Revolutionizing Quantum Computing: Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE)

Solving The Traveling Salesman Problem With Brute Force and VQE Algorithms

A thesis submitted in partial fulfillment of the requirements of the award of the degree of

M.Sc

in

Physics

By

ANKIT KASHYAP (213222006)

DEPARTMENT OF PHYSICS

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI – 620015

JUNE 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **Revolutionizing Quantum Computing: Quantum Fourier Transform(QFT) and Quantum Phase estimation(QPE), Solving The Traveling Salesman Problem With Brute Force and VQE Algorithms** is a bonafide record of the work done by

ANKIT KASHYAP (213222006)

in partial fulfillment of the requirements for the award of the degree of **Master in Science** in **Physics** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2022-24.

Dr. Joy Prakash Das
Project Supervisor

Dr. Justin Josephus
Head of the Department

Project Viva-voice held on _____

Internal Examiner

External Examiner

ABSTRACT

This thesis explores the transformative potential of quantum computing through Quantum Phase Estimation (QPE) and Quantum Fourier Transform (QFT). Quantum computing promises exponential speedups over classical methods by exploiting superposition and entanglement. The study delves into the theoretical foundations of QPE and QFT, highlighting their significance in quantum algorithms and their impact on computational tasks. Practical challenges and advancements in implementing these algorithms on quantum hardware are examined, emphasizing their role in revolutionizing computation across various fields.

Quantum computers will enable calculations beyond the reach of classical computers. The Quantum Fourier Transform (QFT) is a fundamental operation in quantum computing, essential for estimating unknown phase shifts. This project paves the way for a future where quantum computation transforms various scientific and technological frontiers. QFT is crucial in quantum algorithms like Shor's for factoring large numbers and Grover's for searching databases, leveraging principles of quantum mechanics to outperform classical computers in specific tasks.

Using Qiskit, an open-source quantum computing framework, this thesis simulates and analyzes QFT, QPE, and other key quantum algorithms. QFT and QPE are pillars in quantum computation, enabling efficient signal processing and precise estimation of eigenvalues. By leveraging Qiskit's robust simulation capabilities, this research investigates the performance, scalability, and optimization of these algorithms across various quantum hardware configurations. Through detailed simulations and experimental validations, the study sheds light on the behavior and potential of QFT, QPE, and other algorithms in practical quantum computing scenarios. The insights gained pave the way for advancing quantum algorithm design, optimization strategies, and their realization on quantum hardware platforms.

Additionally, the thesis explores two contrasting approaches to solving the Traveling Salesman Problem (TSP): the traditional brute force algorithm and the quantum-inspired Variational Quantum Eigensolver (VQE) algorithm. The TSP involves finding the shortest route that visits a set of cities exactly once and returns to the starting point. The brute force algorithm evaluates all possible city sequences to find the optimal route, which is highly accurate but computationally expensive, especially for large problems. On the other hand, the VQE algorithm leverages quantum computing principles to explore the solution space

more efficiently. Comparative analysis evaluates the computational complexity, scalability, and solution quality of both approaches. Findings suggest that VQE offers significant speedup for large-scale TSP instances compared to brute force. This study enhances the understanding of performance trade-offs between classical and quantum optimization algorithms and underscores the importance of exploring innovative approaches to complex optimization problems.

In summary, this thesis examines the significance of QPE, QFT, and their implementation in solving the TSP and other challenges in quantum information processing. Through theoretical exploration, simulation, and practical analysis, the study highlights the transformative potential of quantum algorithms in advancing computation and solving complex problems more efficiently than classical methods.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the people for guiding me throughout this course without those people this project and the results achieved would not be an easy task. I would like to express my sincere thanks to,

Dr. Joy Prakash Das, Assistant Professor, Department of Physics, for helping me and guiding me in the course of this project. Without his guidance, I would not have been able to complete this project successfully. His patience and genuine attitude are and always will be a source of inspiration to me.

Dr. Justin Josephus, The Head of the Department, Department of Physics, for allowing me to avail the facilities at the department.

I am also thankful to the faculty and staff members of the Department of Physics.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The story begins in the early 1980s with the works primarily of Paul Benioff, who first thought about a quantum mechanical model of a computer, and of Richard Feynman, who pointed out the potentiality of a quantum device in simulating efficiently a quantum system. This culminated in 1985 with the work of David Deutsch [3] that described the first theoretical model of a Universal Quantum Computer, the quantum analog of the classical Turing Machine. In the subsequent years, the major theoretical breakthroughs were represented by the works of Peter Shor (1994) and Lov Grover (1996). Shor demonstrated that the problem of finding the prime factors of an integer and the problem of the discrete logarithm could be performed on a quantum device in polynomial time, while classically they would require exponential time. Grover instead found an algorithm that could outperform classical computers in the task of searching through an unstructured database. These algorithms, in virtue of their widespread applicability, brought considerable attention to this field.

In parallel to the development of the theory of quantum computing, it began the engineering challenge to physically realize such a quantum computer. To this day there are many different candidates, distinguished by the physical systems used to implement the qubits. To mention a few, there are Superconductor Quantum Computing (on which companies such as Google or IBM conduct today's research), Trapped Ion Quantum Computers, and Neutral Atoms in Optical Lattices.

In the realm of modern computing, traditional silicon-based systems are encountering limitations in tackling the exponentially growing complexities of computational tasks. As the demands for processing power continue to escalate, researchers and technologists are turning towards the revolutionary paradigm of quantum computing to unlock unprecedented computational capabilities. At the heart of this paradigm shift lie fundamental quantum algorithms such as Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE), which promise to revolutionize computation by harnessing the principles of quantum mechanics.

Central to many quantum algorithms, the Quantum Fourier Transform (QFT) is a quantum analog of the classical discrete Fourier transform, providing a

powerful tool for signal processing and data analysis in quantum systems. By efficiently transforming between the time and frequency domains, QFT enables quantum computers to process and manipulate complex data structures with unprecedented efficiency, offering transformative applications in fields ranging from cryptography to machine learning.

Similarly, Quantum Phase Estimation (QPE) stands as a cornerstone in the arsenal of quantum algorithms, facilitating the precise estimation of phase parameters for unitary operators—a task that lies at the heart of many quantum algorithms, including Shor's algorithm for integer factorization and quantum simulation. QPE's ability to accurately determine phase information enables quantum computers to perform complex computations with unparalleled precision, unlocking new frontiers in computational capability.

Despite the immense promise of quantum algorithms like QFT and QPE, their practical realization poses significant challenges. Quantum hardware remains in its nascent stages of development, with current systems subject to noise, error, and limited qubit coherence times. Furthermore, the design and optimization of quantum algorithms require careful consideration of quantum circuit architectures, gate operations, and error-correction strategies to mitigate these challenges and unleash the full potential of quantum computation.

What is quantum mechanics? Quantum mechanics is a mathematical framework or set of rules for the construction of physical theories. The rules of quantum mechanics are simple but even experts find them counterintuitive, and the earliest antecedents of quantum computation and quantum information may be found in the long-standing desire of physicists to better understand quantum mechanics. One of the goals of quantum computation and quantum information is to develop tools which sharpen our intuition about quantum mechanics, and make its predictions more transparent to human minds.

The most spectacular discovery in quantum computing to date is that quantum computers can efficiently perform some tasks that are not feasible on a classical computer. For example, finding the prime factorization of an n-bit integer is thought to require,

$$exp(\Theta(n^{1/3}log^{2/3}(n)))$$

(where Θ indicates that the growth rate of the function is tightly bound by the given expression), operations using the best classical algorithm known till now is known as the Number field sieve. This is exponential in the size of the number being factored, so factoring is generally considered to be an intractable problem on a classical computer. It becomes impossible to factor in even modest

numbers.

In contrast, a quantum algorithm can accomplish the same task using,

$$O(n^2 log(n) log[log(n)])$$

operations. That is, a quantum computer can factor a number exponentially faster than the best-known classical algorithms.

A quantum computer is a finite-dimensional quantum system composed of qubits, performing various unitary operations on qubits (quantum gates) as well as quantum measurements. Alternatively, d-dimensional qubits can be used instead of two-dimensional qubits. Qutrit is a special name for the case d=3, while quaquart corresponds to d =4.

Working with qubits instead of two qubits may offer some advantages. The required number of qubits is smaller by a factor $log_2(d)$ compared to the number of qubits for the same dimension of a composite system of n qubits is 2^n, while the same dimension can be reached with only,

$$\log_d 2^n = \frac{log_2 2^n}{log_2 d} = \frac{n}{log_2 d}$$

qubits. Such a reduction of quantum information physical carriers is advantageous, considering the difficulty of reliably controlling a large number of them. When fewer quantum information carriers are used, a decrease in the overall decoherence is expected, helping to alleviate scalability issues.

Another advantage, which is also related to the adverse effect of decoherence, is that fewer multilevel qudit gates are required to construct a quantum circuit implementing a given unitary operation compared to the case of using two-level gates. Fewer gates reduce the number of steps needed to complete the circuit operation (depth) and, consequently, fewer errors are accumulated during the overall operation of the circuit.

Therefore Qubit Technology, with a qubit being a quantum version of d-ary digits for d>2; is emerging as an alternative to qubit for quantum computation and quantum information science. Due to its multilevel nature, qubit provides a larger state space to store and process information and the ability to do multiple control operations simultaneously. These features play an important role in the reduction of the circuit complexity, the simplification of the experiment setup, and the enhancement of the algorithm efficiency.

1.1 Superposition and Entanglement

In quantum mechanics, superposition and entanglement are two fundamental phenomena that differentiate quantum systems from classical systems:

1.1.1 Superposition

- Superposition refers to the property of quantum systems where they can exist in multiple states simultaneously until measured.
- In classical physics, an object can only be in one state at a time, but in quantum mechanics, due to the wave-particle duality of quantum particles, a particle can be in a superposition of multiple states, each with a certain probability amplitude.
- Mathematically, if a quantum system can be described by a wavefunction, then it can exist in a linear combination of its possible states, with each state weighted by a complex probability amplitude.

1.1.2 Entanglement

- Entanglement is a phenomenon where the quantum states of two or more particles become correlated in such a way that the state of one particle is dependent on the state of another, regardless of the distance between them.
- When particles become entangled, measuring the state of one particle instantaneously determines the state of the other particle, regardless of the spatial separation between them.
- Entanglement plays a crucial role in various quantum phenomena and technologies, including quantum teleportation, quantum cryptography, and quantum computing, where it forms the basis for quantum gates and quantum information processing.

1.2 From Cbits to Qbits

We need a term for a physical system that can exist in two unambiguously distinguishable states. The classical bit is the basic unit of information and the simplest system conceivable in classical physics. A classical bit consists of two possible states, denoted by 0 and 1, and can be represented by the position of a ball (upper or lower shelf) or the orientation of a vector (up or down)(Fig.1). The qubit generalizes the classical bit.

In contrast to the classical bit, which can only be either in state 0 or in state 1, a qubit can admit many more possible states. The quantum states corresponding to the classical states 0 and 1 are denoted as $|0>$ and $|1>$ (using the so-called bra-ket notation). In some sense, the quantum system can be simultaneously in both states, forming a superposition. Any superposition of the two basis states is allowed. Each superposition corresponds to a particular arrow of length, one pointing in a specific direction in space. All possible superpositions form a sphere (the so-called Bloch sphere, see Fig. 1), where the north and south pole constitute the states $|0>$ and $|1>$.

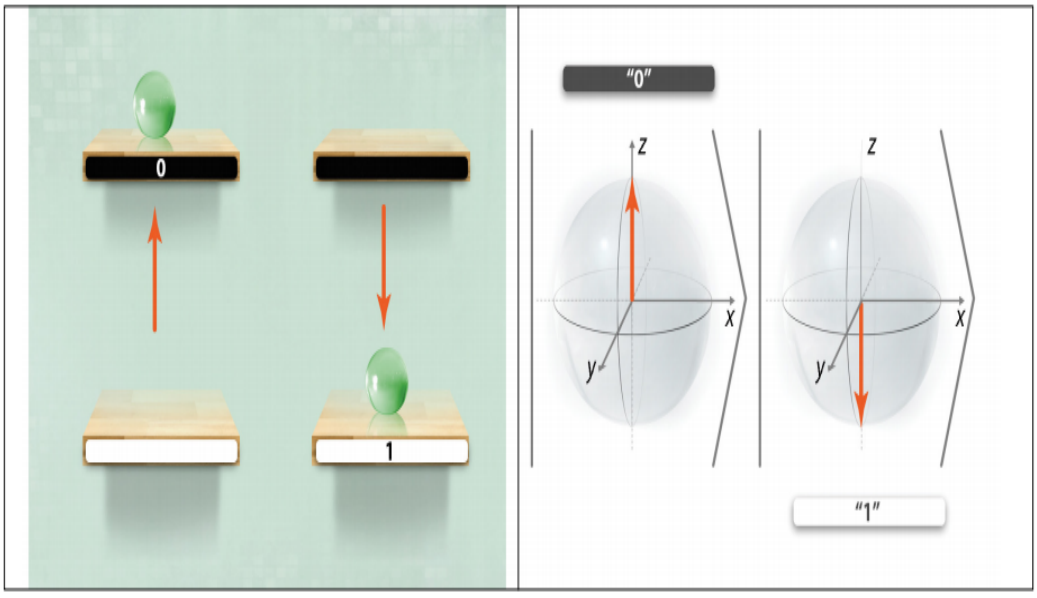


Figure 1.1: Left: Illustration of a classical bit. Right: Illustration of the two basis states $|0>$ and $|1>$ on the Bloch sphere.

