Urmila Mahadev's work on classical verification of quantum computations

Jaikumar Radhakrishnan

Tata Institute of Fundamental Research, Mumbai

February 7, 2019

Urmila Mahadev, PhD student, UC Berkeley

- Urmila Mahadev: Classical Verification of Quantum Computations. FOCS 2018: 259-267. (Best paper and best student paper!)
- Urmila Mahadev: Classical Verification of Quantum Computations. https://arxiv.org/abs/1804.01082
- Urmila Mahadev: Classical Verification of Quantum Computations. Talk at IAS: https://www.youtube.com/watch?v=kql5dSywvy0

Classical computation

- Inputs and outputs: $\{0,1\}^n$
- Gates: ANDs, ORs and NOTs
- The computation is efficient if it uses only a polynomial number of basic gates.
- Randomized gates: R, generates a uniformly random bit
- Allow errors but would like the output to be correct with high probability

Verifying classical computation

- Suppose some computing device claims to solve a large Hamilton cycle instance.
- We do not have the means to solve and verify this claim ourselves.
- But we can still verify the claim efficiently.
- The problem is in NP. So, we can reduce it to a 3-SAT instance and ask the solver to provide us a satisfying assignment.

Quantum computation

- Similar to randomized computation, but it is reversible.
- The Hadamard gate is like a random coin toss, but the input influences the sign of the amplitude. It is usually written as a matrix.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

  H is its own inverse.
- In general, the state of an n-qubit system is an amplitude vector of 2^n entries. At each step, when a gate is applied, the change in the state vector is determined by a unitary matrix.
- We will need (for today) two special one-qubit gates:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{and} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The power of quantum

- We know of some problems that one can solve more easily on a quantum computer.
- A number can be factored in polynomial time on a quantum algorithm. (Shor 1994).
- Quantum cryptography can do things that classical cryptography can't.
- But we do not yet have even a decent quantum computer . . .

What if . . . ?

- . . . quantum computers of moderate size perform certain tasks more efficiently than classical computers?
- Noisy Intermediate-Scale Quantum Technology
- Quantum supremacy
- Can we verify that these quantum computers are indeed solving the problem faster?
- Note the problem they solve may not be in NP.

Urmila Mahadev's work

Theorem

Assuming the existence of an extended trapdoor claw-free family, all decision problems that can be efficiently computed in quantum polynomial time can be verified by an efficient classical machine by interacting with the quantum machine.

A family of extended trapdoor claw-free function can be constructed under the assumption that the problem of learning with errors is hard for quantum computers.

Learning with errors

- Two special distributions on matrix-vector pairs.
- $(A, As + e \pmod q)$ where A is a random $n \times m$ matrix and s is a random m-dimensional vector and e is drawn from a special trucated Gaussian distribution on m-dimensional vectors.
- (A, u), where u is uniformly chosen vector.

- The two distributions are far apart statistically.
- Assumption: No quantum polynomial-time procedure can distinguish these distributions with even negligible advantage.

Based on the hardness of the LWE problem, one can construct a family of extended trapdoor claw-free functions.

Trapdoor functions

Trapdoor claw-free functions

A family $\mathcal{F} = \{f_{k,b} : \mathcal{X} \to \mathcal{Y}\}$ such that the functions $f_{k,0}$ and $f_{k,1}$ are injective and their images are identical. It is computationally hard given k to find a string $d \neq 0$, the bit $d \cdot (x_0 \oplus x_1)$, where $f_0(x_0) = f_1(x_1)$. The trapdoor t_k allows one to invert the function.

Trapdoor injective functions

A family $\mathcal{G} = \{g_{k,b} : \mathcal{X} \to \mathcal{Y}\}$, such that the images of $g_{k,0}$ and $g_{k,1}$ are disjoint. The trapdoor t_k helps invert the function.

The keys k for claw-free and injective functions are computationally indistinguishable.

Trapdoor functions aided measurement

Prover	Has a state $\|\phi\rangle = \alpha_0 \|0\rangle + \alpha_1 \|1\rangle$.
Verifier	Sends a key k (for either a claw-free function or an injective function), but holds on to the trapdoor t_k
Prover	Would like to help the verfier make a measurement of $\|\phi\rangle$. Prepares $\|\phi\rangle \sum_x \|x\rangle$. Appends the bit in the first register to the key and computes the function of x, and sends the result y to the Verifier.
Verfier	The Verifier either asks for a standard basis measurement or a Hadamard measurement of all of Prover's registers.
Prover	Sends the bits to the Verifier.
Verifier	Decodes the result.

The verifier cannon cheat

- If the Prover does pass the Verifier's checks with reasonable probability, then the bits the Verfier receives do correspond to measurement of an underlying quantum state.
- These measurement outcomes can be used by the Verifier to check if the quantum computer's claims are justified.

Thank you!