A Review: Quantum Computing Algorithms and its Applications

Usha Rani J, Anandhi G
Assistant Professors, Dept.of CSE, GSSS Institute of Engineering and Technology for Women, Mysuru
Affiliated to Visvesvaraya Technological University, Belgaum, Karnataka.

Abstract: Quantum Computing is a computing technique in which computers use the laws of quantum mechanics for processing information. Quantum Computers are different from digital electronic computers where quantum computers use quantum bits known as qubits for processing. Many researches are being conducted to deploy a powerful quantum computer. Once built, these computers will be having the ability to solve complex computational problems which cannot be done by classical computers.
Problems which are undecidable using classical computers algorithm remain undecided, but using quantum computers, quantum algorithms might be able to solve some problems faster than classical algorithms because the quantum superposition and quantum entanglement that quantum algorithms exploit probably can't be efficiently simulated on classical computers -Quantum supremacy.

Index Terms - **qubits, Quantum algorithms, Quantum supremacy.**

I.INTRODUCTION

Classical computers are not able to simulate the systems on the basis of sub-exponential time and space complexity due to the enormous growth of the data required to completely represent a quantum system. Comparatively Quantum computers, exploit the unique, non-classical properties of the quantum systems, allowing them to process exponentially large quantities of data only in polynomial time. The study of quantum algorithms has emerged greatly from simulating quantum physical systems to wide variety of fields, including information theory, cryptography, language theory, and mathematics.

Quantum Computers work on quantum algorithms to solve many crucial problems which cannot be solved by classical computing algorithms. The most frequently used algorithms are ***Shor's algorithm*** for factoring, and Grover's algorithm for searching an unstructured database or an unordered list. Shor's algorithms runs much (exponentially) faster than the classical algorithm for factoring. ***Grover's algorithm*** runs quadratically faster than the classical algorithm for a linear search. ***Simon's algorithm*** solves a black-box problem exponentially faster than any classical algorithm, including bounded-error probabilistic algorithms. This algorithm is efficient in speeding up exponentially over all classical algorithms. This was the motivation for Shor's factoring algorithm. Issues like security; hardness which lies in the core of public key cryptosystems generated by classical algorithms is solved with serious concern by quantum algorithms to protect society's digital data.

II.METHODOLOGY: QUANTUM ALGORITHMS FOR IDEAL QUANTUM COMPUTERS

A.Shor's algorithm: Factoring

A quantum algorithm for factoring a number N in O((log N)3) time and O(log N) space, named after Peter Shor. The algorithm is significant because it implies that public key cryptography might be easily broken, given a sufficiently large quantum computer. RSA, for example, uses a public key N which is the product of two large prime numbers. One way to crack RSA encryption is by factoring N, but with classical algorithms, factoring becomes increasingly time-consuming as N grows large; more specifically, no classical algorithm is known that can factor in time O((log N)k) for any k. By contrast, Shor's algorithm can crack RSA in polynomial time. It has also been extended to attack many other public key cryptosystems. Shor's algorithm is probabilistic: it gives the correct answer with high probability, and the probability of failure can be decreased by repeating the algorithm.

Procedure

The problem trying to solve is that, given an integer N, we try to find another integer p between 1 and N that divides N.

Shor's algorithm consists of two parts:

1. A reduction of the factoring problem to the problem of order-finding, which can be done on a classical computer.
2. A quantum algorithm to solve the order-finding problem.

Classical part

1. Pick a pseudo-random number $a < N$
2. Compute gcd(a, N). This may be done using the Euclidean algorithm.
3. If gcd(a, N) $\neq 1$, then there is a nontrivial factor of N, so we are done.
4. Otherwise, use the period-finding subroutine (below) to find r, the period of the following function: $f(x) = a^x \bmod N$
   , i.e. the smallest integer r for which $f(x + r) = f(x)$.
5. If r is odd, go back to step 1.

