A quantum algorithm for the bottleneck travelling salesman problem

Raveel Tejani*
Department of Physics and Astronomy, University of British Columbia
6224 Agricultural Road, Vancouver, British Columbia, V6T 1Z1, Canada
(Dated: November 15, 2023)

the Bottleneck Travelling Salesman Problem is an NP-Hard optimization problem. The implication being there exists no solution in polynomial time. We propose an algorithm to provide a quadratic speedup using the grover's quantum search algorthim as the basis. We will test this algorithm on either real quantum hardware or simulators, implement improvements and discuss our findings.

PACS numbers:

I. MOTIVATION

Continuous advancements in quantum computing have opened up new ways to solve complex computational problems that were previously considered intractable using classical methods. The Bottleneck Traveling Salesman Problem (BTSP), is a good example of an optimization problem that stands as a challenge in the field of logistics and operations research (need ref). Its practical applications range from optimizing delivery routes to circuit design (need ref). As a result, achieving an efficient solution is highly sought after.

We embark on a journey at the intersection of quantum computing and optimization by introducing a quantum algorithm tailored to address the BTSP. It requires finding the shortest closed tour through a set of cities while minimizing the largest cost (the "bottleneck") along any route. Its computational complexity grows exponentially with the number of cities (need ref), rendering classical solutions impractical for large-scale instances.

Our approach is to capitalize on the inherent quantum parallelism, as well as the unique feature of phase encoding in quantum computing. By exploiting these phenomena, we aim to encode and manipulate the costs associated with various routes efficiently. Additionally, the utilization of Grover's search algorithm as a central component of our quantum approach significantly expedites the search for the optimal solution.

In this proposal, our primary objective is to shed light on the theoretical foundation of the quantum algorithm, its potential computational advantages over classical methods. Through this research, we aim to work towards a clear understanding of these aspects. The following sections will detail the BTSP, its expected computational complexity, Grover's algorithm and the significance of its implementation.

This section in the examples was typically a page, needs more substance. Maybe elaborating on the practical examples a little more. Dont love last paragraph

*Electronic address: raveel@student.ubc.ca

II. THEORY

A. Bottleneck Travelling Salesman Problem

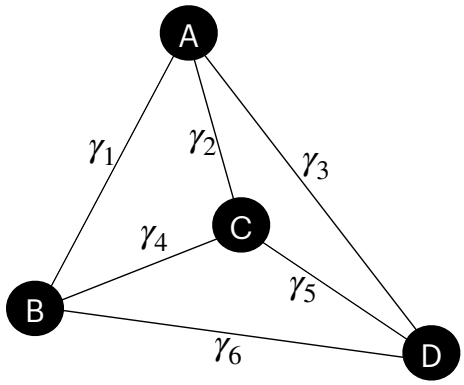

FIG. 1: An undirected weighted graph represention of a symmetric 4-city system. The vertices represent cities and the edge weights represent the cost of travel.

The BTSP can be represented as a graph problem. We start with a graph, whose vertices are labelled A through D, representing a 4-city system. We define movement from one vertex to another as a walk, done through the edges connecting our vertices. We are interested in a particular walk known as a hamiltonian cycle that contains every vertex extactly once before returning to the start. Our graph also includes edge weights, we define as γ_i. The BTSP is to find the hamiltonian cycle in a graph, where the largest edge weight ("bottleneck") is minimized. This is distinct from the Travelling Salesman Problem (TSP) where the combined edge weights in a given cycle is minimized. The total possible hamiltonian cycles is given by $(N-1)!$, where N is the number of nodes. We present a symmetric case in the figure, $N_k \to N_{k+1} = N_{k+1} \to N_k = \gamma_i$. Thus the total possible cycles is $(N-1)!/2$. It is important to note that a solution to either BTSP or TSP is not unique. BTSP solutions also do not nessesarily equate to the TSP solutions. We can illustrate an example with the help of our figure. Consider all the hamiltonian cycles for a symmet-

ric 4-city system:

the discussion of hamiltonian cycles within the paragraph could maybe be a little more formal? seperating the math from the paragraph?

