Columbia Undergraduate Learning Seminar in TCS Fall 2024

Introducing BQP

Quantum Complexity Theory Presenter: Shiv Kampani

1 Classical complexity theory

The objective of complexity theory is to answer the following question: how much of a "resource" do we need to solve a problem? There are several kinds of "resources" that may be valuable for computation (e.g. time, space, nondeterminism, randomness, quantum weirdness). Before trying to address this computation, it is important to agree on a **model of computation** and the structure or kind of **computational problem** we are dealing with.

There are several kinds of computational models: finite automata, Turing machines, Boolean circuits, decision trees, polynomials, etc. In classical complexity theory, the **multi-tape Turing machine** is considered as the default or standard model of computation. There are a few good reasons for this: (i) models like finite-automata are too weak, (ii) Turing machines given polynomial-time capture the notion of "efficient computation," (iii) Turing machines physically resemble computation in the conventional sense, (iv) Efficient **universal** Turing machines exist (that can simulate other Turing machines).

Like models of computation, there are many different kinds of problems: decision problems, search problems, counting problems. We will focus on **decision problems**:

Definition (Decision Problem)**.** $L \subseteq \{0,1\}^*$. Given $x \in \{0,1\}^*$, output 1 if and only if $x \in L$, and 0 otherwise. Decision problems are YES/NO problems.

Intuition: We will use jigsaw puzzles as a running example to build intuition for decision problems and basic complexity classes.

We can create sets of these decision problems, or equivalently languages, based on how many computational resources they require. This is the idea of complexity classes.

Definition (P)**.** $L \in \mathsf{P}$ if $\exists$ deterministic TM M s.t. M decides L in polynomial-time.

The class P is important because it corresponds to our notion of "efficient computation". Another class EXP is defined similarly, but for exponential-time.

Fact. The computation of any polynomial-time Turing machine can be converted to a polynomial-size Boolean circuit using the basis $\{\text{AND}, \text{NOT}\}$.

Definition (NP)**.** $L \in \mathsf{NP}$ if $\exists$ deterministic TM M s.t. $\forall x$ $\exists$ proof w s.t. $M(x, w) = 1$ if and only if $x \in L$ (and 0 otherwise).

The class NP corresponds to our notion of problems that have "efficiently check-able or verifiable" solutions. This class can also be defined in terms of non-deterministic TMs. Consider the co-class coNP. $L \in \mathsf{coNP} \iff \bar{L} \in \mathsf{NP}$. This corresponds to problems for which we can efficiently verify that a candidate is **not** a valid solution.

Definition (coNP)**.** $L \in \mathsf{coNP}$ if $\exists$ deterministic TM M s.t. $\forall x$ $\forall$ proofs w $M(x, w) = 0$ if and only if $x \in L$ (and 1 otherwise).

It is clear, based on these definitions that NP uses an exists quantifier over proofs, and coNP uses a for-all quantifier. By using more quantifiers, we can define more powerful classes. The polynomial hierarchy PH is a generalization of this idea.

Definition (PSPACE)**.** $L \in \mathsf{P}$ if $\exists$ deterministic TM M s.t. M decides L in polynomial-space.

For PSPACE we do not place any restriction on time. $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PH} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$.

Definition (BPP)**.** $L \in \mathsf{BPP}$ if $\exists$ polynomial-time TM M s.t.

$$x \in L \implies \Pr_r[M(x, r) \text{ accepts}] \geq \frac{2}{3}$$
$$x \notin L \implies \Pr_r[M(x, r) \text{ accepts}] \leq \frac{1}{3}$$

TMs are given the ability to flip coins. It is possible to reduce these error probabilities while still remaining in BPP through "amplification." $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{PH}$.

We have discussed the following computational resources: time, space, non-determinism, randomness. Quantum complexity theory studies the use of "quantum weirdness."

