Quantum Computing: Clifford circuits on Classical Computers

Aeriana Narbonne
University of King's College, Halifax Nova Scotia

PART 1 : overview of Quantum computation

Classical computers use *bits* with discrete values of 0 and 1 which interact through discrete operations, which we call *gates*. These classical bits are made up of a large number of atoms, so they behave as macroscopic *classical* objects. Alternatively, if we were to use a smaller number of atoms, then these bits would behave as macroscopic *quantum* objects. What this would mean is that instead of the classical way of a bit being in one position or the other (0 or 1), the bits would be in a superposition of 0 and 1 and it would be unknown which until measurement. We call these bits "quantum bits" or "qubits".

Classical and Quantum gates are often represented as 2x2 matrices. Common classical gates include the NOT, AND (not quantum gate), OR, as well as the XOR gates, where we compute the result of the interactions of these bits in the same manner as we do boolean truth tables. Quantum gates encompass some of the classical gates (but not all) as well as others. In this talk, we will focus mainly on the Pauli gates: X, Y, ad Z.

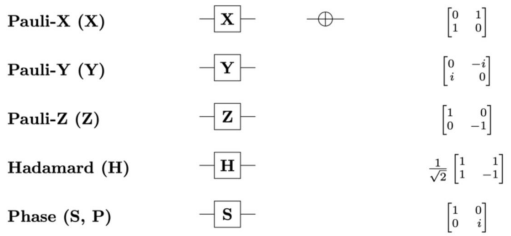
PART 2: The importance of using clifford circuits now

But why would we want to simulate these circuits on a classical computer? Surely if it is our goal to build a quantum computer, should we not be moving away from the classical computer and building quantum circuits without the need to simulate them? If we were to build a quantum computer and it didn't work, there would virtually be no way to find the problem. On a classical computer, we can debug circuits by adding test conditions, monitoring registers, halting at intermediate steps. However, doing so on a quantum computer would probably require measurement and destroy the system.

Why we would want specifically Clifford circuits
Though there are currently programs such as QuIDD, the calculations are slow. The package took more than 22 hours to simulate Grover's algorithm on 40 qubits.

PART 3 : overview of clifford circuits

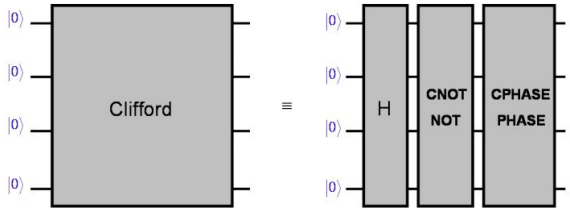
Clifford gates are permutations of the Pauli group. So we start with Pauli X, Y, and Z; from these we can get the Hadamard gate, Controlled-not gate, and phase gates. Clifford circuits can generate a high degree of entanglement.



PART 4 : Gottesman-Knill Theorem

Gottesman-Knill Theorem *Every (uniform family of) Clifford circuits, when applied to the input state |0> and followed by a Z measurement of the first qubit, can be efficiently simulated classically in the strong sense.*

The input starte, |0> N times, followed by Z measurement. When we say "in the strong sense", we mean to "compute the probabilities of the output measurement efficiently classically with high accuracy" .



PART 5 : implementing quantum circuits on classical computers

Haskell is a functional programming language, which a couple researchers have used to create an interface for quantum circuits. One example is the Quantum IO monad. These particular researchers chose this language to be able to exploit the existing means of abstraction present in Haskell to structure their quantum programs.

PART 6 : Applications:

The preference of quantum algorithms over classical algorithms is largely due to the efficiency of the quantum algorithm. All classical algorithms can be run on quantum computers after they have been cast into reversible algorithms. However, quantum computers do not uniformly speed up all classical algorithms.

One of the major applications of quantum computing is Shor's algorithm which takes in a number N and finds N = pq such that p and q are large primes. Shor's algorithm to factors numbers is significantly faster than classical methods. In fact, the fastest classical algorithm for actually finding the prime divisors of N takes exp(O(n1/3)) steps, while Shor's quantum algorithm took only O(n3) steps, later improved to O(n2log[n]).

References

Gottesman, Daniel. (2008). The Heisenberg Representation of Quantum Computers. https://arxiv.org/pdf/quant-ph/9807006.pdf

Gottesman, Daniel and Aaronson, Scott. (2008). Improved Simulation of Stabilizer Circuits. https://arxiv.org/pdf/quant-ph/0406196.pdf

Green, A. & Altenkirch, T. (2008). Shor in Haskell The Quantum IO Monad http://www.cs.nott.ac.uk/~psztxa/publ/qio.pdf

Lanzagorta, M and Uhlmann, J. (2009). Quantum Computer Science. Morgan & Claypool Publishers.

National Academies of Sciences, Engineering, and Medicine. 2019. Quantum Computing: Progress and Prospects. The National Academies Press, Washington, DC. doi: https://doi.org/10.17226/25196.

Van den Nest, Maarten. (2009). Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. https://arxiv.org/pdf/0811.0898.pdf

Viamontes et al. (2003). Improving Gate-Level Simulation of Quantum Circuits. https://arxiv.org/abs/quant-ph/0309060

"Quantum logic gate," n.d.