ALGORITHMS FOR NP-COMPLETES

QUANTUM VS CLASSICAL

ALLISON NEWMAN

**MOTIVATION**

Quantum computing is a relatively new field of computer science. Richard Feynman introduced the idea of a quantum computer in 1981 [1]. Four years later, David Deutsch outlined his reimagining of the Church-Turing Hypothesis – the original theoritical model developed by Alan Turing in the 1930’s upon which modern computing systems are based – dubbed the Church-Turing Principle for quantum computers [2]. Since then, engineers have been invigorated to realize such a computer.

In computational complexity theory, we separate algorithms into two general categories, P (polynomial time) and NP (non-deterministic polynomial time). Understanding the difference between the two class requires some definitions:

* Deterministic algorithms: an algorithm in which a given input always procedes through the same path within the algorithm and a given input always returns the same output.
* Polynomial-time algorithm: an algorithm in which its runtime is a polynomial function on the size of the dataset, eg. O(n), O(n2).
* Exponential-time algorithm: an algorithm in which its runtime is an exponential function on the size of the dataset, eg. O(2n), O(n!).

Thus, an algorithm in class P is one that is both polynomial time and deterministic. An algorithm in class NP is one that is in polynomial time, but is not deterministic. The distinction is that in order to solve a problem that exists in NP determintsically, an exponential-time algorithm is required. Such algorithms that expand into exponential time may be deemed “intractible”, such that over a given dataset, may not be computable in any practical timeframe even on supercomputers or CPU clusters.

Quantum computers represent the possibility for some of these otherwise intractible problems to be solved exponentially faster [3]. Yet no one knows the full potential of this new computing paradigm. In 1994, Peter Shor changed the face of computing with his work on quantum algorithms for integer factorization [4]. Shor’s algorithm can factor an n-digit number in O(lg n)3) time on a quantum computer, exponentially faster than the best current classical algorithms [5].

The P = NP problem is currently one of the most important open questions in mathematics and theoretical computer science; it hypothesizes that for all algorithms in NP, there exists some algorithm in P to solve the same problem. Most computer science researchers reject the hypothesis that P = NP. This implies that a new computational paradigm will be needed to make significant advances in computational complexity theory and quatum computing is one such paradigm.

**SIGNIFICANCE**

The realization of quantum computing algorithms requires an addition to our complexity classes: now labelled “bounded-error polynomial time” (BQP) [3]. This class represents quantum algorithms that are solved in polynomial-time; all quantum algorithms are inherently probabilistic, thus denoting such algorithms in P would be disingenuous. BQP is, however, the goal when adapting classic algorithms for quantum computers. Shor’s algorithm achieved this goal by taking a previously NP-class algorithm and formulating a BQP-class algorithm for integer factorization on quantum computers [4].

WIthin computing, a special class of problem exists called NP-Complete. Such problems have algorithms which require exponential-time to solve, but given a solution can ve verified in polynomial time. A unique facet of NP-Completes is also a concept of “reducibility”. Fundamentally, this extends that all problems in NP-Complete are related, and generating a polynomial-time algorithm for one, would lead to polynomial-time algorithms for all other NP-Completes.

To examine the unfeasiblility of some NP-complete problems, let us examine the Travelling Salesman Problem (TSP). The TSP supposes that a travelling salesman starting at his home city, wishes to visit a number of other cities before returning home, traversing the smallest distance possible. The most simple form of the problem expects that all cities are connected, and that the cost-value for travelling from city A to city B is equal to the cost for travelling from city B to city A. Graphically, we represent TSP as a fully-connected, weighted graph with n-vertices representing cities, and n-1 edges representing the roads between each city.

A picture containing object

Description automatically generated

Figure . Where node 1 represents the home city, with edge weights (distances), and calculations for all possible trips.

Thus we can see in figure one an example of n=4 number of cities, yields 6, or (n – 1)!, possible trips. Also note, that half of trips are merely the reverse route of other trips (1 -> 2 -> 3 -> 4 -> 1 is mathematically the same trip as 1 -> 4 -> 3 -> 2 -> 1), therefore, we calculate a total possible number of trips as total trips. Given the rapid growth of factorials, the number of circuits to search grows extremely rapidly as cities are added to the graph.

In a practical example, if one were wanting to create a GPS system to that calculated the shortest route to naviagate to twenty cities from a given home city, the number of calculations required would be approximately . Assume a CPU has a clockspeed of 3.0 GHz, or calculations per second. Thus, to brute-force the Travelling Salesman at n = 21, this CPU requires over twelve years.

While algorithms that do not rely on a brute-force search have been designed to solve the Travelling Salesman problem and other problems in NP-Complete, these algorithms remain prohibitive within many practical applications that require 100% precision or extremely large datasets [6]. It is important then for quantum algorithms to be designed for these algorithms then to expand their usefulness for applications within business, industry, and other applied science research.

