Modern Complexity Theory (CS1.405)

Dr. Ashok Kumar Das

IEEE Senior Member
Web of Science (Clarivate^{TM}) Highly Cited Researcher 2022, 2023
Professor
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad

E-mail: *ashok.das@iiit.ac.in*
URL: `http://www.iiit.ac.in/people/faculty/ashok-kumar-das`
`https://sites.google.com/view/iitkgpakdas/`

Quantum Complexity Classes

Need of quantum complexity classes

- Quantum complexity classes help to identify problems that quantum computers can solve more efficiently than classical computers.
- By categorizing problems into different complexity classes, we can better understand which problems are inherently easier or harder for quantum computers.
- Understanding the complexity class can lead to optimizations of existing quantum algorithms, making them more efficient.
- Many cryptographic protocols rely on the hardness of certain problems (e.g., factoring large numbers). Understanding how quantum algorithms like Shor's algorithm fit into complexity classes highlights the vulnerabilities of classical cryptography to quantum attacks.

Quantum Complexity Classes

Need of quantum complexity classes (Continued...)

- Knowledge of quantum complexity helps in developing new cryptographic methods that are secure even in a quantum computing era.
- Certain classes of problems, like NP-hard problems, can benefit from quantum approaches, leading to advancements in fields such as logistics, finance, and artificial intelligence.
- By studying complexity classes, researchers can better understand the limitations of quantum computers, including which problems cannot be solved efficiently with quantum algorithms.

Probabilistic Turing Machine (PTM)

Definition

A ***probabilistic Turing machine*** M is a type of nondeterministic Turing machine in which each nondeterministic step is called a ***coin-flip step*** and has two legal next moves. We assign a probability to each branch b of M's computation on input w as follows. Define the probability of branch b to be

$$\Pr[\,b\,] = 2^{-k},$$

where k is the number of coin-flip steps that occur on branch b. Define the probability that M accepts w to be

$$\Pr[M \text{ accepts } w] = \sum_{\substack{b \text{ is an} \\ \text{accepting branch}}} \Pr[\,b\,].$$

In other words, the probability that M accepts w is the probability that we would reach an accepting configuration if we simulated M on w by flipping a coin to determine which move to follow at each coin-flip step. We let

$$\Pr[M \text{ rejects } w] = 1 - \Pr[M \text{ accepts } w].$$

Probabilistic Turing Machine (PTM)

When a probabilistic Turing machine recognizes a language, it must accept all strings in the language and reject all strings out of the language as usual, except that now we allow the machine a small probability of error. For $0 \leq \epsilon < \frac{1}{2}$ we say that ***M recognizes language A with error probability*** ϵ if

1. $w \in A$ implies $\Pr[M \text{ accepts } w] \geq 1 - \epsilon$, and
2. $w \notin A$ implies $\Pr[M \text{ rejects } w] \geq 1 - \epsilon$.

In other words, the probability that we would obtain the wrong answer by simulating M is at most ϵ. We also consider error probability bounds that depend on the input length n. For example, error probability $\epsilon = 2^{-n}$ indicates an exponentially small probability of error.

Probabilistic Turing Machine (PTM)