2 Quantum computational model

For quantum complexity theory, however, our model of computation is quantum circuits, as opposed to a quantum Turing machine. There are good (physical) reasons for this choice. Last week, we discussed quantum gates. For our discussions and proofs, it is useful to consider a **universal** basis of gates. We will make use of $\{\text{CCNOT}, \text{H}\}$ (with additional ancilla bits). Is there a better universal basis?

Theorem (Solovay-Kitaev)**.** If $\mathcal{G}$ is universal and closed under inverses, then any unitary U can be approximated upto ε-accuracy using $O(4^n \cdot \text{polylog}(1/\varepsilon))$ gates from $\mathcal{G}$.

The basis $\{\text{CCNOT}, \text{H}\}$ certainly fits these requirements, so can be used to approximate any unitary U with arbitrary precision with "few" gates.

Remark. We will still focus on decision problems. Most functions are hard to compute.

3 Bounding the class BQP

The quantum analogue of P, BQP corresponds to our notion of "efficient computation using quantum computers." Where in our hierarchy does BQP live?

Definition (BQP)**.** $L \in \mathsf{BQP}$ if $\exists$ poly-time uniform family of quantum circuits $(Q_n)_n$ s.t.

$$\forall n\ Q_n \text{ takes } n \text{ input qubits and outputs 1 bit}$$
$$x \in L \implies \Pr[Q_{|x|}(x) = 1] \geq \frac{2}{3}$$
$$x \notin L \implies \Pr[Q_{|x|}(x) = 0] \leq \frac{1}{3}$$

Remark. poly-time uniformity essentially means that given n, you can construct the circuit Q_n in polynomial-time. It prevents us from "cheating" cleverly. Without the uniformity restriction, we could, for example, solve undecidable problems.

Generally, for the 1 bit of output, we measure the first output qubit. The probability of acceptance then is the probability that the first qubit collapses to a 1 state.

Theorem. $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP}$

Proof. First we will show $\mathsf{P} \subseteq \mathsf{BQP}$. For $L \in P$, $\exists$ poly-time TM M that decides L. We can convert the TM to an equivalent poly-size Boolean circuit using the (universal) basis $\{\text{AND}, \text{NOT}\}$. Boolean circuits, unlike quantum circuits, can also have fan-out. To represent this, we can introduce another gate called COPY that "copies" a bit. In order to construct a quantum circuit to decide L, we can reconstruct each Boolean gate using CCNOT gates. The CCNOT gate acts as follows: $\text{CCNOT}(x, y, z) = (x, y, z \oplus xy)$.

- AND gate: $\text{CCNOT}(x, y, 0) = (x, y, xy)$
- NOT gate: $\text{CCNOT}(1, 1, z) = (1, 1, z \oplus 1) = (1, 1, \neg z)$
- COPY gate: $\text{CCNOT}(x, 1, 0) = (x, 1, x)$.

Remark. The reconstruction of COPY does not violate **no cloning**. The copied state is not separable; it is entangled with the original state.
Clearly, $L \in \mathsf{BQP}$ since we implicitly defined a poly-time uniform family of circuits by specifying a procedure for converting a TM into a quantum circuit in poly-time.
Now, we will show $\mathsf{BPP} \subseteq \mathsf{BQP}$. Let $L \in \mathsf{BPP}$. By definition, $\exists$ poly-time TM M that decides L with $\leq 1/3$-error-probability, given access to uniformly random bits. To simulate randomized computation, we need to generate random bits. Assume that we require r random bits. Our quantum circuit can include r ancilla bits in state $|0\rangle$ and apply H to each (and then measure). This produces r uniformly random bits. M, once given random bits, can then be converted to a poly-size quantum circuit as above. $L \in \mathsf{BQP} \implies \mathsf{BPP} \subseteq \mathsf{BQP}$. $\square$

Remark. In our model of computation, we would prefer to do all measurements at the end. To modify our proof, consider the **deferred measurement principle**. There are a few subtleties regarding deferring measurement to the end; we will not focus on these details.