$$A \to B \to C \to D \to A$$
$$A \to B \to D \to C \to A$$
$$A \to C \to B \to D \to A$$

Assigning some arbritrary weights, we can see the total costs of the cycles below. The first cycle is the solution to BTSP as its largest edge weight at 5 is the smallest among all three. The last cycle is a solution to the TSP as its combined edge weight is the smallest.

$$\gamma_1 + \gamma_4 + \gamma_5 + \gamma_3 = 4 + 4 + 5 + 4 = 17$$
$$\gamma_1 + \gamma_6 + \gamma_5 + \gamma_2 = 4 + 6 + 5 + 2 = 17$$
$$\gamma_2 + \gamma_4 + \gamma_6 + \gamma_3 = 2 + 4 + 6 + 4 = 16$$

By simply changing the weight of γ_6 to 5, we can illustrate all cycles are solutions to the BTSP.

$$\gamma_1 + \gamma_4 + \gamma_5 + \gamma_3 = 4 + 4 + 5 + 4 = 17$$
$$\gamma_1 + \gamma_6 + \gamma_5 + \gamma_2 = 4 + 5 + 5 + 2 = 16$$
$$\gamma_2 + \gamma_4 + \gamma_6 + \gamma_3 = 2 + 4 + 5 + 4 = 15$$

The computational complexity of the BTSP is shown to be NP-hard (reference.). Implying there is no algorithm for a solution in polynomial time. A brute-force approach would imply that we can run an algorithm in $O((N-1)!)$ time

B. Grover's Search Algorithm

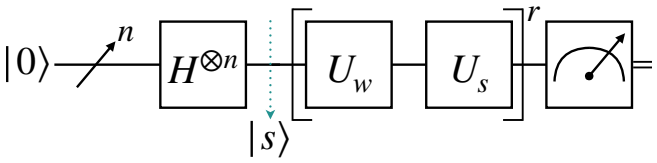

FIG. 2: A concise quantum circuit representation of Grover's search algorithm applied to an n-qubit system. The Hadamard $H^{\otimes n}$ gate performs a uniform superposition over all possible states N (2^n). The U_w gate marks our goal state. The U_s gate amplifies the amplitude of our goal state. the r exponent, refers to the number of iterations needed to maximize the amplitude. The approximate number is $\sqrt{N}$ times.

In 1996, Lov Grover proposed a quantum algorithm for unstructured database search, that would provide at most a quadractic speedup (need ref). This speed up can be realized for suffiently large databases, but does not provide the exponential speedup promised by other quantum algorithms. We can refer to figure 2, to see the quantum circuit representation. A geometric representation of the algorithm as well as the two-qubit example will help us understand.

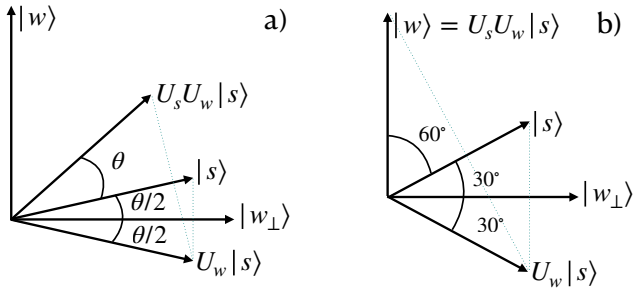

FIG. 3: geometric reprentation of Grover's algorithm. a) An iteration performed on an n-qubit system. Starting from the uniform superposition $|s\rangle$, we perform a reflection over the orthogonal states of $|w\rangle$ with U_w. Then we perform a reflection over the superposition state $|s\rangle$ with U_s. we can see our new state $|x\rangle = U_sU_w|s\rangle$ is closer to the goal state $|w\rangle$. A new iteration, would imply reflecting again over $|w_\perp\rangle$ and then reflecting back over $|x\rangle$. b) An iteration performed on a 2-qubit system. Only one iteration is required to hit any goal state $|w\rangle$.