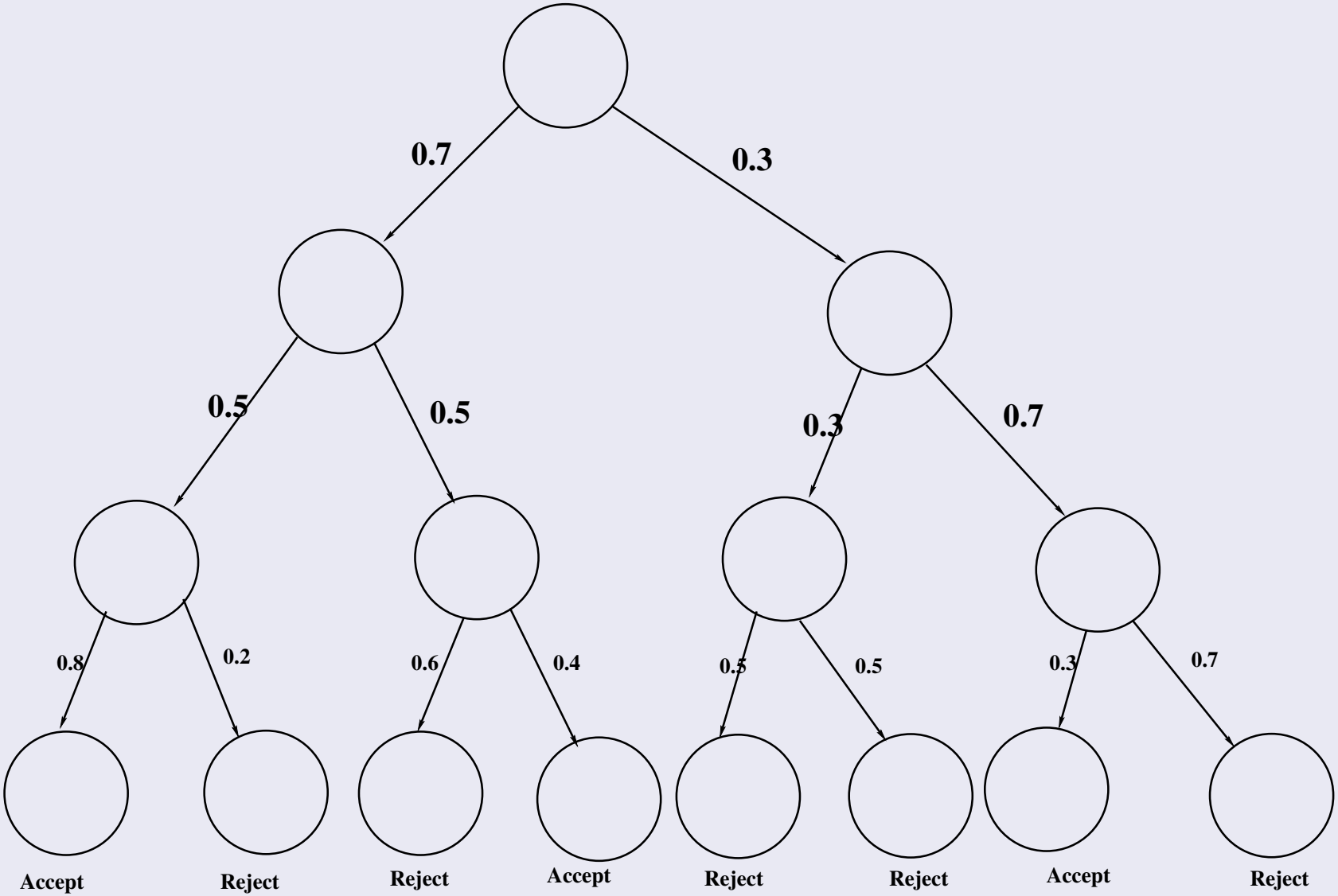

Figure: An example of Probabilistic Turing Machine (PTM)

Bounded-error Probabilistic Polynomial time (BPP)

- In computational complexity theory, bounded-error probabilistic polynomial time (BPP) is the class of decision problems solvable by a probabilistic Turing machine (PTM) in polynomial time with an error probability bounded by $1/3$ for all instances.

Definition (BPP)

A language L is in BPP if and only if there exists a PTM, say M, such that

- M runs for polynomial time on all inputs
- For all x in L, M outputs 1 with probability greater than or equal to $2/3$
- For all x not in L, M outputs 1 with probability less than or equal to $1/3$

Bounded-error Quantum Polynomial time (BQP)

- In quantum information theory, a **quantum circuit (QC)** is a model for quantum computation, similar to classical circuits, in which a computation is a sequence of quantum gates, measurements, and initializations of qubits to known values.
- Bounded-error quantum polynomial time (BQP) is quantum analogue to BPP class. BQP describes the class of problems that can be solved on a quantum computer in polynomial time with a bounded probability of error.