Here we present an approach to quantum physics based on the simplest quantum mechanical system—the quantum bit (qubit). Like its classical counterpart—the bit—a qubit corresponds to a two-level system, i.e., some system with a physical property that can admit two possible values. While typically a physical system has more than just one property or the property can admit more than just two values, in many situations most degrees of freedom can be considered to be fixed or frozen. Hence a variety of systems can be effectively described as a qubit. For instance, one may consider the spin of an electron or atom, with spin up and spin down as two possible values, and where other properties of the particle such as its mass or its position are fixed. Further examples include the polarization degree of freedom of a photon (horizontal and vertical polarization), two electronic degrees of freedom (i.e., two energy levels) of an atom, or the position of an atom in a double well potential (atom in left or right well). In all

cases, only two states are relevant to describe the system.

Quantum systems show some unique features not inherent in classical systems, including the possibility of superposition and an intrinsically probabilistic behavior and state change under measurements, as shown in Figure 1.2.

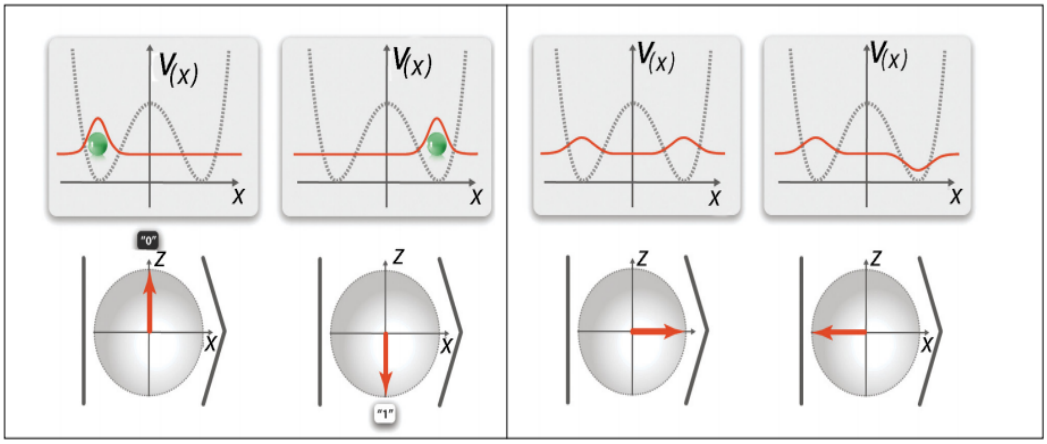


Figure 1.2: Left: Realization of a qubit using the spatial degrees of freedom of a single atom. The red curves indicate the wave function of the atom; however, in the present context, the description can be reduced to a two-level system. The state $|0>$ corresponds to the localization of the atom in the left well of the double well of well potential, while the state $|1>$ corresponds to the localization in the right well. Right: Superposition states of an atom in a double well potential. The state $|0_x>$ corresponds to a symmetric superposition (symmetric wave function), while the state $|1_x>$ corresponds to the anti-symmetric superposition (anti-symmetric wave function).

A superposition state of an atom corresponds to the atom being at two positions simultaneously, which is counterintuitive and difficult to align with our everyday experience. Classical particles can only be at one position, and not at two positions simultaneously. Despite this, it is possible to prepare such superposition states experimentally, where a single atom is trapped in a double well potential. The manipulation of the potential barrier can be used to control a coherent tunneling process and thus manipulate the quantum state of the atom.

Mathematically, the state of a qubit is described by a two-dimensional (complex) vector. We denote the possible quantum states by the basis vectors,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad and \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

respectively and an arbitrary state as a superposition of the basis vectors. Quantum states correspond to vectors of length 1, and can be written as,

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)\exp(i\phi)|1\rangle = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\exp(i\phi) \end{pmatrix}$$

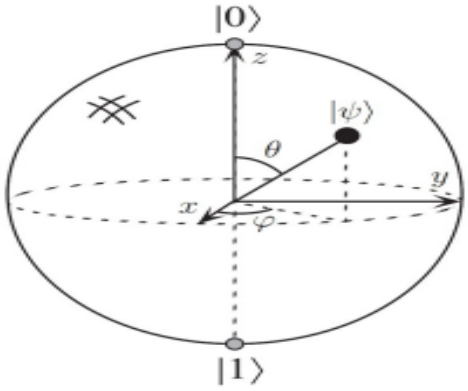


Figure 1.3: Bloch sphere for a single qubit

CLASSICAL versus QUANTUM BITS	Cbits	Qbits
States of n Bits	$\|x\rangle_n$, $0 \leq x < 2^n$	$\Sigma \alpha_x \|x\rangle_n$, $\Sigma \|\alpha_x\|^2 = 1$
Subsets of n Bits	Always have states	Generally have no states
Reversible operations on states	Permutations	Unitary transformations
Can state be learned from Bits?	Yes	No
To get information from Bits	Just look	Measure
Information acquired	x	x with probability $\|\alpha_x\|^2$
State after information acquired	Same: still $\|x\rangle$	Different: now $\|x\rangle$

Table 1.1: Above table shows the differences in classical and quantum bits.

1.3 Significance of quantum algorithms in comparison to classical algorithms

An algorithm is a sequence of logically connected mathematical steps that solve a problem. For example, an algorithm to add three numbers can have two steps: add the first two numbers in the first step and the result to the third number in the second step.

In the early 1980s, Manin (1980) and Feynman (1982) independently observed that computers built from quantum mechanical components would be ideally suited to simulating quantum mechanics. Whereas brute-force classical simulation of a system of n quantum particles (say, two-level atoms) requires storing 2^n complex amplitudes and hence exponentially many bits of information, a quantum computer can naturally represent those amplitudes using only n quantum bits. Thus, it is natural to expect a quantum mechanical computer to outperform a classical one in quantum simulation.

Quantum computers achieve speedup over classical computation by taking advantage of interference between quantum amplitudes. Of course, interference occurs in classical wave mechanics as well, but quantum mechanics is distinguished by the ability to efficiently represent a large number of amplitudes with only a few quantum bits. In Shor's algorithm and its predecessors, the "exponential interference" leading to quantum speedup is orchestrated using a unitary operation called the **Quantum Fourier Transform (QFT)**, an algebraic operation.

A Quantum Fourier Transform (QFT) can calculate the Fourier transform of a vector with time complexity $\mathrm{O}(\log_2(N))^2$, compared to the complexity $\mathrm{O}(N \log_2(N))$ of a classical FFT. However, if one needs to read out the full vector instead, the complexity becomes $\mathrm{O}(N poly \log(N))$ again, without an advantage over the classical algorithm. Thus the QFT is a useful algorithm if the input data is prepared algorithmically, and limited sampling of the result vector is sufficient.

Quantum Phase Estimation algorithm, which is precisely one of the topics of this thesis, is a class of quantum algorithms that have as objective the estimation with arbitrary precision of the eigenvalue of a unitary operator, which is a complex phase. As pointless as this may sound, phase estimation is mainly used as a subroutine in many other quantum algorithms of interest, such as Shor's factoring algorithm.

Therefore, A quantum algorithm is also a series of steps, but its implementation requires quantum gates. Some problems may need fewer steps on the part of a quantum algorithm than the number of steps required by a classical algorithm. That is, the quantum algorithm can speed up the computation.

1.4 Traveling salesman problem (TSP)

The Traveling Salesman Problem (TSP) involves finding the shortest route for a salesman to visit a set number of cities, each only once, and return to the starting point. It assumes equal travel cost between any two cities, regardless of direction. While originally named for practical travel planning, TSP is now primarily used to study general optimization methods applicable to a wide range of problems.

The importance of the TSP lies in its representation of combinatorial optimization problems. It is an NP-complete problem, meaning if an efficient (polynomial-time) algorithm is found for the TSP, it could lead to efficient algorithms for all NP-complete problems.

To solve large TSPs, we need a good mathematical model. In TSP, each city is a node in a graph, and edges connect the nodes with distances (costs). When the salesman can get from every city to every other city directly, then the graph is said to be complete. A round-trip of the cities corresponds to some subset of the lines, and is called a tour or a Hamiltonian cycle in graph theory. The tour length is the sum of the lengths of these lines.

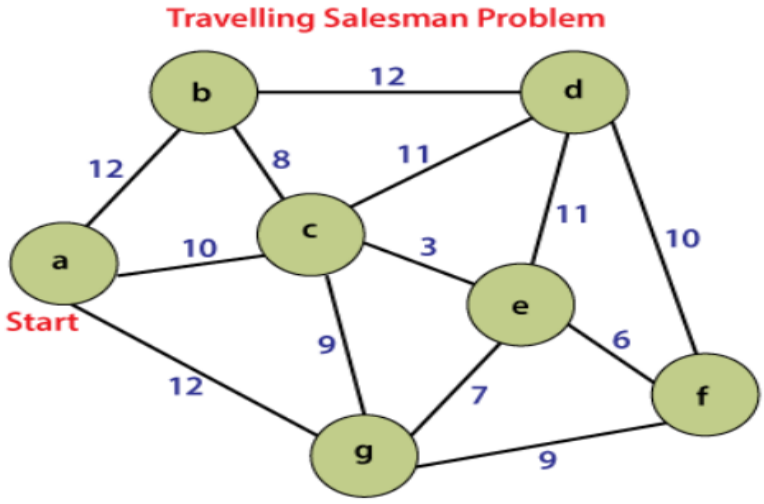


Figure 1.4: The cities are represented by letters, and the distances between them are indicated by the values of the lines.

The Traveling Salesman Problem (TSP) can be asymmetric or symmetric, depending on whether the direction of edges matters. For the asymmetric TSP on n cities,

$$x_{ij} = \begin{cases} 1 & \text{if the edge } i \to j \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

and, given the fact that each node must have exactly one incoming and one outgoing edge, leading to the classic assignment problem. However, this alone can lead to "subtours," or disjoint loops. To address this, additional "subtour elimination" constraints are added. The problem then becomes,

$$\min \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } j = 1, \ldots, n$$

$$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K| - 1 \text{ for all } K \subset \{1, \ldots, n\}$$

For any nonempty proper subset K of cities 1 to n, the cost c_{ij} can differ from c_{ji} ($c_{ij} \neq c_{ji}$), allowing for n(n-1) 0-1 variables. In the symmetric TSP, the direction of traversal doesn't matter, so $c_{ij} = c_{ji}$.

Now for the symmetric case,

$$\min \frac{1}{2} \sum_{j=1}^{n} \sum_{k \in J_{(j)}} c_k x_k$$

$$\sum_{k \in J_{(j)}} x_k = 2 \text{ for all } j = 1, \ldots, n$$

$$\sum_{j \in E(K)} x_j \leq |K| - 1 \text{ for all } K \subset \{1, \ldots, n\}$$

$$x_j = 0 \text{ or } 1 \text{ for all } j \in E$$

where J_j is the set of all undirected edges connected to node j and E(K) is the subset of all undirected edges connecting the cities in any proper, nonempty subset K of all cities.

The TSP problem has garnered attention not only for its simplicity in description but also for its complexity in optimization. It often appears as a subproblem in more intricate combinatorial problems, notably the vehicle routing problem. This problem involves deciding the optimal route for a fleet of vehicles to serve customers, considering factors like customer assignments and visitation order.

