Quantum Computing 4/5/2022

Lecture 18: Fourier Sampling

Instructor: Dieter van Melkebeek Scribe: Jacob Frederick

In this lecture we examine Simon's problem, which involves finding a "hidden XOR-shift" of a function. The quantum algorithm solving Simon's problem shows exponential speedup over the classical counterpart. More generally, this quantum algorithm is an example of Fourier sampling, a paradigm that will be further investigated in subsequent lectures.

At the end of this lecture, we give a brief overview of the achievable gaps between quantum, probabilistic, and classical query complexity. We also mention techniques for establishing lower bounds on quantum query complexity.

1 Finding a Hidden XOR-Shift

1.1 Problem statement

Suppose we are given a black box function

$$f : \{0,1\}^n \to \{0,1\}^n$$

for which there exists a nonzero $s \in \{0,1\}^n$ such that for any two inputs $x_1, x_2 \in \{0,1\}^n$ with $x_1 \neq x_2$

$$f(x_1) = f(x_2)$$

if and only if

$$x_1 = x_2 \oplus s.$$

Put colloquially, f is a 2-to-1 function where each pair of inputs differ by the XOR-shift s. Simon's problem is to find the value of s. Specifically, we want to do this in the fewest number of queries to the black-box function f [Sim94].

1.2 Deterministic Algorithm

Naively, since we know our function is 2-to-1 over all N possible inputs, we can guarantee a collision with $\frac{N}{2} + 1$ queries by the pigeonhole principle.

A more optimal way to solve this problem is known the "baby-step giant-step" approach or sometimes the "meet in the middle" approach. This approach works by dividing the bitstring s into two parts. Let s_{left} represent the left $\lfloor \frac{n}{2} \rfloor$ bits of s and s_{right} represent the right $\lceil \frac{n}{2} \rceil$ bits.

Claim. $f(s_{left}0^{\lceil \frac{n}{2} \rceil}) = f(0^{\lfloor \frac{n}{2} \rfloor}s_{right})$

Proof. $(s_{left}0^{\lceil \frac{n}{2} \rceil}) \oplus s = (s_{left} \oplus s_{left})(0^{\lceil \frac{n}{2} \rceil} \oplus s_{right}) = 0^{\lfloor \frac{n}{2} \rfloor}s_{right}$ □

This means that there exists at least one collision between $x_1 \in \{0,1\}^{\lfloor \frac{n}{2} \rfloor}0^{\lceil \frac{n}{2} \rceil}$ and $x_2 \in 0^{\lfloor \frac{n}{2} \rfloor}\{0,1\}^{\lceil \frac{n}{2} \rceil}$. Querying all of these values for x_1 and x_2 would require $2^{\lfloor \frac{n}{2} \rfloor} + 2^{\lceil \frac{n}{2} \rceil}$ queries, which gives us a complexity of $O(\sqrt{N})$.

1.2.1 Lower Bound

Above, we described an $O(\sqrt{N})$ classical algorithm to solve Simon's problem. In fact, the lower bound is found to be $\Omega(\sqrt{N})$. We can see this with an adversarial approach such that every query to f returns a distinct value (until no longer mathematically possible). In this scenario, the only values of s that can be ruled out are the results of pairwise XOR operations between query inputs.

Generally, after making the kth query against f, we have $\binom{k}{2}$ pairs of inputs that we know are not related by s. If we have chosen our inputs wisely, then each pair of inputs will correspond to unique values of s so that we have eliminated $\binom{k}{2}$ distinct possibilities. In any case there can be no more than $\binom{k}{2}$ for s that are ruled out.

In order to overcome our adversarial function, we need to eliminate $N-2$ values for s. The number 2 comes from the knowledge that (1) $s=0$ is not possible and (2) if we have eliminated all but one remaining possibility then we have found s. Therefore, we require k^* queries where $\binom{k^*}{2} = N-2$.

$$\binom{k^*}{2} = \frac{k^*!}{2!(k^*-2)!} = \frac{k^*(k^*-1)}{2} = N-2$$

Thus we can see that the lower bound for a deterministic classical algorithm requires $k^* = \Omega(\sqrt{N})$ queries.

1.3 Probabilistic Algorithm

In a probabilistic classical setting, we can solve this problem with a simple $O(\sqrt{N})$ algorithm based on the birthday paradox. By making $\sqrt{N}$ queries with random inputs, we will more than likely find a collision. This is conceptually simpler than our deterministic algorithm but does not improve the query complexity beyond a constant factor.

Furthermore, by Yao's principle, it can be proven that a probabilistic algorithm cannot do better than the deterministic $O(\sqrt{N})$ query complexity.