Definition (BQP)

A language L is in BQP if there exists a (polynomial-time) uniform family of quantum circuits $\{QC_n\}$ such that for input $x \in \{0,1\}^n$:

- If $x \in L$, then QC_n accepts with probability at least $2/3$.
- If $x \notin L$, then QC_n accepts with probability at most $1/3$.

Proof Outline

- **Given a BPP Algorithm:** Let $L \in BPP$ be a language for which there exists a probabilistic algorithm A that runs in polynomial time $p(n)$ (where n is the size of the input x, that is, $|x| = n$). By definition, we have:

$$P(A(x) = 1 \mid x \in L) \geq \frac{2}{3}$$

$$P(A(x) = 0 \mid x \notin L) \geq \frac{2}{3}$$

- **Construct a Quantum Algorithm:** We will construct a quantum algorithm B that simulates the probabilistic algorithm A as follows:
  - The algorithm A makes use of random bits. In a quantum setting, we can generate random bits using qubits.

Proof Outline (continued)

- For k random bits, we can represent them using k qubits, each initialized to $|0\rangle$ and then transformed into superposition:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2^k}} \sum_{b \in \{0,1\}^k} |b\rangle$$

- For each possible random input, we will create a quantum version of the decision process. If A uses m random bits, the quantum algorithm can evaluate the result of A for all 2^m possible random outcomes simultaneously.
- We will utilize amplitude amplification to enhance the probability of measuring the correct outcome. We will construct a quantum oracle that simulates the function of A.

Proof Outline (continued)

- **Quantum Decision Process:** Start with the input $|x\rangle$ and the superposition of random bits $|b\rangle$. The quantum algorithm computes $A(x, b)$ for all b simultaneously using quantum superposition. Upon measuring the output, we focus on the states that correspond to accepting 1 and rejecting 0. The measurement will yield 1 with high probability due to amplitude amplification.
- **Error Probability Analysis:**
  - If $x \in L$, then $A(x)$ accepts with probability at least $2/3$. Therefore, the quantum algorithm B can be constructed to accept with probability $P(B(x) = 1 \mid x \in L) \geq 2/3$.
  - If $x \notin L$, then $A(x)$ rejects with probability at least $2/3$. Hence, we have $P(B(x) = 0 \mid x \notin L) \geq 2/3$.

Hence, we can conclude that: $BPP \subseteq BQP$

Other probabilistic complexity classes

- In complexity theory, PP or PPT is the class of decision problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than $\frac{1}{2}$ for all instances. The abbreviation PP refers to probabilistic polynomial time.

Definition (Class PP)

A language L is in PP if and only if there exists a probabilistic Turing machine M, such that

- M runs for polynomial time on all inputs
- For all x in L, M outputs 1 with probability strictly greater than $1/2$
- For all x not in L, M outputs 1 with probability strictly less than $1/2$.

Class ZPP

Definition (ZPP)

ZPP can be defined as the class of problems for which a probabilistic Turing machine exists with these properties:

- It always runs in polynomial time.
- It returns an answer **YES**, **NO** or **DO NOT KNOW**.
- The answer is always either **DO NOT KNOW** or the correct answer.
- It returns **DO NOT KNOW** with probability at most $1/2$ for every input (and the correct answer otherwise).

Class RP and coRP

Definition (RP)

A language $L \subseteq \{0,1\}^*$ is in RP if and only if there is a probabilistic polynomial time Turing machine M such that

- $x \in L \implies P(M(x) = 1) \geq \frac{1}{2}$
- $x \notin L \implies P(M(x) = 0) = 1$

Definition (coRP)

A language $L \subseteq \{0,1\}^*$ is in $coRP$ if $\bar{L} \in RP$. Equivalently, there is a probabilistic polynomial time machine M such that

- $x \in L \implies P(M(x) = 1) = 1$
- $x \notin L \implies P(M(x) = 0) \geq \frac{1}{2}$

