, x_2, \ldots, x_k) = x_k \cdot s_{x_1,x_2,\ldots,x_{k-1}}$. The important part now is the secret string $s_{x_1,x_2,\ldots,x_{k-1}} \in \{0,1\}^n$. This secret string is guaranteed to produce another lower level problem: $g(s_{x_1,x_2,\ldots,x_{k-1}}) = x_{k-1} \cdot s_{x_1,x_2,\ldots,x_{k-2}}$ where $s_{x_1,x_2,\ldots,x_{k-2}} \in \{0,1\}^n$. This continues all the way down: $g(s_{x_1,x_2,\ldots,x_j}) = x_j \cdot s_{x_1,x_2,\ldots,x_{j-1}}$. At the final level, we will have $g(s_{x_1}) = s \cdot x_1$. Our goal is to find $g(s)$.

Now what is the classical query complexity of this problem? Well by querying $f(x_1, x_2, \ldots, x_k)$ it will take us n times to identify $s_{x_1,x_2,\ldots,x_{k-1}}$. But then we will need to compute g of these secret strings another n times to find $s_{x_1,x_2,\ldots,x_{k-2}}$ Continuing onward we see that the query complexity classically must be at least $\Omega(n^k)$.

How do we solve this problem on a quantum computer? Well lets think about the level two problem. We are given access to a unitary which computes, in our standard reversible manner, $f(x_1, x_2) = s_{x_1} \cdot x_2$. Certainly using the recursive Bernstein-Vazirani problem we can use this to find s_{x_1} for a fixed s_{x_1}. But the tricky part of the recursive Bernstein-Vazirani problem is that when we take these s_{x_1}s and apply g to them we get a new Bernstein-Vazirani problem. Somehow we need to solve this new problem. This implies that we somehow have to use the quantum Bernstein-Vazirani problem over all x_1s as well.

Let's take a stab at how we might want to do this. The level two Bernstein-Vazirani oracle is

$$U = \sum_{x_1,x_2 \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x_1\rangle\langle x_1| \otimes |x_2\rangle\langle x_2| \otimes |y \oplus (s_{x_1} \cdot x_2)\rangle\langle y| \tag{12}$$

Suppose that we use the phase kickback trick on the y register and input a superposition over all possible bitstrings on the x_1 and x_2 registers. Then this produces that state

$$\frac{1}{2^n} \sum_{x_1,x_2 \in \{0,1\}^n} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle \otimes |x_2\rangle \otimes |-\rangle \tag{13}$$

Now if we perform the n qubit Hadamard on the second register, we see that it will transform this state to

$$\frac{1}{\sqrt{2^n}} \sum_{x_1 \in \{0,1\}^n} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |-\rangle. \tag{14}$$

But this is rather problematic. We know that applying g to s_{x_1} makes a new Bernstein-Vazirani problem. It is always possible to implement g in the reversible manner we've described previously. This means that we have access to a circuit which corresponds to the unitary

$$G = \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x\rangle\langle x| \otimes |g(x)\rangle \oplus y\rangle\langle y| \tag{15}$$

If we attach an extra ancilla qubit in the state $|0\rangle$, this means that we could use this unitary to turn our state into

$$\frac{1}{\sqrt{2^n}} \sum_{x_1 \in \{0,1\}^n} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |-\rangle \otimes |g(s_{x_1})\rangle. \tag{16}$$

But now look what's happened. We know that $g(s_{x_1})$ is a new Bernstein-Vazirani problem. But certainly if we are going to solve this new problem quantum mechanically needed to use the phase kickback trick, i.e. input $|-\rangle$ into the ancilla instead of $|0\rangle$. If we do this, then the state is

$$\frac{1}{\sqrt{2^n}} \sum_{x_1 \in \{0,1\}^n} (-1)^{g(x_1)} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |-\rangle \otimes |-\rangle. \tag{17}$$

But now something goes horribly wrong. We'd like to apply the n qubit Hadamard to this state to find the hidden string s (because $g(x_1) = s \cdot x_1$). But this doesn't work! Why? Because of the $|s_{x_1}\rangle$ terms. These terms make it so that the different terms in the first register can't interfere properly to extract out s! Doh!

So what do we do? Well we get tricky, of course. What we do is we apply the n qubit Hadamard to the second register and then we apply our oracle U again. What does this do? Well $H^{\otimes n}$ turns our state into

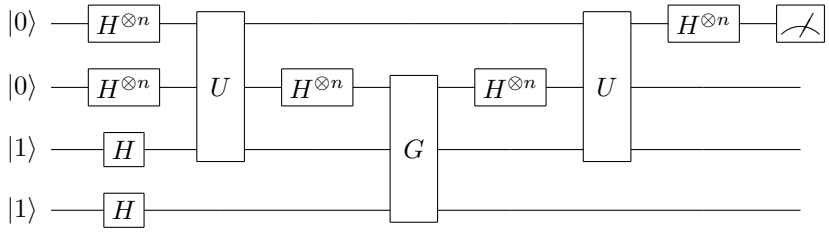
$$\frac{1}{2^n} \sum_{x_1, x_2 \in \{0,1\}^n} (-1)^{g(x_1)} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle \otimes |x_2\rangle \otimes |-\rangle \otimes |-\rangle. \tag{18}$$

But then applying U does another phase kickback and results in

$$\frac{1}{2^n} \sum_{x_1, x_2 \in \{0,1\}^n} (-1)^{g(x_1)} (-1)^{s_{x_1} \cdot x_2} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle \otimes |x_2\rangle \otimes |-\rangle \otimes |-\rangle = \frac{1}{2^n} \sum_{x_1} (-1)^{g(x_1)} |x_1\rangle \otimes |\psi\rangle \otimes |-\rangle \otimes |-\rangle. \tag{19}$$

where $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_2 \in \{0,1\}^n} |x_2\rangle$. Thus we see that we have uncomputed the register s_{x_1} which was causing us trouble. Now if we apply the n qubit Hadamard to the first register, and measure in the computational basis, we will obtain s!

Here is the circuit for the two level Bernstein-Vazirani problem:



(20)

Note that the first two wires on n qubit quantum wires and the G gate acts between the second wire and the fourth wire.

So we've seen how to solve the level two Bernstein-Vazirani problem. The same tricks we played to do this can be used to solve the k level Bernstein-Vazirani problem. How many uses of U will this require? Well for each level of recursion we will need two uses of the gadget which solves the lower level problem. This means that if we are trying to solve the k level problem, then we can solve the problem use $O(2^k)$ queries quantum mechanically.

So now we see how Bernstein and Vazirani showed a superpolynomial separation. If we set $k = \log(n)$ (we will take logarithms in this course in base 2), then our quantum algorithm takes only $O(n)$ queries, while the classical problem requires $\Omega(n^{\log n})$ queries. But $n^{\log n}$ grows faster than any polynomial. Thus there is a superpolynomial separation between the quantum and classical query complexities!

Oftentimes the recursive Bernstein-Vazirani problem is just passed over in a history of quantum computings and never discussed in much detail. This is a shame because it is a beautiful result and was the first really solid evidence that quantum computers might challenge the classical complexity of algorithms. There are also some interesting properties of this problem which make it not fit in with the other problems we will encounter later on. For those of you in the know, one interesting point about the Bernstein-Vazirani problem is that it is not known to be in NP!

Acknowledgments

The diagrams in these notes were typeset using Q-circuit a latex utility written by graduate student extraordinaires Steve Flammia and Bryan Eastin.