1.4 Quantum Algorithm

Below is the circuit that solves Simon's problem in $O(n) = O(\log N)$ complexity. It looks very similar to the circuit that implements the Deutsch-Jozsa algorithm (to determine constant versus balanced functions) except the second register in that circuit consists of a single qubit rather than the n qubits shown here. Below we prove that this circuit lets us solve Simon's problem.

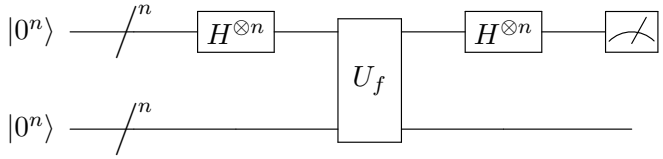

Figure 1: Quantum circuit to solve Simon's problem. The $\not{|}^n$ represents a collection of n wires.

Now we analyze the behavior of this circuit. The first n-fold Hadamard gate takes us from $|0^n\rangle\,|0^n\rangle$ to

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle |0^n\rangle .$$

Then, the $2n$-qubit unitary gate U_f, which performs $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, further transforms the state to

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle .$$

Finally, the last n-fold Hadamard gate transforms the state to

$$\frac{1}{N} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle |f(x)\rangle .$$

At this point in the circuit, we measure the first register which is in a superposition, so we are sampling from some probability distribution. To understand that probability distribution, we look at the amplitudes of the states in our superposition.

Remember that f is a 2-to-1 function so its range is of size $\frac{N}{2}$. States in our superposition can only interfere with each other if they share the same $|f(x)\rangle$ in the second register. Therefore, each value of x gives exactly 2 states that interfere with each other.

For each $z \in \text{Range}(f)$ there exists exactly one corresponding pair of inputs. Let x^* be one of these inputs such that $z = f(x^*) = f(x^* \oplus s)$. (The choice of x^* from the pair is arbitrary, but it is important that we are choosing a single value from each pair). Using this, we can rewrite our state just before measurement as

$$\sum_{z \in \text{Range}(f)} \sum_{y \in \{0,1\}^n} \alpha_{y,z} |y\rangle |z\rangle$$

where

$$\alpha_{y,z} = \frac{1}{N} \left((-1)^{x^* \cdot y} + (-1)^{(x^* \oplus s) \cdot y} \right).$$

Now, using the property that

$$(x^* \oplus s) \cdot y = (x^* \cdot y) \oplus (s \cdot y),$$

we can rewrite $\alpha_{y,z}$ as

$$\alpha_{y,z} = \frac{1}{N}(-1)^{x^* \cdot y}(1 + (-1)^{s \cdot y}) = \begin{cases} \frac{2}{N}(-1)^{x^* \cdot y} & \text{if } s \cdot y = 0 \bmod 2 \\ 0 & \text{if } s \cdot y = 1 \bmod 2 \end{cases}.$$

1.4.1 Output Analysis

The probability of observing a particular state y in the first register is the sum of $|\alpha_{y,z}|^2$ over all $z \in \text{Range}(f)$. Since $|\alpha_{y,z}|$ does not depend on z, our probability of observing y can be written as

$$\Pr[y] = \sum_{z \in \text{Range}(f)} |\alpha_{y,z}|^2 = \begin{cases} \frac{2}{N} & \text{if } s \cdot y = 0 \bmod 2 \\ 0 & \text{if } s \cdot y = 1 \bmod 2 \end{cases}.$$

So our observed y always satisfies $s \cdot y = 0 \bmod 2$. We can define a vector space

$$s^\perp \doteq \{y \in \{0,1\}^n : s \cdot y = 0 \bmod 2\}$$

which always contains our observed y. The cardinality of $s^\perp$ is 2^{n-1}.

We run the quantum circuit in Figure 1 multiple times. The i-th run yields $y_i \in s^\perp$ chosen uniformly at random and independent from the earlier runs. Subsequent runs of the circuit only yield new information if we measure y_i that is nonzero and linearly independent from the previously measured $y_1, \ldots, y_{i-1}$ values. The probability that our new vector is linearly independent from previous measurements can be written as

$$\Pr[y_i \notin \operatorname{span}(y_1, \ldots, y_{i-1})] = 1 - \frac{|\operatorname{span}(y_1, \ldots, y_{i-1})|}{|s^\perp|} = 1 - \frac{2^d}{2^{n-1}}$$

where

$$d = \dim(\operatorname{span}(y_1, \ldots, y_{i-1})).$$

As long as $d < n-1$ then the probability $p = 1 - \frac{2^d}{2^{n-1}} \geq \frac{1}{2}$. Treating this as a Bernoulli experiment, the expected number of trials to achieve a new linearly independent measurement is $\frac{1}{p} = 2$. Thus, the expected number of runs until we reach $d = n-1$ is $O(n)$. At that point, we have $n-1$ linearly independent equations of the form $y_i \cdot s = 0 \bmod 2$. Since we have n unknown variables (i.e., the bits of s) and $n-1$ linearly independent equations, the system has two solutions: the zero vector and the XOR shift s.