BQP with other probabilistic complexity classes

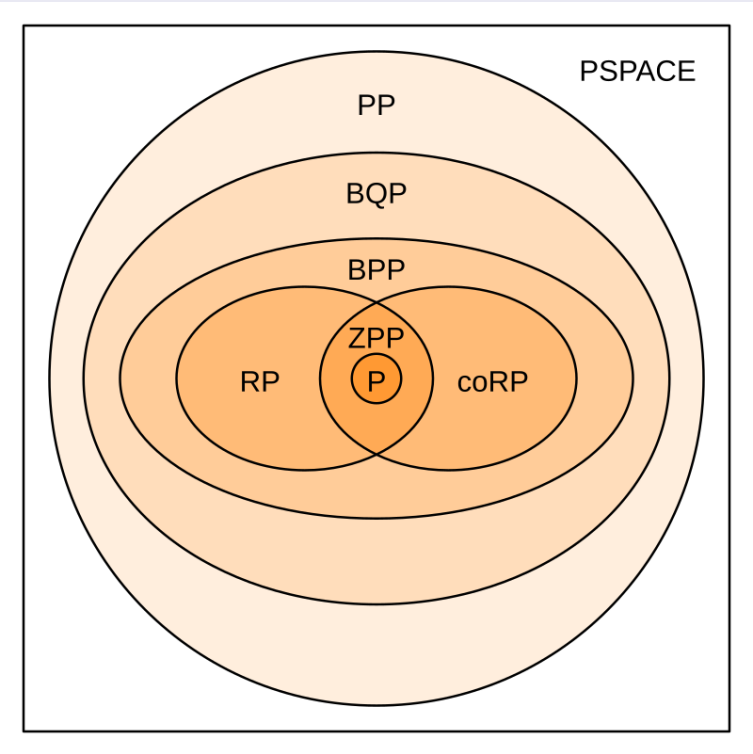

Figure: BQP in relation to other probabilistic complexity classes (ZPP, RP, co-RP, BPP, PP)

$$P \subseteq BPP \subseteq BQP \subseteq PP \subseteq PSPACE \subseteq EXP$$

BQP with P and NP class

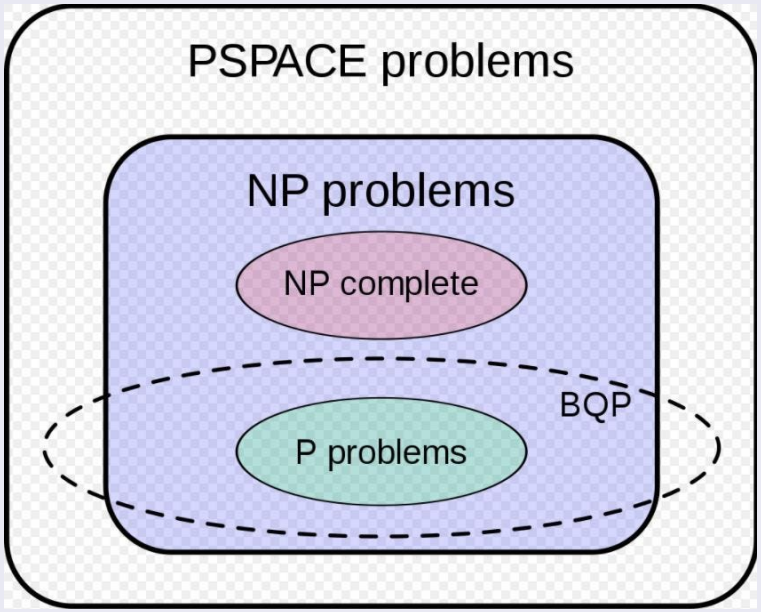

Figure: The suspected relationship of BQP with other probabilistic complexity classes (P, NP)

Open problem

Is $NP \subseteq BQP$?

BQP: Example

Integer factorization

A real-life example of a problem that belongs to the BQP complexity class is **Shor's algorithm** for integer factorization. Shor's algorithm, when run on a quantum computer, can factor large integers efficiently, which is a task that is extremely difficult for classical computers, especially for large numbers used in widely-used public-key cryptosystem (like RSA encryption).