Whether or not this inclusion is **strict** is an open problem. Richard Feynman claimed, in his keynote on "Simulating Physics with Computers," that probabilistic poly-time TMs cannot simulate quantum computers. We do not know this for sure.

Theorem. $\mathsf{BQP} \subseteq \mathsf{EXP}$

Proof. Let $L \in \mathsf{BQP}$. Given any input x, let $Q_{|x|}$ be the quantum circuit that decides whether $x \in L$ or not. In order to prove that $L \in \mathsf{EXP}$, we must define an exponential-time TM that decides L. Define such a TM as follows: on input x, construct quantum circuit $Q_{|x|}$ in poly-time. Compute the matrix product of all polynomial number of $2^{\text{poly}(|x|)} \times 2^{\text{poly}(|x|)}$ unitary matrices in $Q_{|x|}$ and multiply it with the start state of all qubits (and square) to obtain the final probability distribution over states of all qubits. Output 1 if and only if the probability of the first qubit collapsing to $|1\rangle \geq 2/3$. $L \in \mathsf{EXP} \implies \mathsf{BQP} \subseteq \mathsf{EXP}$. □

We can improve our upper bound even further.

Theorem. $\mathsf{BQP} \subseteq \mathsf{PSPACE}$

Proof. Our proof is based on the following idea, which we will call **sum-over-histories**: For exposition, consider $H^2|0\rangle$. We know that this is $|0\rangle$ since the H gate is its own inverse. However, we will compute the product out explicitly and keep basis states in separate branches of a **sum-over-histories** tree. At the first level, we begin with root $|0\rangle$ and children $|0\rangle, |1\rangle$ with amplitudes $\langle 0|H|0\rangle = 1/\sqrt{2}$ and $\langle 1|H|0\rangle = 1/\sqrt{2}$ respectively. We repeat this process at the second level of the tree, multiplying out amplitudes.

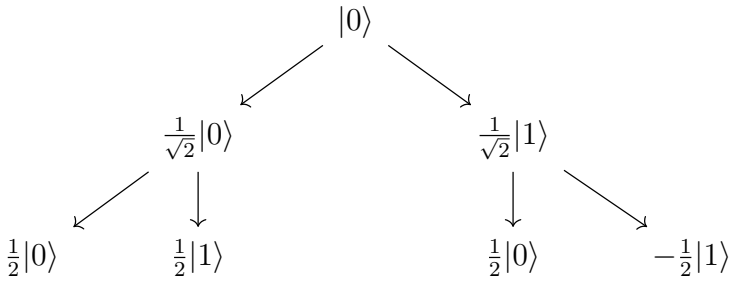

Figure: Sum-over-histories tree for $H^2|0\rangle$

Now, the resulting state is the sum of all leaf-states. Since we are interested in the probability of acceptance, that is, the probability that the first output qubit is measured as 1, it is equal to the square of the amplitude of all states that have the first qubit set to 1, in this case, just $|1\rangle$, with amplitude 0. $0^2 = 0$ probability of a 1 being measured, as expected.

The **sum-over-histories** construction outlined above generalizes to multi-qubit algorithms. Each level of the tree corresponds to the application of one gate at a time. How big is this

tree? Since we have a polynomial number of gates for quantum circuits associated with any $L \in \mathsf{BPP}$, the tree has polynomial depth. The branching factor of the tree is finite, since each basis gate acts on a small, finite number of qubits. Thus, we must have $b^{\text{poly}(n)} = O(2^{\text{poly}(n)})$ possible branches (or leaves) in this tree.

The main idea of our proof is that, while the entire tree is exponentially sized, we can compute each **history** or branch using polynomial space! It is also possible to keep track of which branch we are currently enumerating, by storing a counter, which would require $O(\log 2^{\text{poly}(n)})) = O(\text{poly}(n))$ space, as desired. We are ready to prove the theorem.

