CMSC 858L: Quantum Complexity

Instructor: Daniel Gottesman

Spring 2023

23 Experimental quantum supremacy, PSPACE

23.1 References

Savitch's theorem is a standard of classical complexity theory, but the full proof that BQPSPACE = PSPACE is in Watrous, "On the complexity of simulating space-bounded quantum computations," Computational Complexity vol. 12, 48-84 (2003).

23.2 Complexity of experimental quantum supremacy

In recent years, there has been significant interest in experimental validation of a quantum speedup in NISQ computers by a demonstration of quantum supremacy through one of the classes hopefully satisfying the above properties. However, there are some further complexity-related issues with such experiments.

First, the experiments don't precisely measure the output distribution $\tilde{p}_a$, which would require many samples. Instead, they need a simple criterion to decide if the experiment is close enough to the correct one to demonstrate supremacy. Various criteria have been proposed but they are necessarily weaker than simply being close in statistical distance. In particular, it is possible that it could be hard to generate a distribution that has small statistical distance to p_a while still being possible to generate a distribution that passes this particular test. Thus, hardness of the experimental demonstrations rests on stronger assumptions than those we have discussed.

Second, the whole point of these hardness arguments is that it is hard to calculate p_a classically. Since the actual circuit used in an experiment is generated randomly, this means that the correct distribution p_a is unknown, making it difficult to tell if the observed distribution passes the test (whatever it is) for being sufficiently close to the quantum distribution. For systems just over the threshold of quantum supremacy, a large enough classical computation can find p_a, but that won't work at all for larger systems. There are some possibilities for a quantum superiority experiment with a more straightforward test, but all suggestions so far require even stronger complexity assumptions to establish quantum supremacy.

Finally, quantum supremacy in the asymptotic sense (where complexity-theoretic arguments have their full power) requires fault tolerance. The problem is that in a model without fault tolerance, noise will overwhelm the system after a short amount of time (anything more than log depth), eliminating any advantage for the quantum circuit. (Classical randomized circuits can also generate noise.)

23.3 BQPSPACE vs. PSPACE

We are now ready to move up to a larger complexity class, PSPACE. Recall that at the beginning of class we defined PSPACE as the class of problems solvable by a classical algorithm with polynomial space available to it, and that we said we don't need to add a BQPSPACE class because it is the same as PSPACE. Now is the time to prove that.

Unfortunately, I can't quite do that because it turns out to be more complicated than I realized. We can *almost* prove it, and the almost proof is quite straightforward given what we've done before.

Theorem 1. *Suppose a language L can be decided by a quantum algorithm that runs in at most exponential time using at most polynomial space. Then $L \in PSPACE$.*

Proof. First, remember how we proved that BQP $\subseteq$ PSPACE: by computing the path integral as an exponentially large sum. Of course, it turns out that that can be done in the (probably) smaller complexity class PP. But when we expand the quantum circuits to exponential size, then this argument doesn't work any more because each term in the sum is an exponential size product which naively requires too much space to write down.

However, by improving our method using a version of the proof of Savitch's theorem (which shows PSPACE = NPSPACE), we can get the argument to work. In particular, note that

$$A_{y,x}(0, f(n)-1) = \langle x| \prod_{i=0}^{f(n)-1} U_i |y\rangle \tag{1}$$

$$= \sum_z \langle x| \prod_{i=0}^{f(n)/2-1} U_i |z\rangle\langle z| \prod_{i=f(n)/2}^{f(n)-1} U_i |0\rangle \tag{2}$$

$$= \sum_z A_{y,z}(f(n)/2, f(n)-1) A_{z,x}(0, f(n)/2). \tag{3}$$

That is, a transition amplitude for a circuit of size $f(n)$ between y and x can be written as an exponential sum of terms which are products of two transition amplitudes of circuits of half the size.

With a circuit of size $2^{O(\mathrm{poly}n)}$, we need only polynomially many levels of recursion, and each level can be computed in PSPACE by stepping through the sum over z and computing $A_{y,z}(f(n)/2, f(n)-1)$ and $A_{z,x}(0, f(n)/2)$. At any given time, we need enough space to keep track of one z for each level of recursion, which is a total of $O(n\mathrm{poly}n)$ space. (We only need to keep one z for each level of recursion because we can compute each A one at a time and don't need to keep track of the intermediate values once that it done.) □

But can language in BQPSPACE need more than exponential time to decide? I suspect it is true that a quantum algorithm using polynomial space can run for more than exponential time before halting, which cannot occur for a classical algorithm using polynomial space; but it is still true that there is an efficient algorithm to decide the same problem using classical polynomial space (and therefore exponential time).

The actual proof involves various technicalities. The main idea seems to be to determine if

$$\sum_{t \geq 0} [U^t]_{0,N} > 0, \tag{4}$$

where U is the matrix for the action of a quantum Turing machine on a Hilbert space of polynomially many qubits (those available to the algorithm). Here, the $(0, N)$ matrix element of U^t represents the amplitude for U iterated for t steps to cause a transition from the initial state $|0\rangle$ to the accepting state $|N\rangle$. If this sum is positive, then there is some chance of accepting. The individual matrix elements of U can be computed in PSPACE (and actually in P). The elements of U^t can also be computed in PSPACE provided t is at most $2^{O(\mathrm{poly}n)}$, but as noted above, I don't think this is straightforward to guarantee. Instead, the proof rewrites (4) as a formula involving a limit and determinants, and the determinant can be computed in PSPACE.