CHAPTER 2

GOALS AND OBJECTIVES

2.1 Motivation

- QFT and QPE have profound implications in the field of quantum computing and information processing.
- These play a pivotal role in various quantum algorithms, potentially offering exponential speedups in solving specific problems compared to classical algorithms (DFT).
- QPE has practical applications in fields such as quantum chemistry, cryptography, optimization, and machine learning. By exploring QPE, researchers can uncover new ways to leverage quantum computing for solving real-world problems more efficiently than classical approaches.
- They can lead to advancements in secure communication, encryption methods, and quantum key distribution which are vital in today's data-driven world.
- By bridging the gap between theory and practice, this thesis aims to explore the feasibility, scalability, and optimization potential of QPE and QFT implementations, paving the way for practical applications in quantum computing.
- TSP is used in logistics, transportation, manufacturing, and scheduling to find optimal routes, saving costs and improving efficiency.
- It plays a role in routing for circuit board design in electronics.
- Understanding and solving TSP contributes to advancements in computational complexity and optimization.
- TSP optimizes Global Navigation Satellite System (GNSS) surveying routes, reducing travel distance and costs, and enhancing data collection efficiency and environmental sustainability.
- Exploring these provides a deeper understanding of quantum mechanics, signal processing, and mathematical transformation.

2.2 Problem Statement

This research focuses on overcoming the limitations of classical computing algorithms through the exploration of quantum computing. It emphasizes the theoretical knowledge, implementation, and optimization of Quantum Fourier Transform (QFT), Quantum Phase Estimation (QPE), and other quantum algorithms to revolutionize computation across various domains. The study includes simulating QFT and QPE, visualizing qubit states, and addressing complex optimization problems like the Traveling Salesman Problem (TSP) using classical brute force methods and quantum approaches such as the Variational Quantum Eigensolver (VQE).

2.3 Goals and Objectives

- Investigate the theoretical foundations of Quantum Phase Estimation (QPE), Quantum Fourier Transform (QFT), and other quantum algorithms within the Qiskit environment.
- Gain proficiency in using Qiskit, an open-source quantum computing framework developed by IBM, for simulating and implementing quantum algorithms.
- Design and optimize quantum circuits for QPE, QFT, and other quantum algorithms using Qiskit's circuit construction and optimization tools.
- Develop a Qiskit program to solve the Traveling Salesman Problem using both brute force and Variational Quantum Eigensolver algorithms.

CHAPTER 3

LITERATURE REVIEW

Brigham and Morrow have provided a concise overview of classical classical Fourier transform [1]. A few years later the fundamentals of quantum mechanics encompassed key concepts such as qubits, superpositions, and entanglement. The historical development of various kinds of Fourier transform has been summarized in various works [2, 3, 4, 5, 6]. A concise overview of the classical Fourier transform is provided, outlining its principles and applications [7]. Limitations of classical algorithms are discussed, with an emphasis on scenarios where quantum algorithms, particularly the Quantum Fourier Transform (QFT), present potential advantages [8, 9]. The theoretical underpinnings of the QFT are explained, considering relevant literature. The quantum gates and principles involved in the QFT algorithm are discussed, referencing sources that delve into this quantum computational process. Examples illustrating how the QFT enhances computational efficiency in specific domains are presented [10]. Differences between the Classical Fourier Transform and Quantum Fourier Transform are analyzed, emphasizing scenarios where the QFT outperforms classical approaches. Interdisciplinary applications of the Quantum Fourier Transform are examined, shedding light on its relevance beyond quantum computing contexts [11].

The fast Fourier transform (FFT), recognized as one of the most successful classical algorithms of the 20th century, has been widely applied in various branches of computational science and engineering. The FFT algorithm's derivation emanates from a specific matrix decomposition of a Discrete Fourier Transform (DFT) matrix [12, 13, 14]. Similarly, the Quantum Fourier Transform (QFT) can also be derived through the decomposition of the DFT matrix. Quantum computing, a field experiencing significant growth with the potential to revolutionize computation, is explored here with a specific focus on the efficacy of the Quantum Fourier Transform (QFT). The foundation of quantum computing, especially within the context of the Quantum Fourier Transform, is being investigated, considering the broader implications of these points.

A comprehensive understanding of the foundational concepts and principles of quantum computing is essential. Literature on quantum computing by renowned authors such as Nielsen and Chuang provides insights into quantum

mechanics, quantum algorithms, and quantum information theory [15].

Incorporating these points allows for the exploration of the broader context of the Quantum Fourier Transform.

Quantum Phase Estimation (QPE) is considered a pivotal algorithm in quantum computing, with applications ranging from factorization to quantum chemistry simulations, as eigenvalues of unitary operators are estimated by it [16]. Originating with Kitaev's algorithm and further elucidated by Nielsen and Chuang's foundational text, QPE has undergone iterative refinements for enhanced efficiency and resilience against errors [17, 18]. Meanwhile, combinatorial optimization challenges like the Traveling Salesman Problem (TSP) have traditionally been tackled using classical brute force methods, which quickly become impractical for large instances due to their exponential time complexity [19, 20]. In contrast, the Variational Quantum Eigensolver (VQE) offers a quantum-inspired approach, where solutions to TSP and similar optimization problems are efficiently approximated using parameterized quantum circuits and classical optimization. Despite its promise, the practical implementation of VQE hinges on advances in quantum hardware and algorithmic enhancements to effectively address scalability challenges [21, 22].

CHAPTER 4

METHODOLOGY

In physics, engineering, and mathematics, the Fourier transform (FT) is an integral transform that converts a function into a form that describes the frequencies present in the original function. It converts a function from the time domain to its constituent frequency domain. It is widely used in various fields such as signal processing, communications, image processing etc.

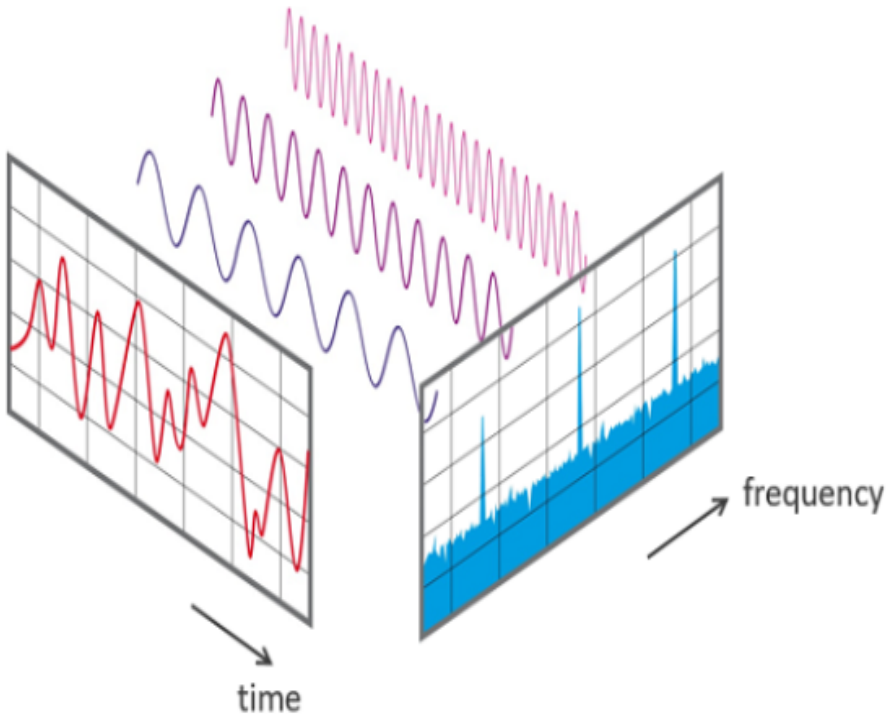


Figure 4.1: View of a signal in the time and frequency domain.

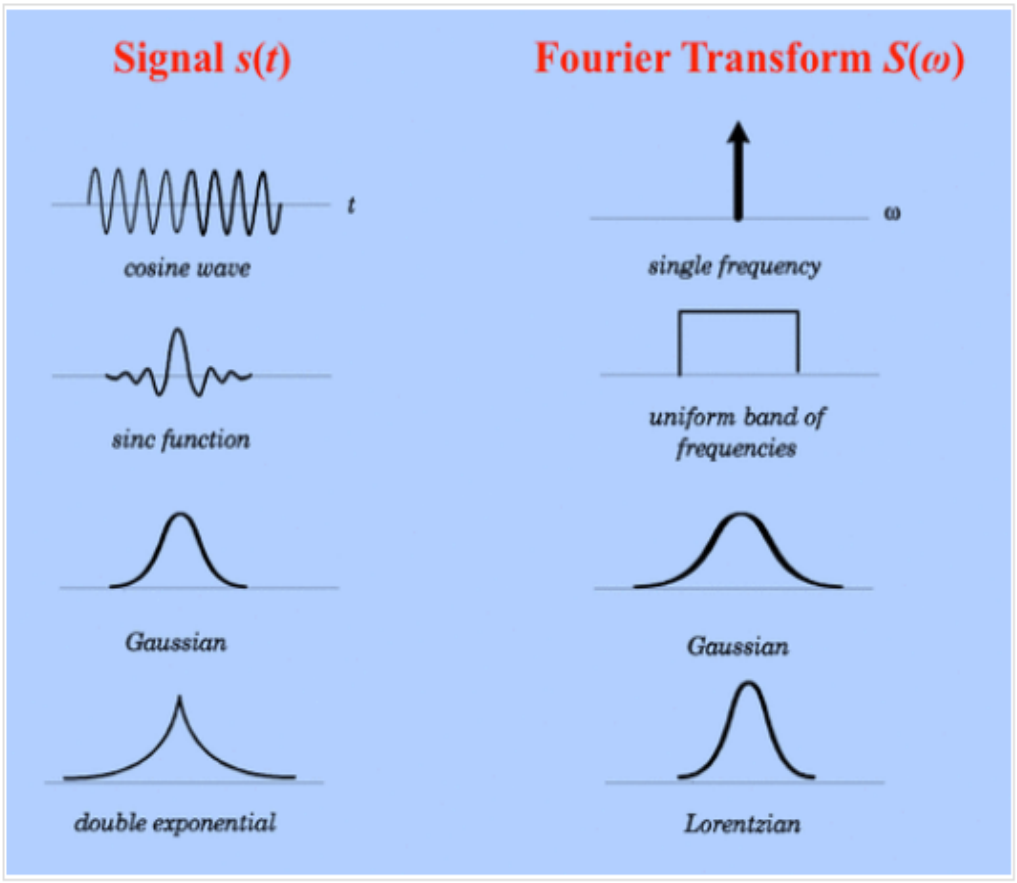


Figure 4.2: Some examples of Fourier transform from time domain to frequency domain.

4.1 Discrete Fourier Transform (DFT)

When the time function f(t) is band-limited, e.g. the number of nonzero Fourier coefficients is bounded, the Fourier series is finite and the transform, and its inverse, are defined by a trigonometric identity,

$$a_k = \sum_{n=0}^{N-1} x_i \cdot \exp\{-i \cdot 2\pi/N \cdot n \cdot k\} \tag{4.1}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} a_k \cdot \exp\{i \cdot 2\pi/N \cdot n \cdot k\} \tag{4.2}$$

Where the inverse transform is equal to the original time function or inverse Fourier transform allows the reconstruction of the original signal from its fre-

quency components.

4.2 Fast Fourier Transform (FFT)

The fast Fourier transform (FFT), is a computer algorithm that computes the discrete Fourier transform much faster than other algorithms. The savings in computer time can be huge; for example*, an N=2^{10} -point transform can be computed with the FFT 100 times faster than with the use of a direct approach. It has opened new avenues of scientific investigations. Problem-solving techniques once considered impractical are now efficiently implemented by the use of the FFT algorithm. It may be useful to point out that FFT not only reduces the computation time; it also substantially reduces round-off errors associated with these computations.

An N-point transformation by the direct method requires a time proportional to $N \log_2 N$. The approximate ratio of FFT to direct computing time is given by,

$$\frac{N \log_2 N}{N^2} = \frac{\log_2 N}{N} = \frac{\log_2 2^{10}}{N} \tag{4.3}$$

where $N = 2^{10}$, the FFT requires less than $1/100$ of the normal computing time.

4.3 Limitations of classical algorithms

- Many classical algorithms exhibit exponential time complexity for certain problems, making them impractical for large inputs and computations.
- Classical algorithms often face difficulty in efficiently solving problems classified as NP (Nondeterministic polynomial return different results every time for the same input values)-hard, where finding an optimal solution may take an impractical amount of time.
- Classical computers use bits that exist in one state at a time, limiting parallel processing (Simultaneous execution of multiple tasks or processes to solve problems more quickly) capabilities.
  Parallelism is crucial for certain computations, and classical algorithms may not exploit it as effectively as quantum algorithms can.