Shor's algorithm: Problem statement

Given a large composite integer N, find its prime factors.

- **Input:** A large composite number N.
- **Output:** Non-trivial prime factors p and q such that $N = p \times q$.

Shor's algorithm

Steps:

It contains a few steps, where only at step 2 the use of quantum computers is required.

1. Choose any random number let say r, such that $r < N$ so that they are co-primes of each other.
2. A quantum computer is used to determine the unknown period a of the function $f_{r,N}(x) = r^x \mod N$.
3. If a is an odd integer, then go back to Step 1. Else move to the next step.
4. Since a is an even integer so, $(r^{\frac{a}{2}} - 1)(r^{\frac{a}{2}} + 1) = r^a - 1 = 0 \mod N$.
5. Now, if the value of $(r^{\frac{a}{2}} + 1) = 0 \mod N$, go back to Step 1. Else move to the next step.
6. Compute p $= \gcd(r^{\frac{p}{2}} - 1, N)$.
7. The answer required is p.

Time complexity of Shor's algorithm

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Time complexity

- **Classical Factorization:** Exponential time $O(2^{\log N})$
- **Shor's algorithm:** $O((\log N)^3)$, where N is the integer to be factored.

- Shor's algorithm solves the integer factorization problem in polynomial time on a quantum computer. Specifically, the runtime of Shor's algorithm scales as $O((\log N)^3)$.
- Shor's algorithm is a probabilistic algorithm with a bounded error. Like other algorithms in BQP, it doesn't guarantee the correct answer with 100% certainty in a single run. However, the probability of getting the wrong answer can be made arbitrarily small by running the algorithm multiple times. This makes Shor's algorithm an example of a bounded-error quantum algorithm.

Other example of BQP

Problem: Unstructured Search (Grover's Algorithm)

Given a database of N unsorted elements and a black-box function (or oracle) $f(x)$, where $f(x) = 1$ for a marked (desired) element and $f(x) = 0$ for all other elements, find the marked element.

Formal statement

- **Input:** A black-box (oracle) function $f(x)$, where $x \in \{0, 1, 2, \ldots, N-1\}$, such that $f(x) = 1$ if x is the desired element (solution) and $f(x) = 0$ otherwise. The input can also be viewed as an unsorted list of N elements with one or more solutions.
- **Output:** The value x such that $f(x) = 1$ (i.e., find the marked element).

Classical vs. Quantum time complexity

Time complexity

- **Classical Approach**: In the worst case, you would need to check all N elements one by one, leading to a time complexity of $O(N)$, which is exponential.
- **Quantum Approach (Grover's Algorithm)**: Grover's algorithm reduces this search time to $O(\sqrt{N})$, providing a quadratic speedup.

Key Steps in Grover's Algorithm

Steps:

- **Initialize a superposition**: Begin by initializing the quantum system to be in an equal superposition of all possible states.
- **Oracle Query**: Apply the oracle function to mark the desired element by flipping its amplitude.
- **Amplitude Amplification**: Use the Grover diffusion operator to amplify the probability amplitude of the correct solution.
- **Measurement**: After $O(\sqrt{N})$ iterations, measure the system to obtain the correct element with high probability.

Why is Grover's Algorithm in BQP?

- Grover's algorithm achieves speedup by using the principles of quantum superposition and amplitude amplification, resulting in a square root speedup compared to classical search algorithms.
- Like other algorithms in the BQP class, Grover's algorithm is probabilistic and has bounded error. After a single application of the algorithm, the probability of finding the correct answer is not 100%, but the probability of success increases significantly with repeated iterations. By running the algorithm multiple times, the error can be reduced, so the likelihood of obtaining the correct answer can be made arbitrarily close to 1.
- Grover's algorithm is in the BQP class because it solves unstructured search problems in polynomial time with bounded error on a quantum computer.

End of this lecture