Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2013)
Andrew Childs, University of Waterloo

LECTURE 13: Query complexity and the polynomial method

So far, we have discussed several different kinds of quantum algorithms. In the next few lectures, we will discuss ways of establishing limitations on the power of quantum algorithms. After reviewing the model of quantum query complexity, this lecture presents the *polynomial method*, an approach that relates quantum query algorithms to properties of polynomials.

Quantum query complexity

Many of the algorithms we have covered work in the setting of query complexity, where the input for a problem is provided by a black box. This setting is convenient since the black box provides a handle for proving lower bounds: we can often show that many queries are required to compute some given function of the black-box input. In contrast, it is notoriously difficult to prove lower bounds on the complexity of computing some function of explicit input data.

We briefly formalize the model of query complexity. Consider the computational task of computing a function $f\colon S \to T$, where $S \subset \Sigma^n$ is a set of strings over some input alphabet Σ. If $S = \Sigma^n$ then we say f is *total*; otherwise we say it is *partial*. The input string $x \in S$ is provided to us by a black box that computes x_i for any desired $i \in \{1, \ldots, n\}$. A query algorithm begins from a state that does not depend on the oracle string x. It then alternate between queries to the black box and other, non-query operations. Our goal is to compute $f(x)$ using as few queries to the black box as possible.

Of course, the minimum number of queries (which we call the *query complexity* of f) depends on the kind of computation we allow. There are at least three natural models:

- $D(f)$ denotes the deterministic query complexity, where the algorithm is classical and must always work correctly.
- R_ϵ denotes the randomized query complexity with error probability at most ϵ. Note that this it does not depend strongly on ϵ since we can boost the success probability by repeating the computation several times and take a majority vote. Therefore $R_\epsilon(f) = \Theta(R_{1/3}(f))$ for any constant ϵ, so sometimes we simply write $R(f)$.
- Q_ϵ denotes the quantum query complexity, again with error probability at most ϵ. Similarly to the randomized case, $Q_\epsilon(f) = \Theta(Q_{1/3}(f))$ for any constant ϵ, so sometimes we simply write $Q(f)$.

We know that $D(\textsc{or}) = n$ and $R(\textsc{or}) = \Theta(n)$. Grover's algorithm shows that $Q(\textsc{or}) = O(\sqrt{n})$. In this lecture we will use the polynomial method to show (among other things) that $Q(\textsc{or}) = \Omega(\sqrt{n})$, a tight lower bound.

Quantum queries

A quantum query algorithm begins from x-independent state $|\psi\rangle$ and applies a sequence of unitary operations $U_1, \ldots, U_t$ interspersed with queries O_x, resulting in the state

$$|\psi_x^t\rangle := U_t O_x \ldots U_2 O_x U_1 O_x |\psi\rangle. \tag{1}$$

To make this precise, we need to specify the action of the oracle O_x.

For simplicity, we will mostly consider the case where the input is a bit string, i.e., $\Sigma = \{0,1\}$. Perhaps the most natural oracle model is the bit flip oracle $\hat{O}_x$, which acts as

$$\hat{O}_x|i,b\rangle = |i, b \oplus x_i\rangle \qquad \text{for } i \in \{1,\ldots,n\},\ b \in \{0,1\}. \tag{2}$$

This is simply the linear extension of the natural reversible oracle mapping $(i,b) \mapsto (i, b \oplus x_i)$, which can be performed efficiently given the ability to efficiently compute $i \mapsto x_i$. Note that the algorithm may involve states in a larger Hilbert space; implicitly, the oracle acts as the identity on any ancillary registers.

It is often convenient to instead consider the phase oracle, which is obtained by conjugating the bit-flip oracle by Hadamard gates: by the well-known phase kickback trick, $O_x = (I \otimes H)\hat{O}_x(I \otimes H)$ satisfies

$$O_x|i,b\rangle = (-1)^{bx_i}|i,b\rangle \qquad \text{for } i \in \{1,\ldots,n\},\ b \in \{0,1\}. \tag{3}$$

Note that this is slightly wasteful since $O_x|i,0\rangle = |i,0\rangle$ for all i; we could equivalently consider a phase oracle O'_x defined by $O'_x|0\rangle = |0\rangle$ and $O'_x|i\rangle = (-1)^{x_i}|i\rangle$ for all $i \in \{1,\ldots,n\}$. However, it is essential to include the ability to not query the oracle by giving the oracle some eigenstate of known eigenvalue, independent of x. If we could only perform the phase flip $|i\rangle \mapsto (-1)^{x_i}|i\rangle$ for $i \in \{1,\ldots,n\}$, then we could not tell a string x from its bitwise complement $\bar{x}$.