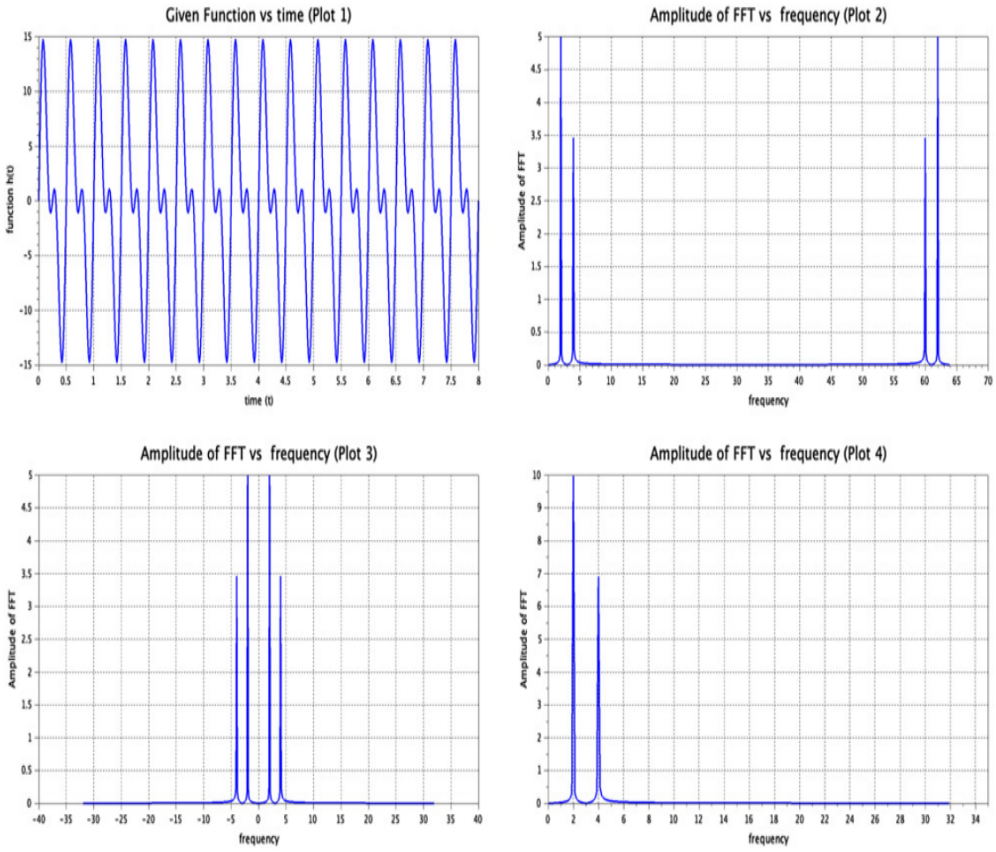


Figure 4.3: Graphical representation of FFT data for function $h(t) = 8\sin(6\pi t) \cdot \cos(2\pi t) + 3\sin(8\pi t) + 6\sin 4\pi t$.

- Factoring large numbers into prime factors is a computationally intensive task for classical algorithms, especially as numbers grow larger.
- Classical cryptographic systems are vulnerable to attack using powerful computational techniques, such as those employed by Shor's algorithm for integer factorization.
- Classical algorithms struggle to simulate complex quantum systems, limiting their ability to model and understand quantum phenomena accurately.

Acknowledging these limitations is crucial for understanding the motivation behind exploring alternative algorithms, such as quantum algorithms, to address challenges that classical algorithms face in certain domains.

4.4 Quantum Fourier Transform

One of the most useful ways of solving a problem in mathematics or computer science is to transform it into some other problem for which a solution is known. There are a few transformations of this type that appear so often and in so many different contexts that the transformations are studied for their own sake. A great discovery of quantum computation has been that some such transformations can be computed much faster on a quantum computer than on a classical computer, a discovery that has enabled the construction of fast algorithms for quantum computers.

One such transformation is the discrete Fourier transform (DFT). In the usual mathematical notation, the discrete Fourier transform takes as input a vector of complex numbers, x_0,.....,x_{N-1} where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of a complex number y_0,.....,y_{N-1}, defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \exp\left(\frac{-2\pi ijk}{N}\right) \tag{4.4}$$

The quantum Fourier transform is exactly the same transformation as DFT, If we decide to use quantum basis states $|j\rangle$ as the basis vectors, we can use DFT as usual. This is called QFT.

Of course, QFT is nothing but a quantum gate, and, thus, it needs to obey the properties of a quantum gate, namely, being unitary.

The conventional notation for the quantum Fourier transform is somewhat different. The quantum Fourier transform on an orthonormal basis $|0\rangle$,.....,$|N-1\rangle$ is defined to be a linear operator with the following action on the basis states,

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{-2\pi ijk}{N}\right) |k\rangle. \tag{4.5}$$

Equivalently, the action on an arbitrary state may be written,

$$\sum_{j=0}^{N-1} x_j \ |j\rangle \rightarrow \sum_{k=0}^{N-1} y_k \ |k\rangle \tag{4.6}$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j.

For n qubits $N = 2^n$, and N is the length of a vector (or data size),

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{-2\pi ijk}{N}\right)|k\rangle \tag{4.7}$$

where,

$$\begin{aligned} |j\rangle &= |j_{n-1}\ j_{n-2}\ j_{n-3}\cdots\cdots\cdots j_2\ j_1\ j_0\rangle \\ &= |j_{n-1}\rangle \otimes\ |j_{n-2}\rangle \otimes\ |j_{n-3}\rangle \otimes \cdots\cdots\cdots\cdots \otimes |j_2\rangle \otimes\ |j_1\rangle \otimes\ |j_0\rangle \\ &= \textbf{Each of these will represent the state of qubit} \end{aligned} \tag{4.8}$$

In which,

$$j = 2^{n-1} \times j_{n-1} + 2^{n-2} \times j_{n-2} + \cdots\cdots\cdots + 2^1 \times j_1 + 2^0 \times j_0 \tag{4.9}$$

$$\begin{aligned} |j\rangle &\rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} exp(\frac{-2\pi ijk}{2^n})|k\rangle \\ &= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} exp(-2\pi ij2^{-l})\ |k_l\rangle)\right] \\ &= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + exp(-2\pi ij2^{-l})|1\rangle\right] \end{aligned} \tag{4.10}$$

where j_{n-1} is most significant bit (M.S.B) and j_0 is least significant bit (L.S.B).

for example, if

$$|j\rangle = |1011\rangle = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1$$

k will also adhere to the same principles.

In the form of the matrix,

$$DFT = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 \cdots \\ 1 & \omega^{-1.1} & \omega^{-1.2} & \omega^{-1.3} \cdots \\ 1 & \omega^{-2.1} & \omega^{-2.2} & \omega^{-2.3} \cdots \\ 1 & \omega^{-3.1} & \omega^{-3.2} & \omega^{-3.3} \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.11}$$

where ω = $\exp\left(\frac{2\pi i}{N}\right)$ is the N-th root of unity.
If we decide to use quantum basis states $|j\rangle$ as the basis vectors, we can use DFT as usual. and this is called QFT.

$$U_{QFT} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 \cdots \\ 1 & \omega^{-1.1} & \omega^{-1.2} & \omega^{-1.3} \cdots \\ 1 & \omega^{-2.1} & \omega^{-2.2} & \omega^{-2.3} \cdots \\ 1 & \omega^{-3.1} & \omega^{-3.2} & \omega^{-3.3} \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.12}$$

Therefore,

$$U_QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{kj} |j\rangle \tag{4.13}$$

U_{QFT} is a symmetric and unitary matrix.
Inverse Fourier transform,

$$U_{IQFT} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 \cdots \\ 1 & \omega^{1.1} & \omega^{1.2} & \omega^{1.3} \cdots \\ 1 & \omega^{2.1} & \omega^{2.2} & \omega^{2.3} \cdots \\ 1 & \omega^{3.1} & \omega^{3.2} & \omega^{3.3} \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.14}$$

4.5 Quantum Phase Estimation

Quantum Phase Estimation (QPE) is a fundamental algorithm in quantum computing. It's crucial for various applications, including Shor's factoring algorithm, quantum sampling, and many more. QPE often incorporates Kitaev's algorithm and the inverse Quantum Fourier Transform (IQFT), serving as a core

component in these quantum processes. The Fourier transform is crucial for phase estimation, a fundamental procedure in many quantum algorithms. Given a unitary operator U with an eigenvector $|\psi\rangle$ and an eigenvalue $\exp(2\pi i\phi)$, where ϕ is unknown, the goal of the phase estimation algorithm is to determine.

$$U|\psi\rangle = \lambda|\psi\rangle$$

where eigenvalue of unitary matrix is $\lambda = \exp(2\pi i\phi)$. The phase of the unitary matrix can be written as $\phi_n = 0.x_1x_2x_3\ldots x_n$, where n is the number of qubits used for phase estimation. The estimated variable ($\hat{\phi}$) can be expressed as a binary representation,

$$\hat{\phi} = \frac{x_1}{2^1} + \frac{x_2}{2^2} + \frac{x_3}{2^3} + \cdots + \frac{x_n}{2^n}$$

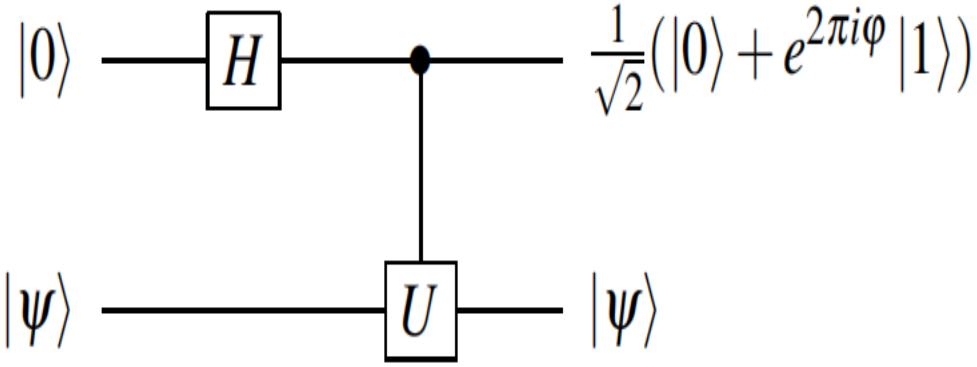


Figure 4.4: The circuit consists of one qubit and an eigenstate, a Hadamard gate (H), and a rotation gate (U). The output of the circuit contains the phase $\frac{1}{\sqrt{2}}$ ($|0\rangle$+ $e^{2\pi i\phi}$ $|1\rangle$).

QPE is a common quantum algorithm for estimating phases of unitary operators, often implemented using the inverse Quantum Fourier Transform (QFT). This method comprises two stages. The first stage starts with n-qubits initialized at $|0\rangle$, prepares the state $|\psi\rangle$. The second stage uses the inverse quantum Fourier transform to estimate the binary digits of the phase. The mathematical analysis,

$$\frac{1}{\sqrt{2}}(|0\rangle+e^{2\pi i2^{n-1}\phi}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle+e^{2\pi i2^{n-2}\phi}|1\rangle).....\frac{1}{\sqrt{2}}(|0\rangle+e^{2\pi i\phi}|1\rangle) = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{\frac{2\pi i\phi k}{2^n}}|k\rangle$$

ϕ = $\frac{x}{2^n}$ where x =$\sum_{i=0}^{n-1} 2^i x_i$ therefore rearranging the terms,

$$\frac{1}{\sqrt{2}}(|0\rangle+e^{\pi i 2^n\phi}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle+e^{\frac{\pi}{2}i2^n\phi}|1\rangle).....\frac{1}{\sqrt{2}}(|0\rangle+e^{\frac{\pi}{2^{n-1}}i2^n\phi}|1\rangle) = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{\frac{2\pi i\phi k}{2^n}}|k\rangle$$

By applying the inverse quantum Fourier transform, we can recover the unknown phase.

$$\frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{\frac{2\pi i\phi k}{2^n}}|k\rangle|\psi\rangle \rightarrow |\widetilde{\phi}\rangle|\psi\rangle$$

where $|\widetilde{\phi}\rangle$ denotes a state which is a good estimator for ϕ when measured.