Lets walk through one iteration of Grover's algorithm. Most quantum circuits are initialized at $|0\rangle$, then we apply the H gate, this results in a uniform superposition:

$$|s\rangle = H^{\otimes n}|0^{\otimes n}\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \tag{1}$$

The orthogonal states to our goal state can be represented as the sum of all states not containing $|w\rangle$

$$|w_\perp\rangle = \frac{1}{\sqrt{2^n - 1}} \sum_{x \in \{0,1\}^n}^{x \neq w} |x\rangle$$

From here we apply the U_w gate, for our reflection over the orthogonal states. This can be thought of flipping the sign onstate $|w\rangle$ contained within $|s\rangle$. We can see the visual representation in figure 3. We then perform a similiar reflection over the original state $|s\rangle$:

$$U_w|s\rangle = (\mathbb{I} - 2|w\rangle\langle w|)|s\rangle = |s'\rangle$$

$$U_s|s'\rangle = (2|s\rangle\langle s| - \mathbb{I})|s'\rangle = |s''\rangle$$

Our $|s''\rangle$ is now closer to our goal state $|w\rangle$. If we want to maximize our probability of measuring $|w\rangle$, we will need to perform a number of iterations (r). For sufficiently large number of states, $N = 2^n$, we can make a few approximations to calculate r

$$\langle w|s\rangle = \frac{1}{\sqrt{N}} = \sin\frac{\theta}{2} \approx \frac{\theta}{2} \tag{2}$$

$$P(w) = |\langle w|s\rangle|^2 = \frac{1}{N} = \sin^2\frac{\theta}{2}$$

$$P(w) = |\langle w|(U_s U_w)^r|s\rangle|^2 = \frac{1}{N} = \sin^2(\theta(\frac{1}{2}+r))$$

to maximize the probability we need P(w) = 1, so we can set the inside of the sine function to $\pi/2$:

$$\theta\left(\frac{1}{2}+r\right) = \frac{\pi}{2}$$

$$r = \frac{\pi}{2\theta} + \frac{1}{2} \approx \frac{\pi}{2\theta} \tag{3}$$

Using eqn. 2 to substitute θ in eqn. 3, we can get an approximate result for r

$$r \approx \frac{\pi}{4}\sqrt{N} \tag{4}$$

C. Quantum Phase Estimation Algorithm

Let's first briefly discuss the Quantum Fourier Transform (QFT). Given a computational basis state $|x\rangle$, applying the QFT (F_N) results in:

$$F_N|x\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi i x k N^{-1}}|k\rangle$$

let's represent this in binary notation and decompose it into a tensor product. We can represent $|x\rangle$ as a string of bits, and the QFT as a tensor product of single qubit basis states:

$$|x\rangle = |x_1 x_2 ... x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes ... \otimes |x_n\rangle$$

$$F_N|x\rangle = \frac{1}{\sqrt{2^n}}\bigotimes_{j=1}^{n}(|0\rangle + e^{2\pi i x k 2^{-j}}|1\rangle) \tag{5}$$

$$= \frac{1}{\sqrt{2^n}}((|0\rangle + \omega_1|1\rangle) \otimes (|0\rangle + \omega_2|1\rangle) \otimes ... \otimes (|0\rangle + \omega_n|1\rangle))$$

$$\begin{aligned}
\omega_1 &= e^{2\pi i x 2^{-1}} = e^{2\pi i(0.x_n)} \\
\omega_2 &= e^{2\pi i x 2^{-2}} = e^{2\pi i(0.x_{n-1}x_n)} \\
&... \\
\omega_n &= e^{2\pi i x 2^{-n}} = e^{2\pi i(0.x_1 ... x_n)}
\end{aligned}$$

An important characteristic of the w_j is the bit shift occuring in the exponent. if we look at w_1, the exponent has a factor $x2^{-1}$, which is equivalent to one right bit shift: $x_1 ... x_{n-1}.x_n$. Integer multiples of the exponent would imply full rotations returning to the same point thus we can ignore all the values on the left hand side of the decimal and we are left with $0.x_n$.

