Quantum computing is a new field of computer science that seeks to introduce a new paradigm for computing faster than classical computers. Research into algorithms that are uniquely efficient on quantum computers has been ongoing since the introduction of Shor’s algorithm for integer factorization in 1994 [1]. In particular, the intractable algorithms in NP are desperate for a more efficient solution that could be provided by a quantum computer [2]– so much so that the P = NP problem is currently the most important question in theoretical computer science [2].

One such NP problem to be examined is the Travelling Salesman Problem. Precise solutions of the TSP can be calculated in no better than O(n22n) [3], becoming intractable at impractically small values of n; even the best approximation algorithms are limited to values of n in the thousands – requiring potentionally hundreds of CPU days to complete [4]. Thus is it critical to explore the capabilities of a quantum algorithm in solving the TSP. Such algorithms can currently be simulated using the Qiskit Aqua library, an open source project that provides applied science researchers the ability to utilize these quantum algorithms to advance their work. It is the intent of this project to contribute to the Qiskit library with the code and documentation of a TSP algorithm based on the algorithm proposed by Srinivasan et al [5].

**BACKGROUND**

**P vs NP**

Definition 1 – **P (Polynomial Time)** The set of problems whose solutions can be deterministically computed by an algorithm with polynomial time complexity.

Definition 2 – **NP (Non-deterministic Polynomial Time)** The set of problems whose solutions can be verified in polynomial, but whose solutions cannot be found deterministically by an algorithm of polynomial time complexity. Such problems require algorithms of exponential time to solve deterministically, or otherwise rely on a non-deterministic algorithm to approximate a solution.

A longstanding question in theoretical computer science, and one of seven Millenium Prize Problems [6], P vs NP asks for proof of set equivalence between P and NP. That is, can a determinstic polynomial time algorithm be found to solve all problems currently in NP? Most computer scientitists believe that P ≠ NP [7], prompting the need for new, faster ways to computing such problems.

**The Travelling Salesman**

The Travelling Saleman Problem (TSP) is one such NP problem. The TSP supposes a travelling salesman wishes to begin at his home city, visit some number of other cities, and then return home, visiting every city exactly one time and doing so in the shortest distance possible. The most simple form of this problem dictates that all cities are connected to one another, with some associated distance to travel between each, and that the distances are symmetric – travelling from city A to city B is the same distance as travelling from city B to city A. TSP is represented as a fully-connected, symmetrically weighted graph (V,E), where and , where n is the number of cities in the graph (including the starting city).

The brute-force search method of solving the TSP requires searching every possible route (tour) the salesman could take to find the shortest tour. In a graph of n-cities, there exists tours. This can be reduced to given symmetic weights, as half of the tours are merely the reverse of other tours. One can see that the limiting factor in such an algorithm is the rapid growth of the factorial function. Consider a TSP of n = 20, thus exists that need to be checked. If one were able to check the distance of 3.0 × 109 tours per second (a reasonable clock speed for a consumer-grade CPU), it would take over 38 years to find the shortest tour.

Requiring decades for such calculations is unacceptable for the many practical applications of the TSP. The most obvious of uses applications of TSP exist in routing problems. Every school district in the United States faces the problem of forming optimal bus routes to pick up students. A typical school bus with maximum occupancy of 72 students poses a TSP of potentially n = 72: find the shortest route to retrieve all students from their homes and then return to the school. UPS delivers an average of 15 million packages per day with each truck making an average of 120 deliveries. This then represents a TSP of n = 120. Even at these relatively small datasets, our a brute-force approach reaches runtimes that exceed the age of the universe. Consider that some applications such as routing the drilling sequence of a CNC router, mapping a telescopes trajectory to photograph stars in the universe, or searching through a genome sequence can involve potentially thousands of points of navigations [4], we see that advancements in the TSP algorithms must be made, or some problems would remain forever unsolvable.

In 1962, Held and Karp present a Dynamic Programming approach for solving TSP in O(n22n). Thusfar no algorithm exists to find an exact solution for TSP in faster than O(2n). Approximation methods for solving TSP have been devised that can find a near-solutions much faster, however. Most notably is the Concorde TSP Solver written by Applegate et al [4]. In 2006, Concorde found a solution at n = 85900, requiring 192 CPU days of computation time.

**Quantum Computing**

First theorized by Feynman in 1981 [8], the possibility of a quantum computer for years represented the possibility to use incredible power of quantum mechanics to store and manipulate information. Four years later, Deutsch outlined the Quantum Turing Machine, a reimagining of a classical Turing Machine allowing for the unique properties of quantum mechanics, such as the inclusion of probability amplitudes [9], [10].

The zeal for realizing a quantum computer, however, was not spurred until the introduction of Shor’s quantum algorithm for integer factorization [1]. This new algorithm not only had dire practical applications – threatening the security of RSA encryption – but did so in polynomial time, an exponential speedup over the best classical algorithms.