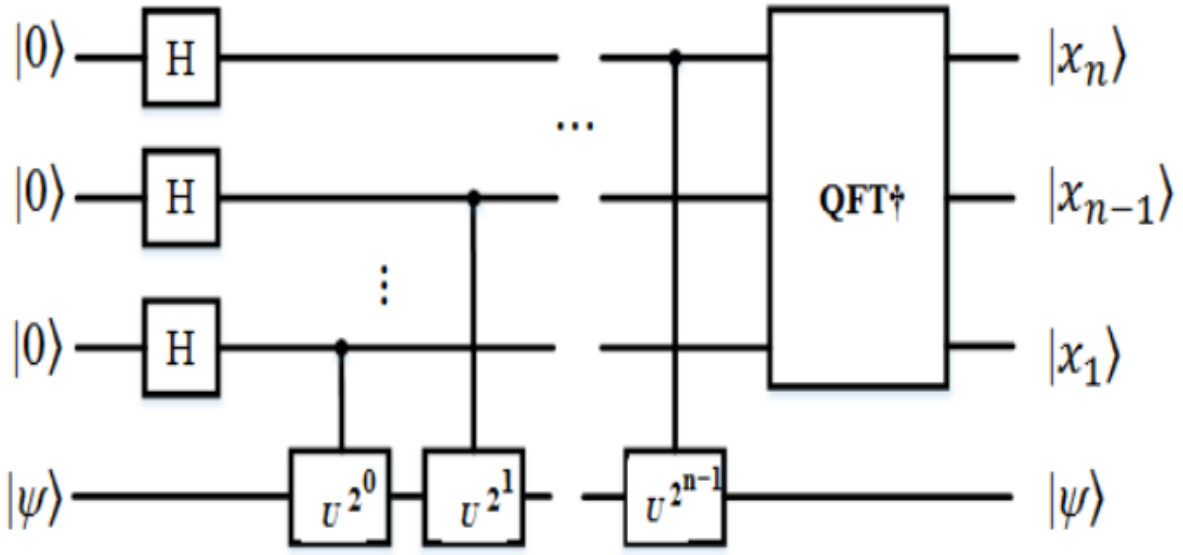


Figure 4.5: In Quantum Phase Estimation, the inverse Quantum Fourier Transform (QFT) is employed for the estimation of the quantum phase.

4.6 Quantum gates

A quantum gate is a gate that performs a unitary operation. In a classical computer, in general, the electrical signals flow through some physical gates (e.g. a NAND gate formed by 4 transistors) in the space. In quantum computing, usually, the qubit and its "signal" do not flow in space. Instead, we apply operations such as microwave or laser pulses to operate/manipulate/change the qubit. Therefore, the flow of a quantum algorithm usually represents the flow of time instead of a physical layout in space.

4.6.1 Hadamard gate

The Walsh–Hadamard Gate is commonly called the Hadamard gate. The symbol of a Hadamard gate is H. The Hadamard gate is so special because it has no classical counterpart. The Hadamard gate is a 1-qubit gate, but, of course, we can also form a multi-qubit Hadamard gate through the tensor products of 1-qubit Hadamard gates. The matrix of H is,

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.15}$$

We are working on a basis formed by eigenvectors of σ_z,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; and \; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.16}$$

On applying the Hadamard gate on the basis vector, they are rotated to a superposition of the basis vector.

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

and

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{4.17}$$

There is no classical counterpart for the Hadamard gate because it is impossible to find a classical gate that gives "$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$" as "0+1" is meaningless in the classical Boolean logic.
The inverse of the Hadamard gate (H^{-1}) equals the Hadamard gate itself.

$$H = H^{-1} \tag{4.18}$$

4.6.2 Controlled-NOT gate

In classical logic, the XOR gate is a 2-input gate. Similarly, an XOR quantum gate is 2-qubit. It is more commonly known as the CNOT gate, which stands for Controlled-NOT gate,

$$U_{XOR}|ab\rangle = |aa \oplus b\rangle \tag{4.19}$$

where,

$$|aa \oplus b\rangle = |a, a \oplus b\rangle = |a\rangle \otimes \ |a \oplus b\rangle \tag{4.20}$$

After the operation, the first number is still a, but the second number becomes a $\oplus$ b, where $\oplus$ is the classical exclusive or (XOR) logical operation.

$$\begin{aligned} U_{XOR}|00\rangle &= |0, 0 \oplus 0\rangle = |0, 0\rangle = |00\rangle \\ U_{XOR}|01\rangle &= |0, 0 \oplus 1\rangle = |0, 1\rangle = |01\rangle \\ U_{XOR}|10\rangle &= |1, 1 \oplus 0\rangle = |1, 1\rangle = |11\rangle \\ U_{XOR}|11\rangle &= |1, 1 \oplus 1\rangle = |1, 0\rangle = |10\rangle \end{aligned} \tag{4.21}$$

The matrix of U_{XOR} is

$$U_{XOR} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.22}$$

Since this is a 2-qubit gate for operating on a vector in the four-dimensional space, the matrix must be 4×4.

The output depends on a $\otimes$ b, where a is the value of the first qubit and b is the value of the second qubit in the basis state. Based on the operations discussed, this is also equivalent to saying "If a is zero, keep b unchanged and if a is one, negate b (i.e. apply NOT to b)." Therefore, it is also called Controlled-NOT. Whether we will apply NOT to the second qubit depends on a (i.e. controlled by a). Therefore, the first qubit is called the control qubit and the second one is called the target qubit. For example, for $|10\rangle$, a is 1. Therefore it will apply NOT to b, which is 0. b will be negated to 1 and the state becomes $|11\rangle$. But for $|00\rangle$, a is 0. Therefore, b is unchanged and is still 0 and thus the basis state is still $|00\rangle$.

4.6.3 Control Phase shift gate

We can define a 2-qubit controlled phase shift gate as the following:

$$U_{CPS,\Phi} \ |ab\rangle = \exp(i(a \cdot b)\Phi) \ |ab\rangle \tag{4.23}$$

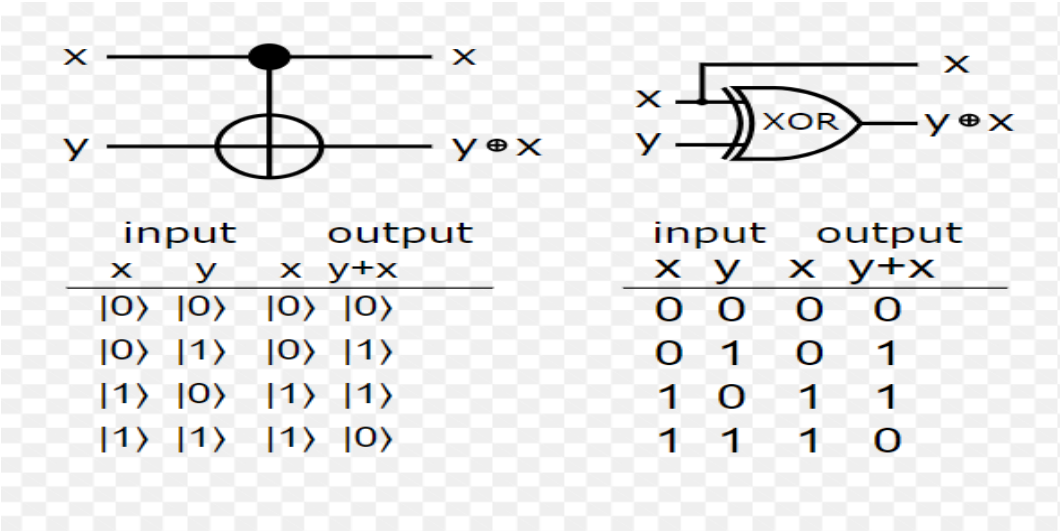


Figure 4.6: Control-Not gate with the comparison of classical XOR gate.

where a and b are just the numbers in the first and the second qubit in the basis states, respectively. They can be either 0 or 1. Here, "·" means the classical logic AND operation. Therefore, a . b equals 1 only when both a and b are 1. Let us see how it changes the basis states,

$$\begin{aligned} U_{CPS,\Phi}\ |00\rangle &= \exp(i(0\cdot 0)\Phi)\ |00\rangle = |00\rangle \\ U_{CPS,\Phi}\ |01\rangle &= \exp(i(0\cdot 1)\Phi)\ |01\rangle = |01\rangle \\ U_{CPS,\Phi}\ |10\rangle &= \exp(i(1\cdot 0)\Phi)\ |10\rangle = |10\rangle \\ U_{CPS,\Phi}\ |11\rangle &= \exp(i(1\cdot 1)\Phi)\ |11\rangle = \exp(i\Phi)\ |11\rangle \end{aligned} \tag{4.24}$$

Therefore, only $|11\rangle$ is changed with an extra phase. When the first qubit is 0 (i.e. in the $|00\rangle$ and $|01\rangle$ cases), nothing is applied to the second qubit. But when the first qubit is 1, a phase shift gate is applied. Therefore, a phase shift gate is applied to the second qubit when the first qubit is 1. So, the first qubit is the control qubit and the second qubit is the target qubit in this controlled phase shift gate.

The Matrix of $U_{CPS,\Phi}$ is,

$$U_{CPS,\Phi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \exp(i\Phi) \end{pmatrix} \tag{4.25}$$

Since this is a 2-qubit gate for operating on a vector in the four-dimensional space, the matrix (operator) must be 4 × 4.

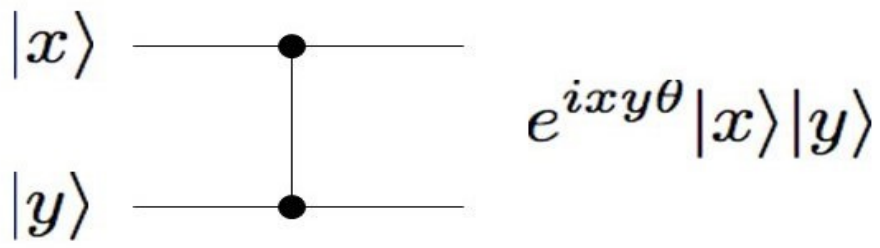


Figure 4.7: Control phase shift gate.

4.6.4 Swap gate

A SWAP gate, as its name implies, swaps the numbers in the basis states of a 2-qubit register. If we consider only the basis states, it is equivalent to swapping the states of two electrons. The definition is,

$$U_{SWAP}\ |ab\rangle = |ba\rangle \tag{4.26}$$

where a and b are just the numbers of the first and the second qubits in the basis states, respectively. They can be either 0 or 1. The easiest way to gain a deeper understanding is just to plug in all the basis states and we get,

$$\begin{aligned} U_{SWAP}\ |00\rangle &= |00\rangle \\ U_{SWAP}\ |01\rangle &= |10\rangle \\ U_{SWAP}\ |10\rangle &= |01\rangle \\ U_{SWAP}\ |11\rangle &= |11\rangle \end{aligned} \tag{4.27}$$

Therefore, only $|01\rangle$ and $|10\rangle$ are changed (to each other) under the SWAP operation. For $|00\rangle$ and $|11\rangle$ they are unchanged because swapping "0" and "0" or "1" and "1" is equivalent to doing nothing.

The matrix of U_{SWAP} is,

$$U_{CPS,\Phi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.28}$$

Let us try an example,

$$U_{SWAP}|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle \tag{4.29}$$

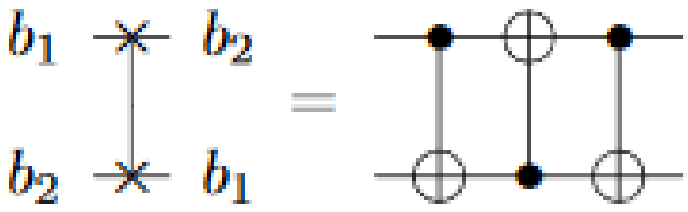


Figure 4.8: Swap gate.

4.7 Bloch Sphere

Bloch Sphere is a very useful tool for us to understand the operation of quantum gates. If we use it correctly, we can use it to help us construct quantum gates and understand the underlying physics.
Therefore Bloch sphere is a useful geometric representation of the state space of a qubit. It is a sphere with a diameter of one, where the North Pole and the South Pole represent the two orthogonal states of the qubit, $|0\rangle$ and $|1\rangle$. The equator of the sphere represents a superposition of these two states.

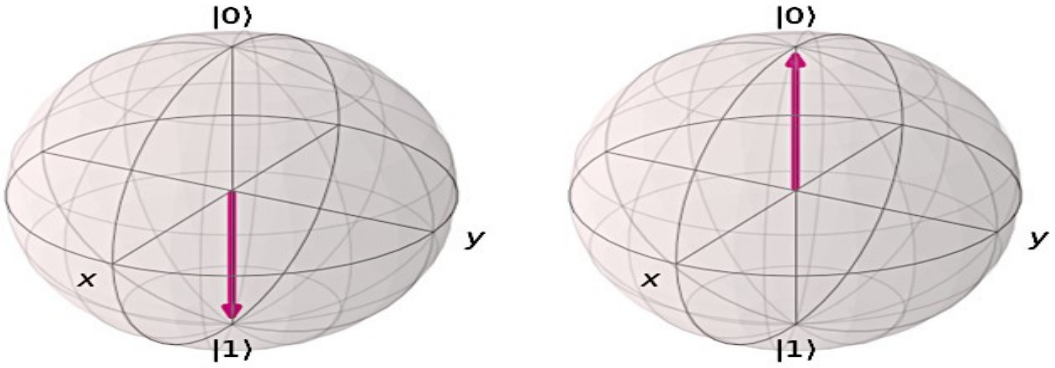


Figure 4.9: left: The $|1\rangle$ qubit state is represented on the Bloch sphere. Right: The $|0\rangle$ qubit state is represented on the Bloch sphere.