These constructions can easily be generalized to the case of a d-ary input alphabet, say $\Sigma = \mathbb{Z}_d$ (identifying input symbols with integers modulo d). Then for $b \in \Sigma$, we can define an oracle $\hat{O}_x$ by

$$\hat{O}_x|i,b\rangle = |i, b + x_i\rangle \qquad \text{for } i \in \{1,\ldots,n\},\ b \in \mathbb{Z}_d. \tag{4}$$

Taking the Fourier transform of the second register gives a phase oracle $O_x = (I \otimes F^\dagger_{\mathbb{Z}_d})\hat{O}_x(I \otimes F_{\mathbb{Z}_d})$ satisfying

$$O_x|i,b\rangle = \omega_d^{bx_i}|i,b\rangle \qquad \text{for } i \in \{1,\ldots,n\},\ b \in \mathbb{Z}_d \tag{5}$$

where $\omega_d := e^{2\pi i/d}$.

Quantum algorithms and polynomials

The following shows a basic connection between quantum algorithms and polynomials.

Lemma. *The acceptance probability of a t-query quantum algorithm for a problem with black-box input $x \in \{0,1\}^n$ is a polynomial in $x_1,\ldots,x_n$ of degree at most $2t$.*

Proof. We claim that the amplitude of any basis state is a polynomial of degree at most t, so that the probability of any basis state (and hence the probability of success) is a polynomial of degree at most $2t$.

The proof is by induction on t. If an algorithm makes no queries to the input, then its success probability is independent of the input, so it is a constant, a polynomial of degree 0.

For the induction step, a query maps

$$|i,b\rangle \overset{O_x}{\mapsto} (-1)^{bx_i}|i,b\rangle \tag{6}$$

$$= (1 - 2bx_i)|i,b\rangle, \tag{7}$$

so it increases the degree of each amplitude by at most 1. □

Consider a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. We say a polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ represents f if $p(x) = f(x)$ for all $x \in \{0,1\}^n$. Letting $\deg(f)$ denote the smallest degree of any polynomial representing f, we have $Q_0(f) \geq \deg(f)/2$.

To handle bounded-error algorithms, we introduce the concept of *approximate degree*. We say a polynomial p ϵ-represents f if $|p(x) - f(x)| \leq \epsilon$ for all $x \in \{0,1\}^n$. Then the ϵ-approximate degree of f, denoted $\widetilde{\deg}_\epsilon(f)$, is the smallest degree of any polynomial that ϵ-represents f. Clearly, $Q_\epsilon(f) \geq \widetilde{\deg}_\epsilon(f)/2$. Since bounded-error query complexity does not depend strongly on the particular error probability ϵ, we can define, say, $\widetilde{\deg}(f) := \widetilde{\deg}_{1/3}(f)$.

Now to lower bound the quantum query complexity of a Boolean function, it suffices to lower bound its approximate degree.

Symmetrization

While polynomials are well-understood objects, the acceptance probability is a multivariate polynomial, so it can be rather complicated. Since $x^2 = x$ for $x \in \{0,1\}$, we can restrict our attention to multilinear polynomials, but it is still somewhat difficult to deal with such polynomials directly. Fortunately, for many functions it suffices to consider a related univariate polynomial obtained by symmetrization.

For a string $x \in \{0,1\}^n$, let $|x|$ denote the Hamming weight of x, the number of 1s in x.

Lemma. *Given any n-variate multilinear polynomial p, let $P(k) := \mathbb{E}_{|x|=k}[p(x)]$. Then P is a polynomial with $\deg(P) \leq \deg(p)$.*

Proof. Since p is multilinear, it can be written as a sum of monomials, i.e., as

$$p(x) = \sum_{S \subseteq \{1,\ldots,n\}} c_S \prod_{i \in S} x_i \tag{8}$$

for some coefficients c_S. Then we have

$$P(k) = \sum_{S \subseteq \{1,\ldots,n\}} c_S \mathop{\mathbb{E}}_{|x|=k} \left[\prod_{i \in S} x_i \right] \tag{9}$$

and it suffices to compute the expectation of each monomial. We find

$$\mathop{\mathbb{E}}_{|x|=k} \left[\prod_{i \in S} x_i \right] = \Pr_{|x|=k} [\forall\, i \in S,\ x_i = 1] \tag{10}$$

$$= \frac{\binom{n-|S|}{k-|S|}}{\binom{n}{k}} \tag{11}$$

$$= \frac{(n-|S|)!\, k!\, (n-k)!}{(k-|S|)!\, (n-k)!\, n!} \tag{12}$$

$$= \frac{(n-|S|)!}{n!} k(k-1)\cdots(k-|S|+1) \tag{13}$$

which is a polynomial in k of degree $|S|$. Since $c_S = 0$ whenever $|S| > \deg(p)$, we see that $\deg(P) \leq \deg(p)$. □

Thus the polynomial method is a particularly natural approach for symmetric functions, those that only depend on the Hamming weight of the input.

Parity

Let PARITY: $\{0,1\}^n \to \{0,1\}$ denote the symmetric function $\text{PARITY}(x) = x_1 \oplus \cdots \oplus x_n$. Recall that Deutsch's problem, which is the problem of computing the parity of 2 bits, can be solved exactly with only one quantum query. Applying this algorithm to a pair of bits at a time and then taking the parity of the results, we see that $Q_0(\text{PARITY}) \leq n/2$.

