CMSC 858L: Quantum Complexity

Instructor: Daniel Gottesman

Spring 2023

21 Quantum Supremacy

21.1 References

The definition of PostBPP and the relationship to $\mathrm{BPP}_{\mathrm{path}}$ comes from Bravyi, DiVincenzo, Oliveira, and Terhal, "The Complexity of Stoquastic Local Hamiltonian Problems," quant-ph/0606140. $\mathrm{BPP}_{\mathrm{path}}$ and its inclusion in $\mathrm{BPP}^{\mathrm{NP}}$ is from Han, Hemaspaandra, and Thierauf, "Threshold Computation and Cryptographic Security," SIAM J. Computing, vol. 26, p. 59 (1997). However, the proof here is simplified from those papers since it avoids the detour into $\mathrm{BPP}_{\mathrm{path}}$.

21.2 Correction

In class last time, I said that Σ_i^p and Π_i^p didn't have complete problems after the third level. I don't know where I got that: that is not true. For instance, let language L be the set of true Σ_i^p statements. Then L is Σ_i^p-complete. But it is true that PH has no complete languages for the reason I said, that any complete language would have to be in one of the levels and that would cause the polynomial hierarchy to collapse.

Also, I looked into why people think the polynomial hierarchy doesn't collapse, and ... there isn't a very strong reason that I can see. Some of it is the same intuition why P $\neq$ NP, that removing quantifiers seems to be hard; and the corresponding experience of people looking at it and being unable to collapse levels of the hierarchy. Also, it is proven that PH is infinite relative to a random oracle (a random oracle is a bit more convincing than finding a specific oracle that separates classes because it is natural in the sense that it is not tailored to the question you are asking). Finally, there is probably an element of "it is helpful to assume this." Since a variety of interesting and plausible results follow from the polynomial hierarchy being infinite, it is a convenient assumption to make.

21.3 PostBPP

We can also apply post-selection to classical probabilistic computation.

Definition 1. *Let PostBPP be the set of languages L such that there exists a polynomial-time randomized algorithm $A(x)$ (i.e., a uniform family of polynomial-size circuits including the ability to generate random bits) with the following properties: For any instance x,*

1. *$A(x)$ has two output bits A and B,*
2. *$\mathrm{Prob}(A = 1) > 0$,*
3. *If $x \in L$, then $\mathrm{Prob}(B = 1|A = 1) \geq 2/3$,*
4. *If $x \notin L$, then $\mathrm{Prob}(B = 0|A = 1) \geq 2/3$.*

Clearly PostBPP $\subseteq$ PostBQP = PP. The argument that NP $\subseteq$ PostBQP actually shows that NP $\subseteq$ PostBPP, but the argument that PostBQP = PP doesn't apply to PostBPP since it relies heavily on quantum superposition and measurement.

So what is the power of PostBPP? It turns out to be equivalent to a somewhat strange class defined in the classical literature under the name $\mathrm{BPP}_{\mathrm{path}}$.

Theorem 1. *PostBPP* $\subseteq$ *BPP*^{NP}.

Proof. Suppose we have an algorithm in PostBPP. We wish to simulate this algorithm in $\mathrm{BPP}^{\mathrm{NP}}$. For a particular run of the PostBPP algorithm, the random bits used are the string r. Some values of r will lead to a valid post-selection, with $A = 1$, and some will have $A = 0$. We can use the NP oracle to find values of r which lead to $A = 1$. In particular, there is some set S_0 of random strings r which lead to $A = 1$. We wish to choose a random string from S_0 and use it to run the PostBPP algorithm. Then the distribution of B under these conditions is exactly $\mathrm{Prob}(B|A = 1)$.

We might try to use the NP oracle using binary search to find a value of r: First pick a random bit r_0 ask the oracle if there is a value of $r \in S_0$ with first bit r_0. This is an NP question, so the oracle can answer it. If there is no such r, flip r_0 and then there must be a solution (since $\mathrm{Prob}(A = 1) > 0$). Then choose random r_1 and ask if there is a value of $r \in S_0$ with first two bits $r_0 r_1$. If not, flip r_1 and continue. Keep choosing random bits and determining if they can be part of r until we have a full string r.

Unfortunately, this strategy doesn't quite work because it doesn't sample r uniformly from S_0. For instance, if there is one string in S_0 that starts with 0 and 10 that start with 1, then this algorithm has a 50% chance of selecting the string that starts with 0, not a 1/11 chance.