6. If $a\ r/2 \equiv -1 \pmod N$, go back to step 1.
7. The factors of N are gcd($ar/2 \pm 1$, N). We are done.

Quantum part: Period-finding subroutine:

1. Start with a pair of input and output qubit registers with log2N qubits each, and initialize them to $N^{-1/2}\sum_x|x\rangle|0\rangle$ where x runs from 0 to N - 1.
2. Construct $f(x)$ as a quantum function and apply it to the above state, to obtain $N^{-1/2}\sum_x|x\rangle|f(x)\rangle$
3. Apply the quantum Fourier transform on the input register. The quantum Fourier transform on N points is defined by:
$$U_{QFT}|x\rangle = N^{-1/2}\sum_y e^{2\pi ixy/N}|y\rangle$$
This leaves us in the following state:
$$N^{-1}\sum_x\sum_y e^{2\pi ixy/N}|y\rangle|f(x)\rangle$$
4. Perform a measurement. We obtain some outcome y in the input register and $f(x_0)$ in the output register. Since f is periodic, the probability to measure some y is given by
$$N^{-1}|\sum_{x\,:\,f(x)=f(x0)} e^{2\pi ixy/N}|^2 = N^{-1}|\sum_b e^{2\pi i(x_0+rb)y/N}|^2$$
Analysis now shows that this probability is higher, the closer yr/N is to an integer.
5. Turn y/N into an irreducible fraction, and extract the denominator r', which is a candidate for r.
6. Check if $f(x) = f(x + r')$. If so, we are done.
7. Otherwise, obtain more candidates for r by using values near y, or multiples of r'. If any candidate works, we are done.
8. Otherwise, go back to step 1 of the subroutine.

B.Grovers Search Algorithm: Unsorted Database

Grover's algorithm is a quantum algorithm for searching an unsorted database with N entries in *O(N1/2)* time and using *O(*log*N)* storage space (see big O notation). It was invented by Lov Grover in 1996. Classically, searching an unsorted database requires a linear search, which is *O(N)* in time. Grover's algorithm, which takes *O(N1/2)* time, is the fastest possible quantum algorithm for searching an unsorted database. It provides "only" a quadratic speedup, unlike other quantum algorithms, which can provide exponential speedup over their classical counterparts. However, even quadratic speedup is considerable when N is large.

Like all quantum computer algorithms, Grover's algorithm is probabilistic, in the sense that it gives the correct answer with high probability. The probability of failure can be decreased by repeating the algorithm.

Consider an unsorted database with N entries. The algorithm requires an N-dimensional state space H, which can be supplied by log$2N$ qubits.

Let us number the database entries by 0, 1, ... (N-1). Choose an observable, Ω, acting on H, with N distinct eigenvalues whose values are all known. Each of the eigenstates of Ω encode one of the entries in the database, in a manner that we will describe. Denote the eigenstates (using bra-ket notation) as
$$\{|0\rangle, |1\rangle, \cdots, |N-1\rangle\}$$
and the corresponding eigenvalues by
$$\{\lambda_0, \lambda_1, \cdots, \lambda_{N-1}\}$$

We are provided with a unitary operator, $U\omega$, which acts as a subroutine that compares database entries according to some search criterion. The algorithm does not specify how this subroutine works, but it must be a *quantum* subroutine that works with superpositions of states. Furthermore, it must act specially on one of the eigenstates, |ω>, which corresponds to the database entry matching the search criterion. To be precise, we require $U\omega$ to have the following effects:
$$U_\omega|\omega\rangle = -|\omega\rangle$$
$$U_\omega|x\rangle = |x\rangle \qquad \text{for all } x \neq \omega$$
Our goal is to identify this eigenstate |ω>, or equivalently the eigenvalue ω, that Uω acts specially upon.

Steps of the algorithm

The steps of Grover's algorithm are as follows:

1. Initialize the system to the state |s\ranglele = \frac{1}{\sqrt{N}} \sum_x |x\ranglele
2. Perform the following "Grover iteration" *r(N)* times. The function *r(N)* is described below.
    a. Apply the operator U_ω
    b. Apply the operator $U_s = 2|s\rangle le\langle les| - I$.
3. Perform the measurement Ω. The measurement result will be λω with probability approaching 1 for N>>1. From λω, ω may be obtained.