23.4 Interactive Proofs

NP and QMA can be thought of as classes involving proofs: NP is the class of languages for which there is a short proof of "yes" instances. MA generalizes the notion of proof slightly to allow probabilistic proofs, and QMA generalizes it further to allow the proof and checking procedure to be quantum.

But there is another way in which we can generalize the notion of a proof. A conventional mathematical proof is a static thing, but fundamentally a proof is about a method of convincing a logical skeptic about some statement. There is no need for this to be static: Instead, a proof can emerge from a conversation.

We can have a prover conversing with a verifier and allow the prover to convince the verfiier through a back and forth conversation. This is known as an *interactive proof.*

To see how interactive proofs can potentially have more power than non-interactive proofs, consider graph isomorphism. Suppose you have two graphs G_1 and G_2. If they are isomorphic, certainly there is an easy non-interactive proof of this: The prover simply describes a permutation to take G_1 into G_2, and the verifier can check this without any further involvement by the prover. However, graph *non*-isomorphism, the complement problem, is not known to be in NP (so graph isomorophism is not known to be in co-NP).

But there is an straightforward interactive proof of graph non-isomorphism: The verifier secretly picks one of the two graphs G_1 or G_2 and permutes it, then tells the verifier the permuted graph, but keeps secret which of the two graphs was chosen. The prover must then tell the verifier which one it was, and the verifier accepts only if the prover is right about thtat. If the graphs are actually isomorphic, then there is no way the prover can tell which was permuted except by guessing, whereas if they are non-isomorphic, a prover with no computational bound can tell which one (for instance by trying all permutations). Thus, if the prover can consistently say which of the two graphs was permuted, then they must have been non-isomorphic.

Definition 1. *A (classical) interactive protocol consists of a sequence of back-and-forth messages between a prover Alice and a verifier Bob. In odd rounds, Bob sends Alice a message based on their previous messages, Bob's private state, and any random bits Bob chooses to use. In even numbered rounds, the roles are reversed, with Alice sending a message to Bob based on ther previous messages, Alice's private state, and any other information needed. In all cases, there are at most polynomially many rounds, the messages sent are polynomial in size, and Bob's computations (but not Alice's) must also be of polynomial size. At the end, Bob outputs either "accept" to accept the protocol or "reject" to reject the protocol.*

A quantum interactive protocol is the same except that Alice and Bob hold a quantum state, Bob can perform polynomial-size quantum computations, and the messages passed back and forth are quantum states as well. At the end, Bob measures his state and outputs either "accept" or "reject" as above.

It makes the most sense to allow even the classical interactive protocols to be probabilistic. If Bob's responses are deterministic functions of his previous interactions, then Alice can compute what they will be herself and skip the interaction.

We can then define a class of problems for which there exist convincing interactive proof systems:

Definition 2. *Let IP be the set of languages L for which there exists an interactive (classical) protocol Π such that for any instance x of L,*

- *If $x \in L$, then there exists a verifier Alice such that Bob outputs "accept" with probability $\geq 2/3$.*
- *If $x \notin L$, then for all actions by the verifier Alice, Bob outputs "reject" with probability $\geq 2/3$.*

Let QIP be the set of languages L for which there exists a quantum interactive protocol with the same conditions.

It is a major result of classical complexity that IP = PSPACE. But what about QIP? This could be potentially stronger. Certainly IP $\subseteq$ QIP since we can run the classical protocol using basis states, but QIP could potentially include additional languages. However, it turns out not to, and proving this will be our next goal.

23.5 Formalizing quantum interactive proofs

Note that for a quantum interactive proof, we may as well assume that Alice and Bob perform only unitary operations. This is because any non-unitary CPTP map can be purified using some extra qubits. Alice or Bob can use the purified version and just keep the extra qubits off to the side, not using them again.

Therefore, we can think of a quantum interactive protocol as a sequence of unitaries $U_1^A, \ldots, U_t^A$ for Alice and a sequence of unitaries $V_0^B, \ldots, V_t^B$ for Bob. Alice and Bob each have their own registers H_A and H_B, plus a message register H_M which is passed back and forth. We can assume all registers are initialized to

$|00\ldots0\rangle$. For instance, if Bob starts the protocol, Bob starts by performing V_0^B on $H_B \otimes H_M$ and then passes H_M to Alice. Alice performs U_1^A on $H_M \otimes H_A$ and then passes H_M back to Bob. This repeats: Bob performs V_i^B on $H_B \otimes H_M$ and passes H_M to Alice, who performs U_{i+1}^A on $H_M \otimes H_A$ and passes H_M back to Bob, then repeat for $i+1$. Finally, Bob performs V_t^B and then measures one or more output qubits in H_B.

We are assuming limits on the power of Bob, but not of Alice, so Bob's Hilbert space H_B has polynomially many qubits and his unitaries V_i^B can be implemented efficiently, but Alice's Hilbert space H_A and unitaries U_i^A need not follow these restrictions.