Let's discuss the phase estimation problem. Given an eigenstate $|\lambda\rangle$ of a unitary operator U, we want to calculate a good approximation for $\phi \in [0,1)$ satisfying:

$$U|\lambda\rangle = e^{2\pi i\phi}|\lambda\rangle \tag{6}$$

the phase estimation algorithm uses two registers of qubits. The first one will be our control qubits that determine the precision of our approximation. The second register will be a set of m qubits initialized to an eigenstate $|\lambda\rangle$.

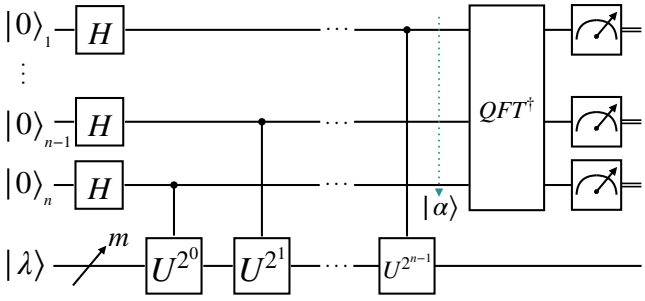

FIG. 4: a quantum circuit representation of the phase estimation algorithm. $U|\lambda\rangle = e^{2\pi i\phi}|\lambda\rangle$, we can approximate a value for $\phi \in [0,1)$. The Using the Hadamard gate, we observe a uniform superposition over n qubits. By performing the apply a controlled$-U$ operation qubit by qubit for n-qubit system, w

Let's walk through through the quantum circuit in figure 4, to understand the inner workings of this algorithm. Our initialized state is $|0^{\otimes n}\lambda\rangle$. From here we perform a the same operation we find in equation 1, where all the qubits in the first register are set to a uniform superposition on all states 2^n. The next portion of the algorithm involves applying controlled gates based on the unitary operator U. The function of these CU gates is to apply the operator U on $|\lambda\rangle$ if the control qubit is in the state $|1\rangle$. We can have a look at the affect on the n$^{\text{th}}$ qubit, after it has been prepared in a superposition by the Hadamard gate:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\lambda\rangle = |0\lambda\rangle + |1\lambda\rangle$$

Applying CU and factoring out the eigenstate:

$$\begin{aligned}
CU\frac{1}{\sqrt{2}}(|0\lambda\rangle + |1\lambda\rangle) &= \frac{1}{\sqrt{2}}(CU|0\lambda\rangle + CU|1\lambda\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\lambda\rangle + e^{2\pi i\phi}|1\lambda\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\phi}|1\rangle) \otimes |\lambda\rangle
\end{aligned}$$

We can see the eigenstate after the CU operation is left unchanged. the phase has been encoded into the control qubit instead, this result is due to the phase kickback (reference). Thus, we can reuse our eigenstate for the next qubit. Consecutive qubits have double the amount of CU operators as the previous, thus squaring the eigenvalue each time:

$$\begin{aligned} \text{qubit}_{\text{n-1}} &: |0\rangle + e^{2\pi i\phi 2}|1\rangle \\ &... \\ \text{qubit}_1 &: |0\rangle + e^{2\pi i\phi 2^n}|1\rangle \end{aligned}$$

We know that the value of $\phi < 1$. We can represent this in binary notation in the form $0.\phi_1\phi_2...\phi_n$:

$$\phi = \sum_{j=1}^{n} \phi_j 2^{-j}$$

If we have another look at the control qubits using binary notation for ϕ instead, we can see the result of the multiple CU operations simply results in right bit shifts:

$$\begin{aligned} \text{qubit}_{\text{n}} &: |0\rangle + e^{2\pi i(0.\phi_1\phi_2...\phi_n)}|1\rangle \\ \text{qubit}_{\text{n-1}} &: |0\rangle + e^{2\pi i(0.\phi_2...\phi_n)}|1\rangle \\ &... \\ \text{qubit}_1 &: |0\rangle + e^{2\pi i(0.\phi_n)}|1\rangle \end{aligned}$$