Let $L \in \mathsf{BQP}$. Define a polynomial-space TM M that operates as follows: given any input x, compute $Q_{|x|}$ in poly-time (implicitly poly-space). Initialize constant-size counters p for the total probability of acceptances and b for the total amplitude of the current branch. $\forall y \in \{0,1\}^{|x|-1}$ repeat the following: our current check-state is $|1y\rangle$. One at a time, follow all branches of the **sum-over-histories** tree. If the branch terminates at our check-state, add the computed amplitude to our counter b. Re-use space while following branches of the tree. After all branches have been followed, $c \leftarrow c + b^2, b \leftarrow 0$. Repeat the process for the next value of y. After enumerating all y's, output $1 \iff c \geq \frac{2}{3}$ and 0 otherwise.
Clearly, $(M(x) = 1 \iff x \in L) \implies L \in \mathsf{PSPACE} \implies \mathsf{BQP} \subseteq \mathsf{PSPACE}$. □

Remark. $\mathsf{BQP} \subseteq \mathsf{PP}$ (a more powerful generalization of BPP). It can be shown that $\mathsf{PP} \subseteq \mathsf{PSPACE}$; PP is an even better upper bound. The best upper bound (so far) is AWPP.

4 Oracles: providing intution

The picture is far from complete: there are several missing relationships and unanswered questions. Do we know anything about the missing relationships? Why are they difficult to prove? **Oracle separations** give us some intuition for this.
Think of an oracle like you would a black-box that magically produces an answer. Oracles aren't physically realizable, but are still useful to think about because they provide intuition about the real world.

Theorem (Baker-Gill-Solovay)**.** $\exists$ oracles A, B s.t. $\mathsf{P}^A = \mathsf{NP}^A$ and $\underline{\underline{\mathsf{P}^B \neq \mathsf{NP}^B}}$.

Theorem. $\exists$ an oracle O s.t. $\mathsf{BPP}^O \neq \mathsf{BQP}^O$.

Remark. This is closely connected to Simon's problem, for which we have an efficient quantum algorithm, but no efficient classical algorithm exists.

There also exists "oracle-separations" between NP and BQP. A recent (2019) celebrated result in this field proved an oracle separation between PH and BQP.

5 Summary of results

The following inclusion diagram is a summary of everything that we have discussed so far. Solid edges indicate inclusions that we have proved. Dashed edges indicate unproven relationships that we have discussed. Red edges indicate oracle separations (for inclusion).

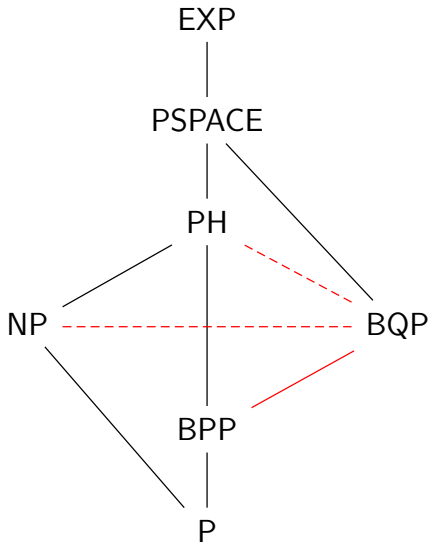

Figure: Inclusion diagram for complexity classes

Concluding Remark. If we prove that $\mathsf{BPP} \neq \mathsf{BQP}$ (i.e. the inclusion is strict), we confirm Feynman's guess. Additionally, this implies that $\mathsf{P} \neq \mathsf{PSPACE}$, an important open problem in classical complexity theory. This is one of the many reasons why it is useful to think about quantum computers despite not knowing how to build scalable ones yet.

References

(1) Grier Lectures 5, 6, 7 (see seminar resources)

(2) Ch. 2, Ch. 13 of de Wolf (see seminar resources)

(3) "Computational Complexity: A Modern Approach" (Arora-Barak)