The solution is to use a better method of randomly selecting subsets of S_0 to focus our search on. We will define a sequence of subsets S_i narrowing down possible values of r. Membership in S_i will be tested by the NP oracle. To define subset S_i given S_{i-1}, choose a random string a_i (of the same length as r) and a random bit b_i and asking the oracle if there are any strings $r \in S_{i-1}$ such that $a_i \cdot r = b_i$. If so, let $S_i = \{r | r \in S_{i-1}, a_i \cdot r = b_i\}$. Otherwise, let $S_i = \{r | r \in S_{i-1}, a_i \cdot r = b_i \oplus 1\}$. We keep going until S_t contains just a single bit string, which can be determined by linear algebra; with high probability this happens when t is slightly larger than r. (Indeed, if we constrain each random a to be independent of the previous ones, then it automatically happens by the time $t = r$.)

Note that S_i can be tested with the NP oracle, since given r, we can run the original algorithm using the random bits r to determine if $r \in S_0$ (which happens when the output A of the circuit is 1), and we can also check if $a_i \cdot r = b_i$ for each i. Thus, the question, are there any elements of S_{i-1} such that $a_i \cdot r = b_i$ is an NP language and can be decided by an NP oracle.

Lemma 1. *Let S be any set of bit strings length n and let $x \neq y \in S$, and let a be a random n-bit string and b be a random bit. Then the probability that $a \cdot x \oplus b = a \cdot y \oplus b$ is $1/2$. Moreover, the probability that $a \cdot x \oplus b = 0$ is $1/2$.*

Proof of lemma. $a \cdot x \oplus b = a \cdot y \oplus b$ iff $a \cdot (x \oplus y) = 0$. But a is random, and there is always a probability 1/2 that a random string will have dot product 0 with a specific non-zero string. (For instance, pick one non-zero bit of $x \oplus y$; then if a is 1 at that location, we will get one value for $a \cdot (x \oplus y)$, and if a is 0 at that location, we will get the other value. These two possibilities are equally likely.)

Also, whatever the value of $a \cdot x$, a random choice of b guarantees that the distribution of $a \cdot x \oplus b$ is uniform. □

I claim that this process chooses a string uniformly at random from S_0. In particular, given any two $r, r' \in S_0$, they have an equal probability of being chosen. To prove this claim, let us assume for the moment that both r and r' are in S_i. By the lemma, the probability that $a_i \cdot r = b_i$ and $a_i \cdot r' = b_i \oplus 1$ is exactly 1/4, the same as the probability that $a_i \cdot r = b_i \oplus 1$ and $a_i \cdot r' = b_i$. In these two cases, assuming if both are in S_{i-1}, then just one of the two is in S_i (which is certainly non-empty since it contains that string and therefore will pass the NP oracle). It is equally likely that r and r' is the one that survives to be in S_i.

We also have a 1/4 probability that $a_i \cdot r = a_i \cdot r' = b_i$, so if both r and r' are in S_{i-1}, so there is a chance that both of them are in S_i (which is also equal to the chance that exaclty one of them is in S_i). There is

also a 1/4 probability that $a_i \cdot r = a_i \cdot r' = b_i \oplus 1$, but in this case, it might be that there are no values of r such that $a_i \cdot r = b_i$ and $r \in S_{i-1}$, so it is possible that this case will be rejected by the NP oracle.

The probability that r is chosen, given that both r and r' are in S_i is thus

$$\text{Prob}(r \text{ chosen}|r, r' \in S_i) = \frac{1 - \text{Prob}(r, r' \notin S_{i+1})}{3} \left[\text{Prob}(r \text{ chosen}|r, r' \in S_i) + \text{Prob}(r \text{ chosen}|r \in S_{i+1}, r' \notin S_{i+1})\right], \tag{1}$$

and similarly for $\text{Prob}(r' \text{ chosen}|r, r' \in S_i)$ with r and r' switched.

Now, we use induction on i, starting with the *last* i to show that all elements of S_i are equally likely to be chosen. Certainly this is true of the last i since there is only 1 element left. Then the inductive hypothesis tells us

$$\text{Prob}(r \text{ chosen}|r, r' \in S_i) = \text{Prob}(r' \text{ chosen}|r, r' \in S_i). \tag{2}$$

By eqn. (1), we now only need to show that

$$\text{Prob}(r \text{ chosen}|r \in S_{i+1}, r' \notin S_{i+1}) = \text{Prob}(r' \text{ chosen}|r' \in S_{i+1}, r \notin S_{i+1}). \tag{3}$$

The inductive hypothesis tells us that when $r \in S_{i+1}$, the probability of r being chosen is $1/|S_{i+1}|$. So it is sufficient to prove that

$$\text{Prob}(|S_{i+1} = s|r \in S_{i+1}, r' \notin S_{i+1}) = \text{Prob}(|S_{i+1} = s|r \in S_{i+1}, r' \notin S_{i+1}). \tag{4}$$

If this is true, then the distribution of sizes of S_{i+1} is the same in the case when $r \in S_{i+1}$ but $r' \notin S_{i+1}$ and vice-versa, and therefore the probabilities of r or r' being chosen in these cases is the same.