What can we say about lower bounds for computing parity? Symmetrizing PARITY gives the function $P\colon \{0,1,\ldots,n\} \to \mathbb{R}$ defined by

$$P(k) = \begin{cases} 0 & \text{if } k \text{ is even} \\ 1 & \text{if } k \text{ is odd.} \end{cases} \tag{14}$$

Since P changes direction n times, $\deg(P) \geq n$, so we see that $Q_0(\text{PARITY}) \geq n/2$. Thus Deutsch's algorithm is tight among zero-error algorithms.

What about bounded-error algorithms? To understand this, we would like to lower bound the approximate degree of PARITY. If $|p(x) - f(x)| \leq \epsilon$ for all $x \in \{0,1\}^n$, then

$$|P(k) - F(k)| = \left| \mathop{\mathbb{E}}_{|x|=k} (p(x) - f(x)) \right| \leq \epsilon \tag{15}$$

for all $k \in \{0,1,\ldots,n\}$, where P is the symmetrization of p and F is the symmetrization of f. Thus, a multilinear polynomial p that ϵ-approximates PARITY implies a univariate polynomial P satisfying $P(k) \leq \epsilon$ for k even and $P(k) \geq 1 - \epsilon$ for k odd. For any $\epsilon < 1/2$, this function still changes direction n times, so in fact we have $\widetilde{\deg}_\epsilon(f) \geq n$, and hence $Q_\epsilon(\text{PARITY}) \geq n/2$.

This shows that the strategy for computing parity using Deutsch's algorithm is optimal, even among bounded-error algorithms. This is an example of a problem for which a quantum computer cannot get a significant speedup—here the speedup is only by factor of 2. In fact, we need at least $n/2$ queries to succeed with any bounded error, even with very small advantage (e.g., even if we only want to be correct with probability $\frac{1}{2} + 10^{-100}$). In contrast, while the adversary method can prove an $\Omega(n)$ lower bound for parity, the constant factor that it establishes is error-dependent.

Note that this also shows we need $\Omega(n)$ queries to exactly count the number of marked items in an unstructured search problem, since exactly determining the number of 1s would in particular determine whether the number of 1s is odd or even.

Unstructured search

Next we will see how the polynomial method can be used to prove the $\Omega(\sqrt{n})$ lower bound for computing the logical OR of n bits. Symmetrizing OR gives a function $F(k)$ with $F(0) = 0$ and $F(1) = 1$. We also have $F(k) = 1$ for all $k > 1$, but we will not actually need to use this. This function is monotonic, so we cannot use the same simple argument we applied to parity. Nevertheless, we can prove that $\widetilde{\deg}(\text{OR}) = \Omega(\sqrt{n})$ using the following basic fact about polynomials, due to Markov.

Lemma. *Let $P\colon \mathbb{R} \to \mathbb{R}$ be a polynomial. Then*

$$\max_{x \in [0,n]} \frac{\mathrm{d}P(x)}{\mathrm{d}x} \leq \frac{\deg(P)^2}{n} \left(\max_{x \in [0,n]} P(x) - \min_{x \in [0,n]} P(x) \right). \tag{16}$$

In other words, if we let

$$h := \max_{x \in [0,n]} P(x) - \min_{x \in [0,n]} P(x) \tag{17}$$

denote the "height" of P in the range $[0, n]$, and

$$d := \max_{x \in [0,n]} \frac{\mathrm{d}P(x)}{\mathrm{d}x} \tag{18}$$

denote the largest derivative of P in that range, then we have $\deg(P) \geq \sqrt{nd/h}$.

Now let P be a polynomial that ϵ-approximates OR. Since $P(0) \leq \epsilon$ and $P(1) \geq 1 - \epsilon$, P must increase by at least $1 - 2\epsilon$ in going from $k = 0$ to $k = 1$, so $d \geq 1 - 2\epsilon$.

We have no particular bound on h, since we have no control over the value of P at non-integer points; the function could become arbitrarily large or small. However, since $P(k) \in [0, 1]$ for $k \in \{0, 1, \ldots, n\}$, a large value of h implies a large value of d, since P must change fast enough to start from and return to values in the range $[0, 1]$. In particular, P must change by at least $(h - 1)/2$ over a range of k of width at most $1/2$, so we have $d \geq h - 1$. Therefore,

$$\deg(P) \geq \sqrt{\frac{n \max\{1 - 2\epsilon, h - 1\}}{h}} \tag{19}$$

$$= \Omega(\sqrt{n}). \tag{20}$$

It follows that $Q(\text{OR}) = \Omega(\sqrt{n})$.

Note that the same argument applies for a function that takes the value 0 whenever $|x| = w$ and the value 1 whenever $|x| = w + 1$, for any w; in particular, it applies to any non-constant symmetric function. (Of course, we can do better for some symmetric functions, such as PARITY and also MAJORITY, among others.)