Example-: For n=1, qubit=2^1=2
from eq. 4.4,

$$\begin{aligned} QFT|0\rangle &= \frac{1}{\sqrt{2}}\sum_{k=0}^{2-1} exp(2\pi\iota 0k/2)|k\rangle \\ &= \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle] \\ &= |+\rangle \end{aligned} \tag{4.30}$$

and

$$\begin{aligned} QFT|1\rangle &= \frac{1}{\sqrt{2}}\sum_{k=0}^{2-1} exp(2\pi\iota 1k/2)|k\rangle \\ &= \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle] \\ &= |-\rangle \end{aligned} \tag{4.31}$$

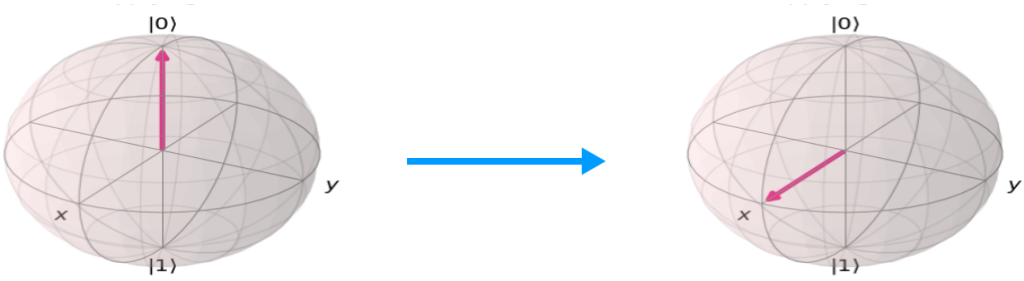


Figure 4.10: Figure for equation 4.30.

and

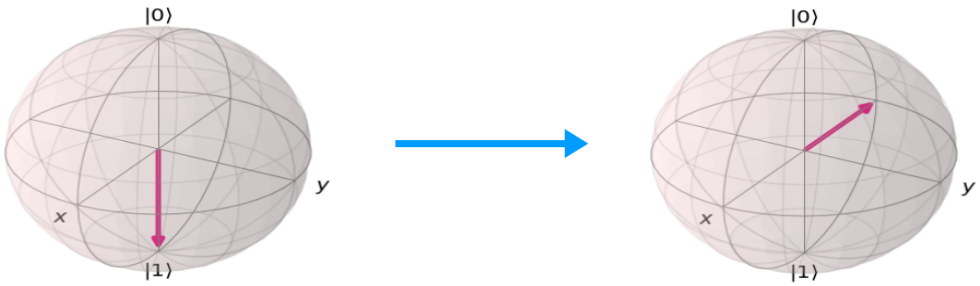


Figure 4.11: Figure for equation 4.31.

CHAPTER 5

TRAVELING SALESMAN PROBLEM AND ITS SOLUTION BY CLASSICAL BRUTE FORCE AND VARIATIONAL QUANTUM EIGENSOLVER ALGORITHM

The Traveling Salesman Problem (TSP) has puzzled mathematicians for years, with no satisfying solution found yet. In the 1800s, Sir William Rowan Hamilton and Thomas Penyngton Kirkman, Irish and British mathematicians, developed related mathematics.

5.1 Definition

Suppose a salesperson needs to travel from one city to all the other cities exactly once to sell his products and then return to the starting city. He wants to do this while covering the minimum total distance.

Given n number of cities to be visited, the number of paths that must be explored is $n!$, and the total number of possible routes can be given as $\frac{(n-1)!}{2}$.

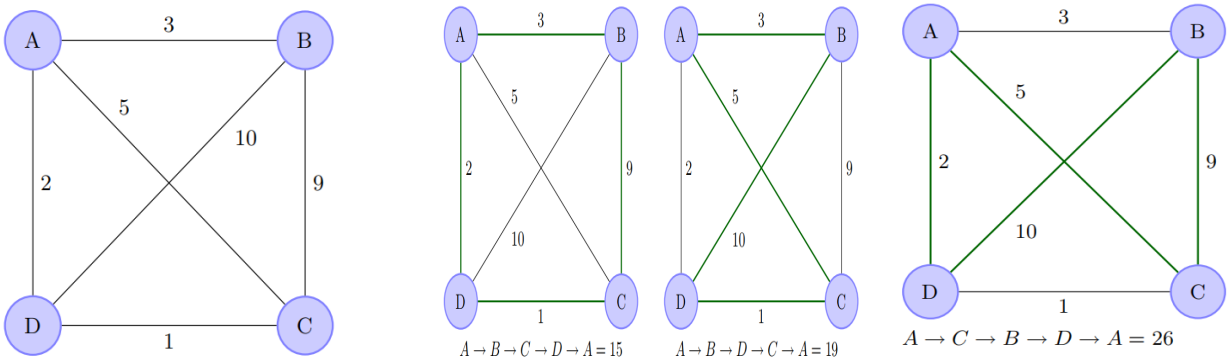
5.2 Methods of TSP solution

5.2.1 Classical Brute force algorithm

A brute force algorithm is a technique for problem-solving that exhaustively explores all potential solutions without employing any particular heuristics or optimizations. It works by systematically generating and evaluating each possible solution until the optimal one is found. While this method ensures correctness, it can be inefficient and impractical for problems with large solution spaces due to its exhaustive nature.

Steps to solve the TSP using the brute-force method:

- Calculate the total number of possible tours.
- List all possible tours.
- Calculate the distance of each tour.
- Choose the shortest tour as the optimal solution.



(a) (b) (c)

Figure 5.1: The optimal path is $A \to B \to C \to D \to A$, of value 15.

5.2.2 Variational quantum eigensolver algorithm

The Variational Quantum Eigensolver (VQE) algorithm is a hybrid quantum-classical algorithm designed to find the minimal eigenvalue and corresponding eigenvector of a given Hamiltonian. It consists of two main components: one executed on a quantum computer and the other on a classical computer.

On the quantum side, the algorithm utilizes a parameterized quantum circuit (PQC) to prepare a quantum state from an initial state. By adjusting the parameters of the PQC, different quantum states can be generated. These quantum states are then used to calculate the expectation value of the Hamiltonian through sampling outcomes.

On the classical side, the expectation values obtained from the quantum computations are processed to determine the next set of parameters for the PQC. This iterative process continues until the algorithm converges to the minimal eigenvalue, providing a solution to the optimization problem at hand.

Overall, the VQE algorithm leverages quantum and classical computing resources synergistically to tackle optimization problems efficiently, making it a promising approach for current quantum devices with limited capabilities.

In simple words, The VQE algorithm finds the lowest eigenvalue and its corresponding eigenvector for a given Hamiltonian. It uses the variational principle, where the expectation value of the Hamiltonian(H) in any quantum state($|\psi\rangle$) is greater than or equal to the lowest eigenvalue(λ_{min}).

$$\lambda_{min} \leq \langle\psi|H|\psi\rangle \tag{5.1}$$

The algorithm operates in two parts: one on quantum computers and the other on classical computers.

CHAPTER 6

RESULTS

1. A Quantum Fourier Transform circuit and Bloch spheres for different numbers of qubits were successfully simulated using the Qiskit platform.

   The code includes the implementation of the QFT circuit using Qiskit, a widely used open-source quantum computing framework. This code involves defining qubits, applying quantum gates such as Hadamard gates, controlled not gates, and controlled phase gates, and executing the QFT algorithm.

```
In [1]: from qiskit import QuantumCircuit, execute, Aer, BasicAer,QuantumRegister,ClassicalRegister
        from qiskit.visualization import plot_histogram
        import numpy as np
        pi =np.pi
        from qiskit import *
```

```
In [2]: n=4

        qftn=QuantumCircuit(n,name='QFT')

        for i in range(n-1,-1,-1):
            qftn.h(i)
            p=0
            for j in range(i):
                p+=1
                qftn.cp(pi/(2**p),i-j-1,i)
            qftn.barrier()

        for i in range(int(n/2)):
            qftn.swap(i,n-1-i)

        qftn.draw('mpl')
```

Figure 6.1: The program of n=4 qubit vector space

The QFT circuit is a fundamental operation in quantum computing. Here the diagram i.e. 6.2 consists of horizontal lines representing qubits and vertical lines representing quantum gates applied to the qubits. The specific type of gate is indicated by the symbol used on the line. For example, a Hadamard gate is represented by an “H” symbol on a vertical line. Each gate manipulates the quantum gate of the qubits according to the principles of quantum mechanics. The purple line represents a specific type of gate or operation that is being used in the circuit.

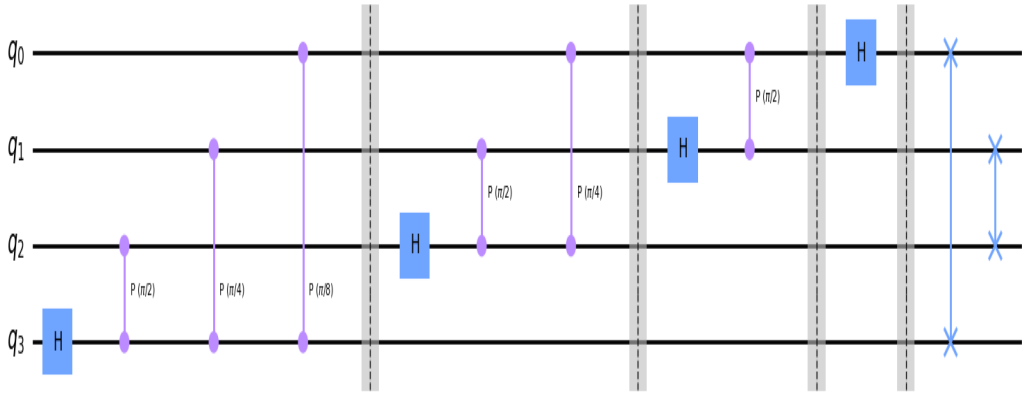


Figure 6.2: Quantum Fourier Transform Circuit

The code associated with the figure may provide instructions for generating the Bloch sphere visualization of the quantum state of a qubit. The figure may illustrate a specific qubit state $|0111>$. This state would be represented by a vector on the Bloch sphere that is located at a specific point on the sphere's surface. The position of the vector on the sphere represents the state of the qubit.

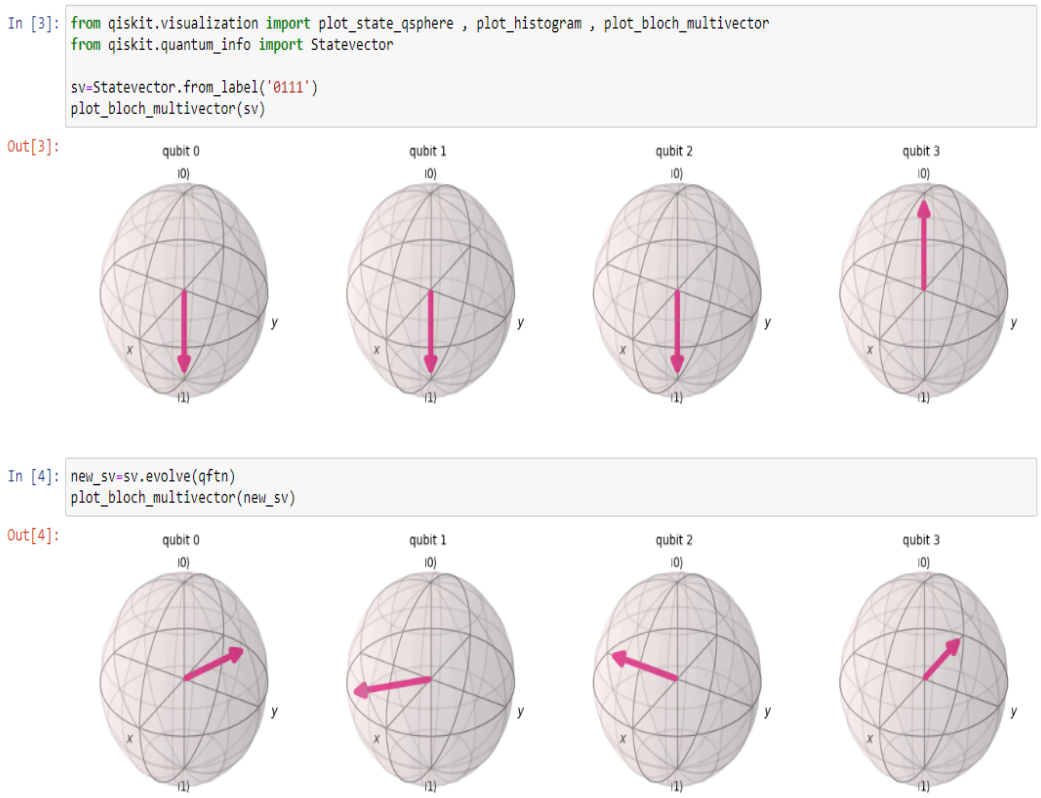


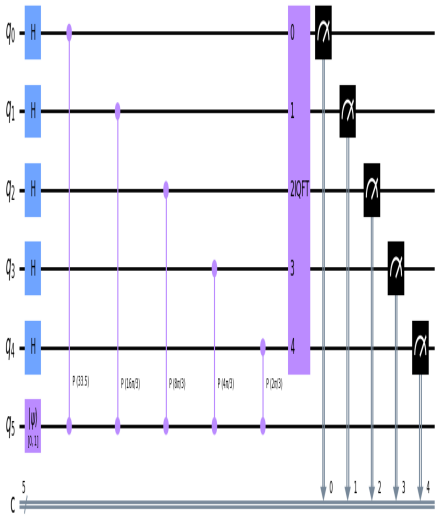
Figure 6.3: Bloch sphere which is representing the $|0111>$ qubit vector space. and below one will be the state after evolving.