Explanation of the algorithm

Our initial state is

|s⟩=1N−−√∑x|x⟩|s⟩=1N∑x|x⟩

Consider the plane spanned by |s> and |ω>. Let |ω×> be a ket in this plane perpendicular to |ω>. Since |ω> is one of the basis vectors, the overlap is

⟨ω|s⟩=1N−−√⟨ω|s⟩=1N

In geometric terms, there is an angle (π/2 - θ) between |ω> and |s>, where θ is given by:

cos(π2−θ)=1N−−√cos(π2−θ)=1N
sinθ=1N−−√sinθ=1N

The operator Uω is a reflection at the hyperplane orthogonal to |ω>; for vectors in the plane spanned by |s> and |ω>, it acts as a reflection at the line through |ω×>. The operator Us is a reflection at the line through |s>. Therefore, the state vector remains in the plane spanned by |s> and |ω> after each application of Us and after each application of Uω, and it is straightforward to check that the operator UsUω of each Grover iteration step rotates the state vector by an angle of 2θ toward |ω>.
We need to stop when the state vector passes close to |ω>; after this, subsequent iterations rotate the state vector *away* from |ω>, reducing the probability of obtaining the correct answer. The number of times to iterate is given by *r*. In order to align the state vector exactly with |ω>, we need:

π2−θ=2θrπ2−θ=2θr
r=(πθ−2)4r=(πθ−2)4

However, *r* must be an integer, so generally we can only set *r* to be the integer closest to (π/θ - 2)/4. The angle between |ω> and the final state vector is O(θ), so the probability of obtaining the wrong answer is O(1 - cos2θ) = O(sin2θ).
For N>>1, θ ≈ N-1/2, so

r→πN−−√4r→πN4

Furthermore, the probability of obtaining the wrong answer becomes O(1/N), which goes to zero for large N.

C.Simon’s Algorithm: Exponential solution

Simon's algorithm was the first quantum algorithm to show an exponential speed-up versus the best classical algorithm in solving a specific problem. This inspired the quantum algorithm for the discrete Fourier transform, also known as quantum Fourier transform, which is used in the most famous quantum algorithm: Shor's factoring algorithm.
Simon’s Problem
We are given an unknown blackbox function ff, which is guaranteed to be either one-to-one or two-to-one, where one-to-one and two-to-one functions have the following properties:

- *one-to-one*: maps exactly one unique output for every input,
- eg. F(1)→1f(1)→1, f(2)→2f(2)→2, f(3)→3f(3)→3, f(4)→4f(4)→4.
- *two-to-one*: maps exactly two inputs to every unique output, eg. F(1)→1f(1)→1, f(2)→2f(2)→2, f(3)→1f(3)→1, f(4)→2f(4)→2, according to a hidden bitstring, ss

where: given x1,x2:f(x1)=f(x2)it is guaranteed :x1⊕x2=swhere: given x1,x2:f(x1)=f(x2)it is guaranteed :x1⊕x2=s
Thus, given this blackbox ff, how quickly can we determine if ff is one-to-one or two-to-one? Then, if ff turns out to be two-to-one, how quickly can we determine ss? As it turns out, both cases boil down to the same problem of finding ss, where a bitstring of s=000...s=000... represents the one-to-one ff.
Classical Solution
Classically, if we want to know what ss is for a given ff, with 100% certainty, we have to check up to 2N−1+12N−1+1 inputs, where N is the number of bits in the input. This means checking just over half of all the possible inputs until we find two cases of the same output. Although, probabilistically the average number of inputs will be closer to the order of (o)(2)(o)(2). Much like the Deutsch-Jozsa problem, if we get lucky, we could solve the problem with our first two tries. But if we happen to get an ff that is one-to-one, or get *really* unlucky with an ff that’s two-to-one, then we’re stuck with the full 2N−1+12N−1+1.