If we look at the form of first register after all the CU operations in the state $|\alpha\rangle$, it will resemble the result of performing the QFT we saw in equation 5. Where our ω_j are:

$$\begin{aligned} \omega_1 &= e^{2\pi i x 2^{-1}} = e^{2\pi i(0.\phi_n)} \\ \omega_2 &= e^{2\pi i x 2^{-2}} = e^{2\pi i(0.\phi_{n-1}\phi_n)} \\ &... \\ \omega_n &= e^{2\pi i x 2^{-n}} = e^{2\pi i(0.\phi_1...\phi_n)} \end{aligned}$$

simply performing the inverse QFT will give us $|\phi\rangle = |\phi_1\phi_2...\phi_n\rangle$. We can immediately see the approximation is limited by the number of qubits in the first register. A simple strategy would be to increase the number of qubits, however this would also increase computational cost as we double our use of CU gates for each additional qubit.

III. DETAILS OF PROPOSED EXPERIMENT/CALCULATION

Using the tools we explore in the theory section, i propose to create a quantum algorithm that will calculate a solution for the BTSP in $\mathcal{O}(\sqrt{(N-1)!})$. Solving the BTSP can be broken down into two steps:

1. Finding all hamiltonian cycles.

2. Comparing the max edge weight contributions and finding a minimum.

There is a strategy of finding the Hamiltonian cycles through the phase estimation algorithm. Outlined in "paper 2018", they store edge weights as phases and create an N^N size diagonal matrix in which $(N-1)!$ entries correspond to the hamiltonian cycles. Their approach is great, however it is not easy to see each edge weight contribution in the cycle. For the BTSP we need to see the max edge weight contribution, thus I hope to leverage their approach and construct a strategy to find it in each Hamiltonian cycle.

Once all the max edge weights are available, the next challenge is using Grover's algorithm to find the minimum. We need to construct a U_w. Given a function $f : \{0, 1, ..., N-1\} \to \{0, 1\}$:

$$U_w|x\rangle = (-1)^{f(x)}|x\rangle = \begin{cases} -|x\rangle & \text{for } x = w \\ |x\rangle & \text{for } x \neq w \end{cases} \tag{7}$$

Here we need to map our minimum edge weights to the inputs of our function f and devise a strategy where the minimum edge weight input would output a value of 1. If both of these steps are accomplished, we will have ourselves a quantum algorithm for the BTSP. We can then test this algorithm on open source Quantum Simulators or even possibly use IBM Quantum Experience (10 min/month free access). We can test the efficiency, errors and drawbacks of our algorithm in the process. We can then implement improvements and test again.

IV. PLANNED SCHEDULE

Timeline	**Plan**
October - January	Research and Algorithm Construction
January - February	Algorithm Testing and Improvements
March	Thesis Writing

V. ACKNOWLEDGEMENTS

I want to express my gratitude to Dr. Matthew Choptuik for not only supervising me but also for his encouragement and support to pursue a topic that truly captivates my interest.

VI. APPENDIX

some basics on quantum computing to go here, plus details on the common used dirac notation, gates, etc:

single qubit states:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

2-qubit states:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

$$|00\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |01\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Multiple qubit states are achieved through tensor products of single states. an n- qubit system will span a 2^n Hilbert space. We can see with the 2-qubit system we have 4 states

Hadamard Gate: commonly used to create a uniform superpositon:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

We can refer to equation 1 to see the result of Hadamard on an n-qubit system. Let's have a look at the Hadamard gate applied to $|00\rangle$:

$$\begin{aligned} H^{\otimes 2}|00\rangle &= H|0\rangle H|0\rangle \\ &= |+\rangle|+\rangle \\ &= (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \\ &= \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned}$$

Here we can see why Dirac notation is powerful as it simplifies tensor products and removes any need for matrix representation.

No references listed yet