It would be even better if we did not need to rely on probabilistic success to find linearly independent values of y_i. Since we know the exact probability of success (which is $1 - \frac{d}{2^{n-1}}$) then we can use amplitude amplification to guarantee that we measure a linearly independent y_i. Our success criteria is $s \notin \operatorname{span}(y_1, \ldots, y_{i-1})$ and our unitary circuit A is the circuit in Figure 1 (without the final measurement) which is also its own inverse. Since our success probability is always $\geq \frac{1}{2}$ and known exactly, we only require one iteration of amplitude amplification. This results in 3 applications of our circuit: one application to generate the initial state $A|0^{2n}\rangle$, one application of A^{-1} to get to the $|0^{2n}\rangle$ state and reflect around it, and one more application of A after the reflection. Therefore we will make 3 queries to the black box function per y_i. In order to achieve $n-1$ linearly independent measurements, we require exactly $3(n-1)$ queries to f, at which point we can solve for s.

Thus the query complexity of this quantum algorithm is $O(n) = O(\log N)$, representing an exponential improvement over the best possible classical algorithms.

Exercise 1. *Given two one-to-one functions $f, g : \{0,1\}^n \to \{0,1\}^n$ where $g(x) = f(x \oplus s)$ for some $s \in \{0,1\}^n$. Find s, with certainty, using $O(n)$ queries. Note that s may be 0.*

2 Fourier Sampling

The quantum algorithm above that solves Simon's problem belongs to a broader class of Fourier sampling algorithms. Fourier sampling proceeds in two steps.

1. Transform a pure state $\sum_x \alpha_x |x\rangle$ into a Fourier state $\sum_y \hat{\alpha}(y) |y\rangle$ where $\hat{\alpha}$ is the Fourier transform of α.
2. Observe (i.e., sample) the Fourier state.

The specific behavior of the Fourier transform depends on the group G over which it is defined.

2.1 The XOR-Shift

In our quantum algorithm above, $H^{\otimes n}$ implements the Fourier transform over the group of $\mathbb{Z}_2^n$ under the operation $\oplus$. For a fixed $z \in \text{Range}(f)$ our pure state had amplitudes

$$|\alpha_x| \sim \chi[f(x) = z]$$

where χ is the boolean indicator function. After applying the $H^{\otimes n}$ Fourier transform, we sampled the Fourier state by measuring the first register.

2.2 Benefits to Quantum Computing

Fourier transforms are good at extracting the symmetries captured by their underlying groups. There are efficient classical algorithms to compute the Fourier transform over several groups, but classical Fourier sampling algorithms would require $\Omega(N)$ applications of the Fourier transform where N is the size of the group. The advantage of quantum Fourier sampling algorithms comes from the ability to perform the Fourier transform over a superposition of input states representing N group elements as $n = \log N$ qubits. However, this is only a viable strategy when we have a way to extract useful information from the superposition of output states. Naively, repeatedly measuring the superposition would require $\Omega(N)$ queries, negating any quantum advantage. In Simon's algorithm, we use the second "tag" register along with amplitude amplification to extract a solution in $O(\log N) = O(n)$ queries. Clearly input and output representations are important factors in quantum speedup.

We will explore other quantum algorithms that leverage Fourier transforms in future lectures. For now, we mention two additional groups for which the quantum Fourier transform can be realized efficiently.

- $\mathbb{Z}_N$ (the group of integers modulo N) under addition. The quantum Fourier transform for this group has been applied successfully to find the period of functions, which is the basis of Shor's algorithm for factoring integers.
- The symmetric group of permutations on n elements, under function composition. A potential application of this quantum Fourier transform is the graph isomorphism problem. However, all attempts to date have stalled. In particular, while an efficient quantum Fourier transform has been shown, the Fourier sampling approach would require an exponential number of samples as discussed above. This does not rule out the possibility of achieving quantum speedup with other approaches based on the quantum Fourier transform.