Quantum Solution

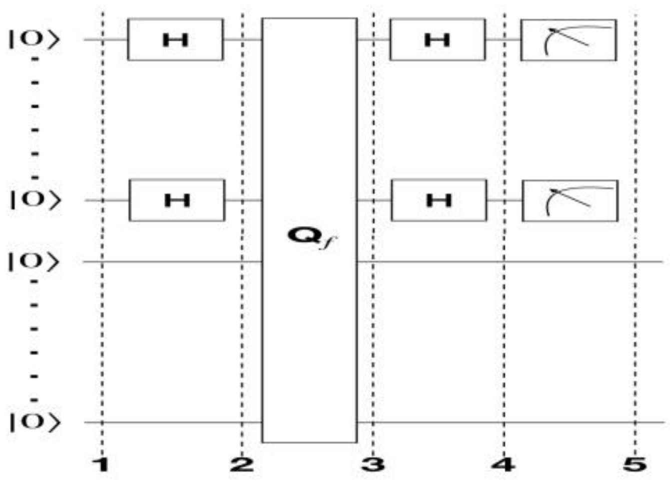


Fig: 2.1 *The quantum circuit that implements Simon's algorithm*

Where the query function, QfQf acts on two quantum registers as:
|x⟩|0⟩→|x⟩|f(x)⟩|x⟩|0⟩→|x⟩|f(x)⟩
The algorithm involves the following steps.

1. Two nn-qubit input registers are initialized to the zero state:
   |ψ1⟩=|0⟩⊗n|0⟩⊗n|ψ1⟩=|0⟩⊗n|0⟩⊗n
2. Apply a Hadamard transform to the first register:
   |ψ2⟩=12n−−√∑x∈{0,1}n|x⟩|0⟩⊗n|ψ2⟩=12n∑x∈{0,1}n|x⟩|0⟩⊗n
3. Apply the query function QfQf:
   |ψ3⟩=12n−−√∑x∈{0,1}n|x⟩|f(x)⟩|ψ3⟩=12n∑x∈{0,1}n|x⟩|f(x)⟩
4. Measure the second register. A certain value of f(x)f(x) will be observed. Because of the setting of the problem, the observed value f(x)f(x) could correspond to two possible inputs: xx and y=x⊕sy=x⊕s. Therefore the first register becomes:
   |ψ4⟩=12–√(|x⟩+|y⟩)|ψ4⟩=12(|x⟩+|y⟩)

where we omitted the second register since it has been measured.

5. Apply Hadamard on the first register:
   |ψ5⟩=12n+1−−−−−√∑z∈{0,1}n[(−1)x·z+(−1)y·z]|z⟩|ψ5⟩=12n+1∑z∈{0,1}n[(−1)x·z+(−1)y·z]|z⟩
6. Measuring the first register will give an output of:
   (−1)x·z=(−1)y·z(−1)x·z=(−1)y·z

which means:

x·z=y·zx·z=(x⊕s)·zx·z=x·z⊕s·zs·z=0 (mod 2)x·z=y·zx·z=(x⊕s)·zx·z=x·z⊕s·zs·z=0 (mod 2)

A string zz whose inner product with ss will be measured. Thus, repeating the algorithm ≈n≈n times, we will be able to obtain nn different values of zz and the following system of equation can be written:
⎧⎩⎨⎪⎪⎪⎪s·z1=0s·z2=0...s·zn=0{s·z1=0s·z2=0...s·zn=0

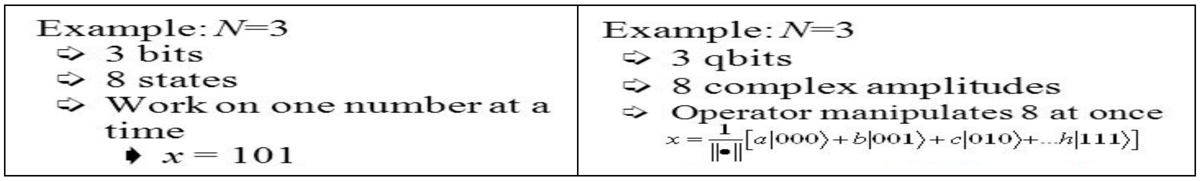
From which ss can be determined, for example by Gaussian elimination.