The rate of growth within classical computing hardware is also projected to slowdown. Moore’s Law – a prediction that the size of integrated circuit shrinks by half every 18 to 24 months, thus increasing computing power on a chip – has seen a decline in accuracy over the past decade [11]. It is expected that this trend of slower improvement in classical circuitry will continue into the future. Quantum computers then provide an alternative for faster computing, at a time when classical computing innovation is slowing to a crawl.

Quantum computing relies on a qubit’s (quantum bit) property of superposition, wherein it exists in some combination of the 0 and 1 states. When the qubit is measured, however, it is read as either a 0 or 1 discretely. The chance of a qubit being measured as 0 or 1 can be described by probability amplitudes – equal to the squared absolute value of the coefficient associated with each state [11]. More information on the notation describing these qubit systems is provided in Appendix A.

**METHODOLOGY**

1-2 paragraphs on Srinivasan Algorithm

1-2 paragraphs on Qiskit

**ANTICIPATED RESULTS**

1-2 paragraphs on Jupyter Notebooks

**APPENDIX A** – Notation and Vector Representation of the Qubit

The state of a quantum computer is often represented with Dirac notation. The following ket-vectors: represent the 0 and 1 states (analogous to bits of 0 and 1 in a classical computer) of a single qubit within a quantum computing system. These states can be re-represented as vectors of the forms and , respectively.

To expand the system for multiple qubits, one takes the direct product of all qubits within the system. For example, a two-qubit system, with both qubits in state 0 is expressed by the following:

and in vector format:

.

The associated bra-vector, or co-vector, for each state () is given as the transpose, row-vector, of the state with the complex-conjugate taken for each element. Therefore, we show the following: .

Dirac notation typically omits the operator symbols when operating on two states within an equation. For clarification, we look at the three combinations of two states:

* implies direct product, as shown above;
* implies scalar product, such that: ;
* implies cross product, such that: .

**Qubit State**

Because a qubit does not exist discretely in the states of or , we must find the probability that the qubit will be measured in either state. The probabilities are given by complex coefficients operating on either state, typically denoted α and β as scalars for the states and , respectively. Thus, we can describe the overall state of a single qubit given:

.

We find the probability amplitudes that a qubit is measured as a 0 or 1 given the probability amplitudes of these coefficients: [12].

**APPENDIX B –** Quantum Gates

**Hadamard Gate**

The Hadamard gate acts to superpose the state of a qubit, giving equal probability that the qubit is measured as either 0 or 1 [2].

Hadamard Gate: . When applied to a qubit of either the or basis vectors:

;

.

Likewise, we can refer to the superposed state produced as the Hadamard basis vectors and :

**Pauli-X Gate**

The Pauli-X gate is the equivalent of a not-gate in classical computers, and acts to invert the values of the and states on a single qubit. This gate is represented by the Pauli-X matrix: .

It is the case then that .

**Controlled Gates**

Controlled gates require two qubits wherein 1 of the qubits acts as the control, while the other is being acted upon. A Controlled gate is notated as , where is the gate to be controlled [13]. In its generalized form, a controlled gate is constructed as: [14].

The effect is that the controlled gate only acts on the 2nd qubit if the control qubit’s state is 1, and has no effect if the control qubit’s state is 0.

The Controlled-Not gate (CNOT or ) then takes the form:

.

This gate then maps the following two-qubit system as:

With the following as an example:

**Toffolli Gate**

The Toffolli Gate (a Controlled-Controlled-Not) Gate has the following form:

.

Similar to the Controlled Not gate, the Toffoli acts to switch the 3rd bit iff the first and second bits are 1.

REFERENCES

[1] 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.

[2] “The Golden Ticket,” *Princeton University Press*. [Online]. Available: https://press.princeton.edu/titles/9937.html. [Accessed: 15-Jul-2019].

[3] M. Held and R. M. Karp, “A Dynamic Programming Approach to Sequencing Problems,” *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, 1962.

[4] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Travelling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[5] 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.

[6] A. M. Jaffe, “The Millennium Grand Challenge in Mathematics,” vol. 53, no. 6, p. 9, 2006.

[7] L. A. Hemaspaandra, “SIGACT News Complexity Theory Column 36,” p. 15.

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

[9] 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.

[10] E. Bernstein and U. Vazirani, “Quantum Complexity Theory,” *SIAM J. Comput. Phila.*, vol. 26, no. 5, p. 63, Oct. 1997.

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

[12] “Quantum computing at Microsoft.” [Online]. Available: https://blogs.msdn.microsoft.com/uk\_faculty\_connection/2018/02/06/introduction-to-quantum-computing/. [Accessed: 25-Jun-2019].

[13] J. Preskill, “Caltech Physics 219/Computer Science 219 Quantum Computation,” 2013. [Online]. Available: http://www.theory.caltech.edu/%7Epreskill/ph219/index.html#lecture. [Accessed: 22-Jun-2019].

[14] E. G. Rieffel and W. H. Polak, *Quantum Computing: A Gentle Introduction*. MIT Press, 2011.