3 Quantum Query Complexity Gaps

As classical algorithms can be simulated efficiently on a quantum computer, quantum query complexity can never be (much) higher than classical query complexity. Depending on the problem, we have seen a variety of gaps in the other direction. In some cases the speedup is polynomial while in others the speedup is exponential. The following section analyzes the largest and smallest possible gaps for promise problems versus fully specified problems.

A promise problem is one that guarantees some property of the input. Put another way, the set of allowed inputs is a strict subset of all possible inputs. By contrast, a fully specified problem is one that is well-defined for every possible input.

3.1 Fully Specified Problems

For fully specified problems, the gap between classical and quantum query complexity can be at most polynomial. More specifically, let D denote the query complexity in the deterministic setting, R in the probabilistic setting (with error probability bounded by some constant, say $1/2$), and Q in the quantum setting (again, with error probability bounded by some constant, say $1/2$). It can be shown that

$$D = O(Q^4)$$

and

$$R = O(Q^4)$$

The relationship for D is known to be tight up to polylog factors. That is, there exists a fully specified (albeit contrived) problem for which $D = \Omega(N/\log N)$ and $Q = \tilde{O}(N^{1/4})$. The relationship for R is not known to be tight, but it is known that the exponent of 4 cannot be reduced below 3 [ABB$^+$17].

3.2 Promise Problems

As Simon's problem illustrates, for promise problems, exponential gaps are possible between classical and quantum query complexity. In fact, even larger gaps can be achieved.

If $Q = 1$ then D can be up to $\Omega(N)$. We discussed such a problem in an earlier lecture, namely the problem of deciding whether a function is constant or balanced.

If $Q = 1$ then R is $O(\sqrt{N})$ which is known to be tight up to polylogarithmic factors. In particular, the promise problem known as Forrelation can be decided by a quantum algorithm with a single query, while any probabilistic algorithm needs to make $\Omega(\sqrt{N}/\log N)$ queries.

The term "Forrelation" is a contraction of "Fourier correlation". The problem is as follows. Given as input two functions $f, g : \{0,1\}^n \to \{0,1\}$, let

$$\sum_x \alpha_x |x\rangle \doteq \frac{1}{\sqrt{N}} \sum_x (-1)^{f(x)} |x\rangle$$

and

$$\sum_x \beta_x |x\rangle \text{ is the Fourier transform of } \frac{1}{\sqrt{N}} \sum_x (-1)^{g(x)} |x\rangle$$

. Then the Forrelation of f and g is the expected value $E_x[\alpha_x \beta_x]$. The problem is to distinguish whether the Forrelation is above a certain threshold or below a smaller threshold in absolute value, where $x \in \{0,1\}^n$ is chosen uniformly at random. The "promise" in this problem is that the Forrelation will never be between these two thresholds. A swap-test algorithm solves this problem with a single quantum black-box query [AA18].

Exercise 2. *Find the probability of measuring* 0 *at the end of the swap-test circuit, as a function of* ψ *and* ϕ*. Why is it called a swap-test?*

4 Quantum Query Lower Bounds

There are two well-established methods for proving lower bounds on quantum query complexity:

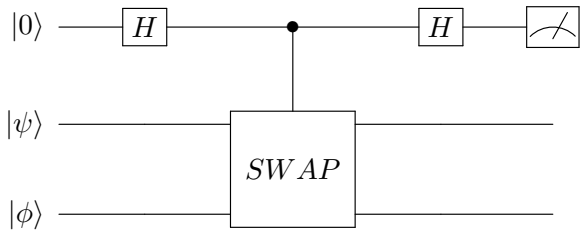

Figure 2: Quantum circuit to perform a swap-test, which solves the Forrelation problem.

- The quantum adversary method, which is a quantum version of the deterministic adversary method. We used the deterministic adversary method to prove a lower bound for classical solutions to Simon's problem.
- The polynomial method, which is unique to the quantum setting. This method hinges on the fact that the final amplitudes of a k-query unitary circuit are given by multivariate polynomials of degree at most k in variables that represent the values in the truth-table of the input function f. Lower bounds on the degree needed to represent the problem thereby yield lower bounds on the quantum query complexity. As an example, for the OR function, $Q(OR) = \Theta(\sqrt{N})$, or to get the answer exactly, $Q_{\text{exact}}(OR) = N$. This implies the optimality of Grover's algorithm in terms of query complexity.

References

[AA18] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. *SIAM J. Comput.*, 47(3):982–1038, 2018.

[ABB+17] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *J. ACM*, 64(5):32:1–32:24, 2017.

[Sim94] Daniel R. Simon. On the power of quantum computation. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 116–123. IEEE Computer Society, 1994.