So, in this particular problem the quantum algorithm performs exponentially fewer steps than the classical one. Once again, it might be difficult to envision an application of this algorithm (although it inspired the most famous algorithm created by Shor) but it represents the first proof that there can be an exponential speed-up in solving a specific problem by using a quantum computer rather than a classical one.

III.COMPARISON OF CLASSICAL COMPUTING VERSUS QUANTUM COMPUTING

Table: 3.1 *Comparison Of Classical Computing Versus Quantum Computing*

Classical Computing	*Quantum Computing*
N Particles : 2^N unique states	N Particles : 2^N unique states
Computations are sequential • Select 1 state • Operate on it • Put it back into memory	Computations occur in parallel • States interact • They are entangled • Operate on all states at once



Example: N=3	Example: N=3
➩ 3 bits ➩ 8 states ➩ Work on one number at a time ➧ $x = 101$	➩ 3 qbits ➩ 8 complex amplitudes ➩ Operator manipulates 8 at once $x = \frac{1}{\|\bullet\|}[a|000\rangle + b|001\rangle + c|010\rangle + \dots h|111\rangle]$

IV.APPLICATIONS OF ALGORITHMS

Table: 4.1 *Applications of Quantum Algorithms*

Quantum Algorithms	*Applications*
Shor's algorithm	Fctoring-RSA Encryption,Quantum simulation,spin-off technology-spintronics,spin-off theory,DMRG theory,N represent ability theory.
Grover's alorithm	speedup over naïve algorithms, amplitude amplification, Determining connectivity of an n-vertex graph, to obtain speedups over classical algorithms, e.g.: Finding the minimum of n numbers in O(√ n) time.
Simons Algorithm	used to break a cryptographic primitive, assuming an appropriate query access model eg: Feistel networks.

V.EXPERIMENTAL IMPLEMENTATIONS OF QUANTUM ALGORITHMS

Table: 5.1 Test Results on Experimental Implementations of Quantum Algorithms

Algorithm	*Technology*	*Problem solved*
Shor's Algorithm	Integrated Optics	Factorisation of 21
Grover's Algorithm	NMR	Unstructured Search,N=8
Quantum annealing	D-Wave 2X	Ising model on a "Chimera" graph with 1097 vertices
HHL algorithm	Bulk Optics,NMR	2*2 system of linear equations

VI.CONCLUSION

Quantum computers are potential enough to solve the intraceable problems classically,through the quantum supremance memory components can be scaled to an atom or even smaller ,quantum registers will increases the computation speed in algorithms that will give advantage of quantum parellesim.

Shor's algorithm makes factoring large numbers tractable in quantum computing which is not possible for classical computing,database search algorithm shows a noteworthy research for a quantum computer to perform faster than classical computer.quantum simulators are making strides in fields varying from molecular energetics in physics.

REFERENCES

1.L. Adleman, J. Demarrias, M. A. Huang, "Quantum computability," SIAM Journal on Computing, vol. 5, pp. 1524–1540, 1997.
2. D. Aharonov, V. F. R. Jones, Z. Landau, "A Polynomial Quantum Algorithm for Approximating the Jones Polynomial," Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, 2006, arXiv:quant-ph/0511096.
3. D. Aharonov, T. Naveh, "Quantum NP – a survey," arXiv:quant-ph/0210077.
4. S. Bigelow, "Braid groups and Iwahori-Hecke algebras," arXiv:math.GT/0505064.
5. J. Birman, Braids, Links and Mapping Class Groups, Princeton University Press, 1974.
6. J. Birman, H. Wenzl, "Braids, link polynomials and a new algebra," Transactions of the American Mathematical Society, vol. 313, no. 1, 1989.
7. M. Bordewich, M. Freedman, L. Lov´asz, D. Welsh, "Approximate counting and quantum computation," Combinatorics, Probability and Computing, vol. 14, pp. 737–754, 2005.
8. C. M. Dawson, M. A. Nielsen, "The Solovay-Kitaev algorithm," Quantum Information and Computation, vol. 6, no. 1, pp. 80–95, 2006, arXiv:quant-ph/0505030.