2. The code represents a quantum circuit implementing the Quantum Phase Estimation (QPE) algorithm on a 5-qubit system. The goal of this code is for the phase(here $\theta = \frac{1}{3}$) encoded in the state of the system qubit to be estimated using the counting qubits. After the Hadamard gates and controlled phase rotations are applied, the phase information is encoded in the relative phases of the states of the counting qubits. The inverse QFT is used to translate this phase information into a form that can be measured.

```
In [48]: from qiskit import QuantumCircuit, transpile, assemble
         from qiskit.circuit.library import QFT
         from qiskit import Aer, execute
         from qiskit.visualization import plot_histogram
         import numpy as np
         import math
         import matplotlib.pyplot as plt
         from itertools import product

         pi=np.pi
```

```
In [49]: c=5
         n=1
         qc=QuantumCircuit(c+n,c)

         for qubit in range(c):
             qc.h(qubit)

         init_st=[0, 1]
         qc.initialize(init_st,c+n-1)

         theta=1/3

         for x in range(c):
             exponent=2**(c-x-1)
             qc.cp(2*pi*theta*exponent,x,c+n-1)

         qc.append(QFT(c).inverse(),range(c))

         for i in range(c):
             qc.measure(i,i)

         qc.draw('mpl')
```

(a) Qiskit Code



(b) Quantum Circuit

Figure 6.4: Quantum circuit of the quantum phase estimation.

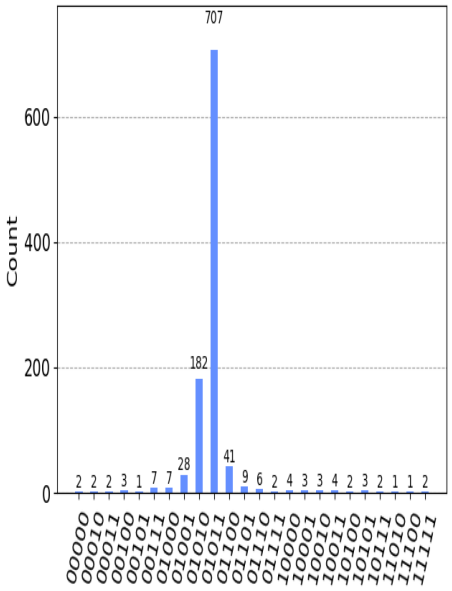
The histogram in Figure 6.5(b) is a visual representation of the results obtained from running the quantum circuit. Each bar in the histogram represents a different measurement result, which is shown as a binary number. The height of each bar indicates how many times that particular result was measured out of the 2048 runs of the circuit. In simpler terms, taller bars mean those results were more frequent.

In Figure 6.6(a) when the circuit is run on a quantum simulator or quantum computer, the measurement results will be binary strings corresponding to the estimated phase. In an ideal scenario with infinite precision, these results would correspond to the binary representation of θ.

The histogram in Figure 6.6(b) is a bar chart that shows the probabilities of each possible 5-bit state after executing the quantum circuit 10,000 times. The code identifies the most probable state by finding which state was measured most frequently. This most frequent state is then used to estimate

```
In [5]: backend = Aer.get_backend('qasm_simulator')
shots=2048
transpiled_qc = transpile(qc, backend)

# Assemble the transpiled circuit
qobj = assemble(transpiled_qc)

# Run the transpiled circuit directly using the backend's run method
results = backend.run(transpiled_qc).result()

answer=results.get_counts()
plot_histogram(answer)
```

(a) Qiskit code

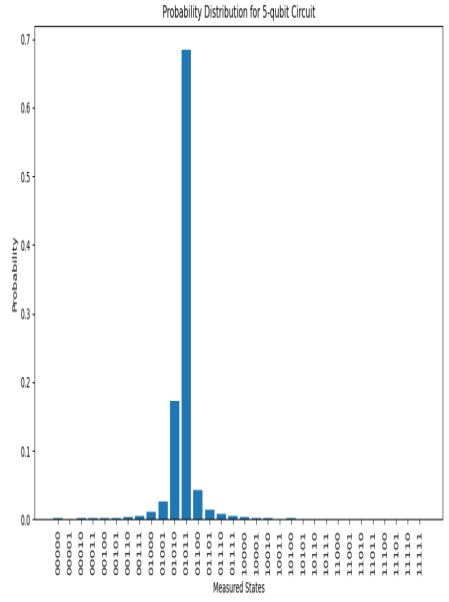


(b) Histogram Visualization of most frequent state.

Figure 6.5: Histogram Visualization of QPE circuit.

```
In [51]: simulator = Aer.get_backend('qasm_simulator')
#job_sim = execute(qc, simulator, shots=10000)


# Transpile the quantum circuit for the target backend
transpiled_qc = transpile(qc, backend=simulator)

# Execute the transpiled circuit on the backend
job_sim = simulator.run(transpiled_qc, shots=10000)

# Get the result from the simulator
result_sim = job_sim.result()
counts = result_sim.get_counts(qc)

# Create a list of all possible c-qubit binary strings
all_states = [''.join(map(str, state)) for state in product([0, 1], repeat=c)]

# Initialize probabilities for all states
probabilities = {state: counts.get(state, 0) / 10000 for state in all_states}

# Plot the probabilities
plt.figure(figsize=(10, 6))
plt.bar(probabilities.keys(), probabilities.values())
plt.xlabel('Measured States')
plt.ylabel('Probability')
plt.title('Probability Distribution for 5-qubit Circuit')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()


max_probability_state = max(probabilities, key=probabilities.get)
max_probability = probabilities[max_probability_state]

print("State with the highest probability:", max_probability_state)
print("Probability:", max_probability)


measured_theta = int(max_probability_state, 2)/(2**c)
print("Using",c,"qubits with theta = ",theta,", measured_theta = ",measured_theta)
```

(a) Qiskit code



(b) Probability of the quantum states.

Figure 6.6: Histogram showing the probabilities of each possible state.

the phase value θ. The binary representation of the most probable state is converted back to a phase value. For $\theta = \frac{1}{3}$, the most frequent result should

be close to the binary equivalent of $\frac{1}{3}$, which is approximately 01011 in binary with 5 bits. Consequently, the calculated measured theta should be close to the actual phase θ.

This difference was calculated to be 3.125%, highlighting the precision of the quantum phase estimation process in determining the phase information of the quantum state.

3. **TSP solution using brute force algorithm:**

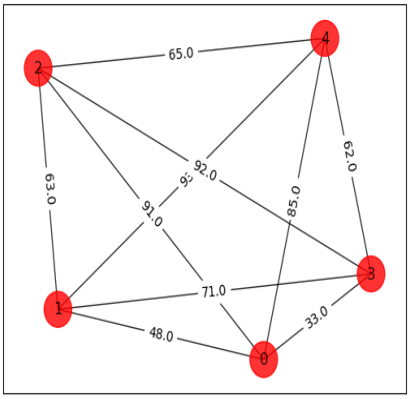
   The input for this code consists of generating a random instance of the Traveling Salesman Problem (TSP) with 5 cities. The number of nodes (n) is set to 5, and the **'Tsp.create_random_instance(n, seed=123)'** function generates a random graph representing the distances between these cities. This graph is then converted into an adjacency matrix **(adj_matrix)**, which stores the distances between each pair of cities. The nodes are colored red **('colors = ["r" for node in tsp.graph.nodes]')**, and their positions are stored in the pos variable. The brute force approach is then used to compute the shortest route by checking all possible permutations of the cities.

```
In [27]: # useful additional packages
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
import time
import dimod
from qiskit.circuit.library import TwoLocal
from qiskit_optimization.applications import Maxcut, Tsp
from qiskit_algorithms import SamplingVQE, NumPyMinimumEigensolver
from qiskit_algorithms.optimizers import SPSA
from qiskit_algorithms.utils import algorithm_globals
from qiskit.primitives import Sampler
from qiskit_optimization.algorithms import MinimumEigenOptimizer
from itertools import permutations
from qiskit import Aer
from qiskit_algorithms import VQE
from qiskit_algorithms.optimizers import COBYLA
from qiskit_optimization.converters import QuadraticProgramToQubo
```

```
In [28]: def draw_graph(G, colors, pos):
    default_axes = plt.axes(frameon=True)
    nx.draw_networkx(G, node_color=colors, node_size=600, alpha=0.8, ax=default_axes, pos=pos)
    edge_labels = nx.get_edge_attributes(G, "weight")
    nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=edge_labels)
# Generating a graph of 3 nodes
n = 5
num_qubits = n**2
tsp = Tsp.create_random_instance(n, seed=123)
adj_matrix = nx.to_numpy_array(tsp.graph)
print("Distance\n", adj_matrix)

colors = ["r" for node in tsp.graph.nodes]
pos = [tsp.graph.nodes[node]["pos"] for node in tsp.graph.nodes]

# Timing the brute force TSP computation
start_time = time.time()
#best_distance, best_order = brute_force_tsp(adj_matrix, n)
end_time = time.time()
elapsed_time = end_time - start_time
#print(f"Elapsed time: {elapsed_time:.4f} seconds")
draw_graph(tsp.graph, colors, pos)
```

(a) Qiskit code of TSP problem for n=5 cities.

```
Distance
 [[ 0. 48. 91. 33. 85.]
 [48.  0. 63. 71. 93.]
 [91. 63.  0. 92. 65.]
 [33. 71. 92.  0. 62.]
 [85. 93. 65. 62.  0.]]
```



(b) Input: This graph visualizes the distance between the cities.

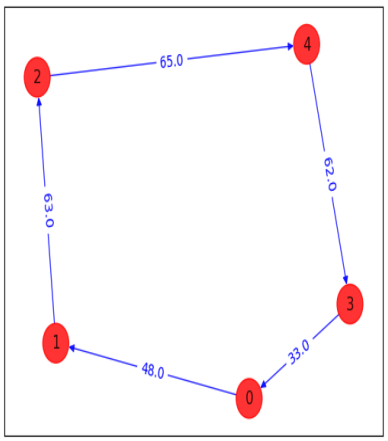
Figure 6.7: Input case for TSP.

