|  |
| --- |
| East Tennessee State University |
| Complexity Analysis of Quantum Algorithms for Travelling Salesman Problem |
| [Document subtitle] |

|  |
| --- |
| Allison Newman  7-22-2019 |

1. P vs NP
   1. P = NP
   2. Definition of P
      1. Deterministic
      2. Polynomial Time
   3. Definition of NP
      1. Verifiability in Polynomial Time
      2. Reduceability
      3. Non-deterministic algorithms
2. Moore’s Law
3. Big-O Notation
4. Complexity Theory
   1. Time Complexity Analysis
   2. Space Complexity Analysis
5. Travelling Salesman
   1. Definition
   2. Practical applications
   3. Brute-force algorithm
   4. Current Best-Known Heuristic Algorithm
6. History of Quantum Computing
   1. Shor’s Algorithm for Integer Factorization
   2. How Shor’s was analyzed for time & space complexity
   3. Future Projections on qubit availability
7. Introduction to Notation
   1. Dirac Notation for Quantum Computing
      1. Basis Set
      2. Cross Products
      3. Direct Products
      4. Direct Sums
   2. Quantum Gates and Circuits
      1. Pauli-X
      2. Hadamard
      3. Controlled
      4. Interaction of Hadamard Bases on C-gates
      5. Toffoli
   3. Quantum Fourier Transform
   4. Measurement
8. Srinivasan et al. Brute-Force Algorithm for TSP
9. Bang et al. Heuristic Algorithm for TSP

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.

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, while acknowledging that there is some level of probabilistic error given that 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 1. 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.

Given the intractable nature of the brute-force algorithm on even relatively small data sets, many algorithms have been created for the TSP using heuristic methods. Most notably in recent years is the Concorde algorithm [Applegate]. The algorithm solved a TSP at n=85900 in 2004 with a runtime of 192 CPU days. 85900 roughly represents every populated city in the United States. This, however, is an atypical dataset for TSP, with most practical applications remaining in the hundreds or few thousands of datapoints.

The simplest of practical applications involve routing problems – similar to the dilema of the travelling salesman for which the problem is named; however, many problems can be interpretted as a TSP problem. In fact, any problem that involves some distance or expense between a number of points can be structured as a TSP.

Consider a problem faced by every public school district in the United States: bus routes. The average occupancy of an American school bus is 72 students. That represents a TSP of potentially 72 nodes; leaving the school to pick-up up to 72 children at their homes before returning to the school.

UPS delivers an average of about 15 million packages per day, with each truck making an average of 120 deliveries per day. This then represents a TSP of 120 cities, not including the factory.

The above two problems are relatively small TSPs (both of which, however, are intractable by a brute-force algorithm), yet need to be repeated for thousands of busses of trucks. Optimizing the efficiency of the routes for these vehicles then represents a significant savings in both time and fuel, when considering the sheer amount of vehicles that could be affected.

There exist less obvious TSPs with greatly increased n-values. Consider an astronomer who wishes to observe the galaxies visible in the night sky with a telescope. If one were want to photograph some number of stars in succession, but wished to minimize the movement required by the telescope and the time wasted while the telescope is moving to its next target, one could then implement such a problem as a TSP.

CNC routers make thousands of cuts or drills per job. If one were wanting to drill thousands of holes in a circuit board in preperation for it to be assembled, this can be represented as a TSP. It is particularly cruicial to optimize these drills for efficiency when the scale is thousands of holes, done on thousands of boards in a commercial setting.

INTRODUCTION TO NOTATION

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: [1].

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 [2]. In its generalized form, a controlled gate is constructed as: [3].

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.

**PURPOSE**

Several algorithms for solving TSP on a quantum computer have been proposed, yet little or no information is provided regarding their practicality in the future. Therefore a timeline should be constructed as to predict when quantum computer capabilities will meet or surpass various practical benchmarks for solving the Travelling Salesman Problem.

**RESEARCH QUESTIONS**

1. With respect to the time complexity analysis, at what value of n does the quantum algorithm surpass the efficiency of the classical algorithm?
2. With respect to the space complexity analysis, how many logical qubits are required by a gate-model quantum computer to meet various benchmark n-values.
3. Given the predicted growth of quantum computer capabilities, provide a timeline for when quantum computing will be capble of computing the various benchmark n-values.
4. Assuming Moore’s Law remains consistent for the future, provide a timeline for the time analysis of classical algorithms in the future.
5. What is the approximate time for the intersection of the above two graphs (if an intersection exists); when does the capability of quantum computers exceed classical computers in solving the Travelling Salesman Problem?

Time Complexity Analysis

We feel that the asymptotic analysis of Big-O classications is not sufficient for precise analysis of the runtime of these algorithms. Therefore we will be calculating the exact number of steps to perform each algorithm at the benchmark n-values. The processor time of each algorithm at the benchmark n-values can be determined by the total steps divided by the clock speed.

So do we even need to talk about Big-O notation? Not really… I don’t feel like I have to talk in detail about big-o, but I can mention the big-o of each algorithm.

Space complexity Analysis

Do we care about the space complexity analysis of the classical algorithms? I think we have to.

While quantum computers are in their infancy, the primary restraint on their capability lies in the data storage capacity – the qubit. At current the best gate-model quantum computers in the world contain qubits numbered in the teens. In 2018, SOMEBODY ran Shor’s algorithm on a quantum computer. The total of the machines capacity, 16 qubits, was only capable of computing the factors of 35. As such, even the most advanced quantum computers currently available aren’t even capable of computing data outside the reach of even the human mind. Therefore, the space complexity of the proposed algorithms then is the most crucial consideration for these early quantum algorithms. We will be analyzing the growth rate of the qubit requirement relative to the benchmark n-values.

Logical Qubits:

We accept that major constraint on the growth of quantum computing is the development of quantum error correction. As such, it is seen that quantum computers will require some number of error correction qubits in addition to the logical qubits that are useable by the algorithms. As such, we restrict the space analysis of each algorithm to the logical qubits required by the algorithm. We also restrict analysis of available qubits in the future to available logical qubits.

Benchmark n-values:

We define the benchmark n-values to be rough representations of practical dataset sizes. The benchmark n-values are set at n=200, 1000, and 5000. n-values greater than 10,000 are considered to be very large datasets in the scope of TSP and do not represent practical applications.

Moore’s Law

Moore’s Law applies to classical computers. It predicts that every two years the volume of transistors is halved. This effectively doubles the number of transistors that can be installed on a chip, increasing the computing potential of a chip. While concerns have been raised about the efficacy of Moore’s Law in the future, for the purposes of our predictions, we assume that Moore’s Law remains true for all projections.

There isn’t however an equivalent to Moore’s Law