**RESEARCH QUESTIONS**

We will be examaning a quantum algorithm for solving an NP-Complete problem as compared to its classical algorithm counterpart. This examination includes:

* Runtime calculations – What is the current runtime of the best-known classical algorithm and quantum algorithm for solving an NP-Complete problem?
* What is presently possible? – What is the size of the dataset that is currently considered feasible to solve for using a classical algorithm? What is the size of the dataset that is currently feasible to solve for using a realized quantum computer? What is the effective cost of production for the computers running these benchmarks?
* Future projections: Using Moore’s Law, can we predict the computing power of both classical and quantum computers in the future? Can we predict the point at which realized quantum computers will surpass the capabilities of modern computers in solving an NP-Complete problem? What is the projected cost of production for these computers?
* Errors: Given that classical algorithms that solve large datasets for NP-Completes are non-deterministic, and quantum algorithms are inherently probabilistic, how do these errors affect the accuracy of the produced results?

In short, the purpose of this study is to determine the future feasibility of quantum computers in solving an NP-Complete problem such that there exists a benchmark for when we can expect quantum computers to surpass classical computers for sovling problems in this domain.

**HYPOTHESIS**

It has been calculated that in order to break a 1024-bit RSA key, a quantum computer requires 2,050 logical qubits, while currently the best quantum computers contain only 10’s of logical qubits [3]. While efforts to expand the number of qubits in quantum computers is rapidly underway, it is expected that quantum computers hosting thousands of qubits are more than a decade away. Given the overwhelming scale of the combinatorics within an NP-Complete problem, we can expect many thousands of qubits to be required to solve such problems, and can then quantify our expect timeframe for a feasible quantum computer in decades.

Srinivasan et al provide a starting point for calculating the number of qubits required to calculate a brute-force TSP, given n-number of cities as , where ε represents the desired error rate [7]. This calculation is used to find the number of qubits used in the “phase estimation” portion of their algorithm. In this case, we can see that the number of qubits used in the algorithm scales linearly with an increasing number of cities.

**METHODOLOGY**

The most daunting part of solving an NP-Complete problem is in the brute-force calculations on an n!-size set of elements. It follows then, that calculations on the runtime of the algorithm should focus more on this aspect, leaving our approach fairly problem-agnostic.

We will need to calculate the number of calculations performed by algorithm, and the number of qubits required to perform the algorithm on a quantum computer.

We will need to extrapolate current progress in quantum computer engineering to build a timeframe of when how many qubits will be available on computers in the future.

We will need to examine the error rates of classical algorithms, and the probability errors of quantum algorithms to measure the effects on the overall runtime and accuracy of such algorithms.

Computing a solution to the Travelling Salesman Problem (TSP) can be approached in two general ways: brute-force, or heuristically. The brute-force approach attacks the problem by checking every possible solution. Applied to TSP, this effectively means, generating a list of every possible tour on the map, calculating the distance of each tour, and then searching for the minimal tour distance. Brute-force always returns an optimal result but requires O(n!) time to do so.

A heuristic algorithm is one in which the solution is approximated, sacrificing the accuracy of the solution for significantly faster runtimes of the algorithm. As an example, one may solve for the TSP by greedily navigating from the current city to whatever the nearest unvisited city is – similar to a shortest path algorithm to find a minimal spanning tree. Such a solution may or may not return the optimal result when applied to TSP, but does so in O(n) time.

Methods for solving the TSP using a quantum algorithm have been shown by Srinivasan et al, and Bang et al [7] [8]. In these we find some phase states of the system representing each tour of the graph, operated on by some coefficient representing the cost of the tour, such that

.

**REFERENCES**

[1] R. P. Feynman, “Simulating physics with computers,” p. 22.

[2] D. Deutsch, “Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer,” *Proc. R. Soc. Lond. Ser. Math. Phys. Sci.*, vol. 400, no. 1818, pp. 97–117, 1985.

[3] E. National Academies of Sciences, *Quantum Computing: Progress and Prospects*. 2018.

[4] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, 1994, pp. 124–134.

[5] V. Singh, “Analysis of Quantum Algorithm to Find the Solution of Integer factoriza-tion Problem,” *Int. J. Adv. Res. Comput. Sci. Udaipur*, vol. 2, no. 4, Jul. 2011.

[6] U. Roy, S. P. Chawdhury, and S. Nayek, “Quantum Computer and quantum algorithm for travelling salesman problem,” *Natl. J. Syst. Inf. Technol. New Delhi*, vol. 2, no. 1, pp. 54–72, 2009.

[7] K. Srinivasan, S. Satyajit, B. K. Behera, and P. K. Panigrahi, “Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience,” *ArXiv180510928 Quant-Ph*, May 2018.

[8] J. Bang, S. Yoo, J. Lim, J. Ryu, C. Lee, and J. Lee, “Quantum heuristic algorithm for traveling salesman problem,” *J. Korean Phys. Soc.*, vol. 61, no. 12, pp. 1944–1949, Dec. 2012.