If $r \oplus r' \in S_i$, then $a \cdot (r \oplus r') = 1$ always if exactly one of r and r' is in S_{i+1}. Thus, whether $r \oplus r' \in S_{i+1}$ depends only on the value of b. If $r = 0$ or $r' = 0$, then $r + r'$ is not a new element, so we don't need to think about it, but if $r, r' \neq 0$, then the probability that r is in S_{i+1} but not r' is independent of b and therefore, whether $r \oplus r' \in S_{i+1}$ is independent of whether r or r' is in S_{i+1}. Similarly, if $0 \in S_i$ and $r, r' \neq 0$, then whether $0 \in S_{i+1}$ only depends on b (it is if $b = 0$) and is independent of which of r and r' is in S_{i+1}.

S_i consists of r, r', possibly 0 and/or $r + r'$, and remaining elements which are linearly independent of r and r' (although not necessarily of each other). This means there is some vector $c \neq 0$ such that $c \cdot r = c \cdot r' = 1$ but $c \cdot v = 0$ for $v \in S_i \setminus \{r, r'\}$. Let a be any string, $a' = a \oplus c$, and $T_a = \{v \in S_i \setminus \{r, r'\} | a \cdot v = b\}$. Then $a' \cdot r = a \cdot r \oplus c \cdot r = a \cdot r \oplus 1$, $a' \cdot r' = a \cdot r' \oplus 1$, and $a' \cdot v = a \cdot v \oplus c \cdot v = a \cdot v$ for $v \in S_i \setminus \{r, r'\}$. Thus, $T_{a,b} = T_{a',b}$. Suppose we have some (a, b) for which $r \in S_{i+1}$ but $r' \notin S_{i+1}$, so $S_{i+1} = T_a \cup \{r\}$. Then for (a', b), $S_{i+1} = T_a \cup \{r'\}$. That is, every case where r is in S_{i+1} but r' is not, there is a matched case where $r' \in S_{i+1}$, $r \notin S_{i+1}$, but all other elements of S_{i+1} are the same, and in particular, the size is the same. This proves eqn. (4) and therefore proves the claim that all strings are equally likely to be chosen.

Since the string r is chosen uniformly from S_0, this algorithm correctly simulates the distribution of outcomes of the original PostBPP algorithm and therefore this approach will have the same acceptance probability as the conditional acceptance probability of the PostBPP algorithm, as needed. □

Now, it turns out that BPP is in the second level of the polynomial hierarchy, so $\text{PostBPP}^{\text{NP}}$ is in the third level of the hierarchy, since the NP oracle can be simulated with an extra quantifier.

21.4 Exact Quantum Supremacy

Now, as we have just seen, PostBPP is in the third level of the hierarchy. PostBQP, in contrast, is equal to PP, which contains the whole polynomial hierarchy. There, if PostBQP = PostBPP, then the hierarchy collapses to the third level. To the extent that this is unlikely, that implies that PostBQP and PostBPP are unequal.

What, if anything, does this imply about BQP? Well, suppose there is an exact weak classical simulation of any quantum algorithm. Then you could simulate the algorithm with two qubits measured instead of just one, and in particular have the exactly correct distribution of outputs on qubits A and B in an arbitrary algorithm for a PostBQP language. That means that a probabilistic classical algorithm with post-selection

would be able to decide any language in PostBQP; i.e., we would have PostBQP = PostBPP. Thus, we have the following theorem:

Theorem 2. *If there is an exact weak classical simulation of an arbitrary quantum circuit, then the polynomial hierarchy collapses to the third level.*

This result can be immediately strengthened to show that (unless the polynomial hierarchy collapses), there can be no exact classical simulation of even a number of non-universal quantum models. In particular, any model that can be promoted to universality by adding post-selection works in exactly the same way. Here is a (non-exhaustive) list of examples of non-universal models for which this argument can be made:

- Constant-depth quantum circuits
- Clifford group circuits in a rotated basis
- Linear-optical networks with single-photon states
- Diagonal gates with X basis initial states and measurements
- Random quantum circuits

In the case of random quantum circuits, one first needs to show that being able to simulate a randomly chosen circuit implies being able to simulate any specific circuit (in particular, the hardest ones). Then the result follows from thm. 2. This ability to reduce the worst case problem to the average case problem is very helpful in real-world applications of complexity, including (classical) cryptography and quantum supremacy. This is because finding worst-case instances is itself a hard problem (it is not even clear how to tell if a particular instance is a hard instance or an easy instance), but if randomly chosen instances are very likely to be hard, then there is a straightforward way of generating probably hard instances (not provably hard instances, unfortunately), even if you have no way to be certain you were not unlucky in your specific choice of instance.