The output of this code includes both textual and graphical representations of the solution. Textually, the best route (order of cities) and the total distance for this route are printed on the console. For example, the output might show: "Best order from brute force = (0, 1, 2, 4, 3) with total distance

```
In [29]: from itertools import permutations
def brute_force_tsp(w, N):
    a = list(permutations(range(1, N)))
    last_best_distance = 1e10
    for i in a:
        distance = 0
        pre_j = 0
        for j in i:
            distance = distance + w[j, pre_j]
            pre_j = j
        distance = distance + w[pre_j, 0]
        order = (0,) + i
        if distance < last_best_distance:
            best_order = order
            last_best_distance = distance
            print("order = " + str(order) + " Distance = " + str(distance))
    return last_best_distance, best_order
best_distance, best_order = brute_force_tsp(adj_matrix, n)
print(
    "Best order from brute force = "
    + str(best_order)
    + " with total distance = "
    + str(best_distance)
)
def draw_tsp_solution(G, order, colors, pos):
    G2 = nx.DiGraph()
    G2.add_nodes_from(G)
    n = len(order)
    for i in range(n):
        j = (i + 1) % n
        G2.add_edge(order[i], order[j], weight=G[order[i]][order[j]]["weight"])
    default_axes = plt.axes(frameon=True)
    nx.draw_networkx(
        G2, node_color=colors, edge_color="b", node_size=600, alpha=0.8, ax=default_axes, pos=pos
    )
    edge_labels = nx.get_edge_attributes(G2, "weight")
    nx.draw_networkx_edge_labels(G2, pos, font_color="b", edge_labels=edge_labels)
# Timing the brute force TSP computation
start_time = time.time()
end_time = time.time()
elapsed_time = end_time - start_time
#print(f"Elapsed time: {elapsed_time:.4f} seconds")
draw_tsp_solution(tsp.graph, best_order, colors, pos)
```

(a) Qiskit code to solve the TSP problem using the Brute force algorithm.

```
order = (0, 1, 2, 3, 4) Distance = 350.0
order = (0, 1, 2, 4, 3) Distance = 271.0
Best order from brute force = (0, 1, 2, 4, 3) with total distance = 271.0
```



(b) Output: This is showing the optimal tour (Shortest tour) for the n=5 cities.

Figure 6.8: Output to solve TSP using Brute force algorithm.

= 271". Graphically, the initial setup of the graph is plotted, showing all cities and their connections, with nodes colored red. After computing the optimal TSP route, the graph is plotted again to visualize the solution, highlighting the optimal route that connects all cities in the order found by the brute force method. This graphical output helps in understanding the shortest path visually.

4. **TSP solution using VQE algorithm:**

   Now for using the variational quantum eigensolver algorithm, the TSP problem is converted to a quadratic program and subsequently to a Quadratic Unconstrained Binary Optimization (QUBO) problem, which is further transformed into an Ising Hamiltonian suitable for quantum algorithms.

   The code employs the classical **'NumPyMinimumEigensolver'** to solve the QUBO problem, finding the minimum eigenvalue of the Ising Hamiltonian and extracting the most likely solution, which represents the order of cities and the total distance of the optimal TSP route. This solution is visualized on the graph using the **'draw_tsp_solution'** function.

```
In [30]: def solve_tsp(n):
             start_time = time.time()

             # Generating a graph with n nodes
             tsp = Tsp.create_random_instance(n, seed=123)
             adj_matrix = nx.to_numpy_array(tsp.graph)
             print("distance\n", adj_matrix)
             # Visualizing the graph
             colors = ["r" for node in tsp.graph.nodes]
             pos = [tsp.graph.nodes[node]["pos"] for node in tsp.graph.nodes]
             draw_graph(tsp.graph, colors, pos)

             # Converting TSP to Quadratic Program and QUBO
             qp = tsp.to_quadratic_program()
             #print(qp.prettyprint())
             qp2qubo = QuadraticProgramToQubo()
             qubo = qp2qubo.convert(qp)
             qubitOp, offset = qubo.to_ising()
             #print("Offset:", offset)
             #print("Ising Hamiltonian:")
             #print(str(qubitOp))

             # Solving the QUBO using NumPyMinimumEigensolver (Classical Algorithm)
             ee = NumPyMinimumEigensolver()
             result = ee.compute_minimum_eigenvalue(qubitOp)
             #print("energy:", result.eigenvalue.real)
             #print("tsp objective:", result.eigenvalue.real + offset)
             x = tsp.sample_most_likely(result.eigenstate)
             #print("feasible:", qubo.is_feasible(x))
             z = tsp.interpret(x)
             #print("solution:", z)
             #print("solution objective:", tsp.tsp_value(z, adj_matrix))
             draw_tsp_solution(tsp.graph, z, colors, pos)
```

Figure 6.9: Qiskit code for VQE algorithm.

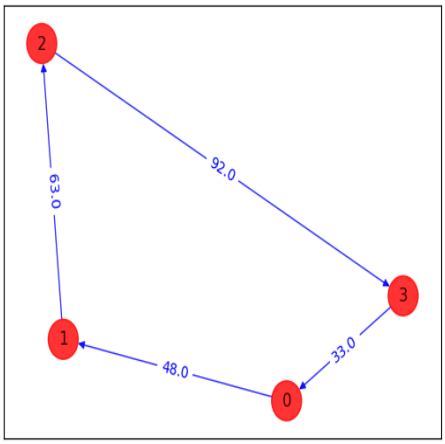
```
    # Using VQE (Variational Quantum Eigensolver) to Solve the Problem(Quantum Algrithm)
    algorithm_globals.random_seed = 123
    seed = 10598
    optimizer = SPSA(maxiter=300)
    ry = TwoLocal(qubitOp.num_qubits, "ry", "cz", reps=5, entanglement="linear")
    vqe = SamplingVQE(sampler=Sampler(), ansatz=ry, optimizer=optimizer)
    result = vqe.compute_minimum_eigenvalue(qubitOp)
    #print("energy:", result.eigenvalue.real)
    print("Time spent by the optimization algorithm:", result.optimizer_time)
    x = tsp.sample_most_likely(result.eigenstate)
    print("feasible:", qubo.is_feasible(x))
    z = tsp.interpret(x)
    #print("solution:", z)
    #print("solution objective:", tsp.tsp_value(z, adj_matrix))
    draw_tsp_solution(tsp.graph, z, colors, pos)
    end_time = time.time()
    print(f"Total time for solving TSP with n={n}: {end_time - start_time} seconds")

# Example usage with n = 4
solve_tsp(4)
```

Figure 6.10: Qiskit code for VQE algorithm.

The VQE is configured with a specific optimizer **('SPSA')** and a variational form **('TwoLocal')**, and the optimal route is visualized similarly. The total time taken to solve the TSP with 'n' cities is measured and printed. The function concludes with an example usage call, **'solve_tsp(4)'**, to solve the TSP for 4 cities.

```
distance
 [[ 0. 48. 91. 33.]
 [48.  0. 63. 71.]
 [91. 63.  0. 92.]
 [33. 71. 92.  0.]]
time: 4080.48877120018
feasible: True
```

(a) This Matrix shows the distance between the cities and the time taken to solve the TSP problem.



(b) This is showing the optimal tour (Shortest tour) for the n=4 cities.

Figure 6.11: Output to solve the TSP using VQE algorithm.

CHAPTER 7

CONCLUSION

7.1 Conclusion

The present work has successfully simulated the Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE) algorithms, as well as tackled the Traveling Salesman Problem (TSP) using both brute force and variational quantum eigensolver (VQE) approaches on the Qiskit platform. The comparison between the brute force and VQE algorithms revealed varying performances in terms of solution quality, computational resources utilized, and scalability. Insights from the simulations of QFT and QPE underscored their accuracy, efficiency, and potential applications in real-world problem-solving. However, as the number of cities increases, the code's performance in solving the TSP declines due to exponential complexity, quantum resource constraints, and algorithmic challenges. Limitations encountered during the VQE simulation included factors such as the qubit requirement, hardware limitations, and the efficiency of the algorithm. Furthermore, simulating quantum circuits on classical computers proved to be computationally expensive and time-consuming due to the exponential growth of quantum states with the number of qubits (beacause if we have n number of cities then n^2 will be the number of qubits and 2^{n^2} quantum states, which will be large in numbers). These challenges highlight the need for further research and optimization strategies to overcome quantum resource constraints and enhance the efficiency of quantum algorithms. Despite these limitations, the findings of this research contribute significantly to advancing our understanding of quantum computing and its potential applications in solving complex optimization problems.

7.2 Future Direction

- After learning the Quantum Fourier transform and Quantum phase estimation we can learn another quantum algorithm such as Shor's algorithm and Grover's algorithm.
- We can familiarize ourselves with quantum hardware development.

- We can learn how to program quantum computers using quantum programming languages like Qiskit or Cirq.
- In determining the most efficient routes for planetary rovers to explore and conduct experiments on the surface of other planets or moons.
- In determining the optimal path for placing components on printed circuit boards to reduce production time.
- For drones in finding efficient flight path planning for surveys, mapping, and deliveries to enhance coverage and reduce energy use.
- Solving the TSP can pave the way for tackling other complex optimization problems like vehicle routing, job scheduling, and supply chain optimization.

REFERENCES

1. **Brigham, E. Oran, and R. E. Morrow** (1967). The fast Fourier transform. IEEE Spectrum **4.12** 63-70.

2. **Mermin, N. David** (2003). From Cbits to Qbits: Teaching computer scientists quantum mechanics. American Journal of Physics **71**, no. 1 23-30.

3. **Hiu Yung Wong** (2023) Introduction to Quantum Computing: From a Layperson to a Programmer in 30 Steps. Springer Nature.

4. **Michael A.Nielsen , and Isaac L. Chuang** (2001). Quantum computation and quantum information. Phys. Today **54.2**.

5. **Dziubelski, Natalia** (2023). Theoretical Foundations of Quantum Computing and the Implementation of the Quantum Fourier Transform.

6. **Camps, Daan, Roel Van Beeumen, and Chao Yang** (2021). Quantum Fourier transform revisited. Numerical Linear Algebra with Applications **28**, no. 1 e2331.

7. **Cochran, William T., James W. Cooley, David L. Favin, Howard D. Helms, Reginald A. Kaenel, William W. Lang, George C. Maling, David E. Nelson, Charles M. Rader, and Peter D. Welch** (1967). What is the fast Fourier transform? Proceedings of the IEEE **55**, no. 10.

8. **Troyer, Matthias** (2020). A Comparison of Quantum and Traditional Fourier Transform Computations." Authorea Preprints.

9. **Musk, Damian R** (2020). A comparison of quantum and traditional Fourier transform computations. Computing in Science and Engineering **22**, no. 6 103-110.

10. **Hales, Lisa, and Sean Hallgren** (2000). An improved quantum Fourier transform algorithm and applications. In Proceedings 41st Annual Symposium on Foundations of Computer Science, pp. 515-525. IEEE.

11. **Weinstein, Yaakov S., M. A. Pravia, E. M. Fortunato, Seth Lloyd, and David G. Cory** (2001). Implementation of the quantum Fourier transform. Physical review letters **86**, no. 9.

12. **Ashdhir, Pragati, Jyoti Arya, and Chaudhary Eksha Rani** (2021). Exploring the fundamentals of the fast Fourier transform technique and its elementary applications in physics. European Journal of Physics **42**, no. 065805.

13. **Beard, James** (2003). The FFT in the 21st century: Eigenspace processing. Springer Science and Business Media.

14. **Cochran, William T., James W. Cooley, David L. Favin, Howard D. Helms, Reginald A. Kaenel, William W. Lang, George C. Maling, David E. Nelson, Charles M. Rader, and Peter D. Welch**. What is the fast Fourier transform? Proceedings of the IEEE **55**, no. 10 1664-1674.

15. **Thomas G. Wong** (2022). Introduction to Classical and Quantum Computing .**Camps, Daan, Roel Van Beeumen, and Chao Yang** (2021). Quantum Fourier transform revisited. Numerical Linear Algebra with Applications **28.1** e2331.

16. **DELLA FISICA ELETTRICA, E.D.S. and SUCCESSIVI**, S., Alma Mater Studiorum· Universita di Bologna.

17. **Mohammadbagherpoor, H., Oh, Y.H., Singh, A., Yu, X. and Rindos**, A.J., 2019. Experimental challenges of implementing quantum phase estimation algorithms on ibm quantum computer. arXiv preprint arXiv:1903.07605.

18. **Mohammadbagherpoor, H., Oh, Y.H., Dreher, P., Singh, A., Yu, X. and Rindos**, A.J., 2019, November. An improved implementation approach for quantum phase estimation on quantum computers. In 2019 IEEE International Conference on Rebooting Computing (ICRC) (pp. 1-9). IEEE.

19. **Matai, R., Singh, S.P. and Mittal, M.L.**, 2010. Traveling salesman problem: an overview of applications, formulations, and solution approaches. Traveling salesman problem, theory and applications, **1(1)**, pp.1-25.

20. **Saiyed, A.R.**, 2012. The traveling salesman problem. Indiana State University, **2**, pp.1-15.

21. **Matsuo, A., Suzuki, Y., Hamamura, I. and Yamashita, S.**, 2023. Enhancing VQE Convergence for Optimization Problems with Problem-Specific Parameterized Quantum Circuits. IEICE TRANSACTIONS on Information and Systems, **106(11)**, pp.1772-1782.

22. **Schnaus, M., Palackal, L., Poggel, B., Runge, X., Ehm, H., Lorenz, J.M. and Mendl, C.B.**, 2024. Efficient Encodings of the Travelling Salesperson Problem for Variational Quantum Algorithms. arXiv preprint arXiv:2404.05448.