Building goal-specific quantum circuits with evolutionary algorithms

Shailendra Bhandari

Thesis submitted for the degree of
Master in Applied Computer and Information Technology: Data Science
60 credits

Department of Computer Science
Faculty of Technology, Art and Design

OSLO METROPOLITAN UNIVERSITY

Spring 2023

Building goal-specific quantum circuits with evolutionary algorithms

Shailendra Bhandari

© 2023 Shailendra Bhandari

Building goal-specific quantum circuits with evolutionary algorithms

http://www.oslomet.no/

Printed: Oslo Metropolitan University

Contents

List of Figures

List of Tables

Abstract

This thesis investigates the potential of bio-inspired evolutionary algorithms for designing quantum circuits that prepare highly entangled quantum states. Entanglement is a crucial quantum property and a valuable resource for quantum computing, necessitating the preparation of highly entangled states for quantum computing with minimal effort. The project addresses two problems, one of the interests in the field of Artificial Intelligence (AI) and the other in the field of Quantum Computing (QC). The first aims to evolve quantum circuits to generate cellular automata (CA) rules, and the second to use evolutionary computing to design highly entangled quantum circuits.

In the first part, an evolutionary algorithm is employed to optimize quantum circuits that generate deterministic and critical stochastic CA rules. Various parameter values, including the number of gates in each circuit and mutation rates, are tested, with the results benchmarked against known CA rules and randomly generated ones (stochastic CA). This research began as part of the ongoing work at NordSTAR, but it was later developed into a master's thesis project to validate the initially studied parameters. The findings have been drafted into a paper, which has already been published [1]. Up to this point, the quantum circuits are evolved successfully using CA rules. The next step is to measure the entanglement of the evolved quantum circuits. The necessity of entanglement lies in its ability to enable quantum computers to perform complex computations and solve problems that are intractable for classical computers. Entanglement forms the foundation of many quantum algorithms and protocols, making it essential to develop methods for preparing highly entangled states efficiently and reliably.

The second part applies an evolutionary algorithm to optimize quantum circuits for entanglement generation. The Meyer-Wallach entanglement measure serves as a fitness function for three-, four-, and five-qubit quantum circuits. The study reveals that an optimal mutation rate achieves a balance between exploration and exploitation while increasing the number of gates in the quantum circuit diminishes its entanglement capability. The research offers insights into the trade-off between circuit complexity and performance, with implications for designing quantum circuits across various quantum computing applications. The findings from the second part have been consolidated into a research paper that was submitted for publication at the IEEE International Conference on Quantum Computing and Engineering (QCE23) [2]. This submission aims to contribute to the expanding body of knowledge in quantum computing and circuit optimiz-

ation by illuminating the factors that impact entanglement generation and circuit performance. Additionally, the overall primary outcome of the thesis will be submitted for publication in the International Journal of Unconventional Computing [3].

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisors, Professors Pedro G. Lind, Sergiy Denysov, and Stefano Nichele, for their constant support and advice throughout this project. They have always been positive and encouraging, and it is difficult to overstate the efforts that made this work thrive. I really appreciate their guidance and the unique opportunity they gave me to work on a quantum computing project with the Artificial Intelligence Group. I am also grateful for the time I spent with everybody I met during this period, as they all contributed in different ways to enrich my studies and life experiences. Thank you for your good advice, help, and enriching ideas.

I am grateful to the AI Lab at Oslomet and NordSTAR for providing me with a fantastic working environment that brings together dedicated professors, employees, and students for meetings and conferences. Thank you for giving me the opportunity to present and attend well-organized and informative international conferences. It was an excellent opportunity to present our paper titled "Evolving quantum circuits to implement stochastic and deterministic cellular automata rules" [1] at the ACRI 2022 conference held at the University of Geneva, Switzerland from 12-15 September 2022. Additionally, attending the IEEE International Conference on Quantum Computing and Engineering (QCE22) in Broomfield, Colorado, USA, was a lovely experience. Thanks to these opportunities, I have many adventures to look back on.

I want to express my heartfelt gratitude to Sebastian Overskott and Ioannis Adamopoulos, my esteemed co-authors of the paper [1]. Their immense hard work, encouragement, support, and guidance throughout the project have been invaluable. I am especially grateful to Sebastian for his remarkable contribution to developing the evolutionary algorithm codes at the beginning of the project while working with the NordSTAR team. I deeply appreciate their unwavering commitment and dedication to this project, which made it a success. Once again, thank you, Sebastian and Ioannis, for your significant contributions to this project.

These acknowledgments would not be complete without thanking my family for their unwavering support throughout this period of my life. I could not have made it this far without the love and support of my brothers, Dr. Janarjan Bhandari, Rameshwor Bhandari, and Saroj

Bhandari. They have always believed in me and have been there for me through thick and thin. My father deserves a special mention for nourishing my curiosity and allowing me to explore the world. His wisdom and guidance have been instrumental in shaping me into the person I am today. I am also grateful to my mother for her constant love and unwavering faith in me. Her encouragement has been a source of strength for me throughout my journey.

I want to express my sincere gratitude to the Oslo Metropolitan University and the state of Norway for allowing me to pursue free and quality education. As an international student, I feel privileged to have access to the excellent facilities and resources the University offers. Without this opportunity, I could not have written this thesis and accomplished what I have today.

I want to thank everyone who has contributed to my journey in one way or another. Your support, guidance, and encouragement have been invaluable to me, and I am deeply grateful for everything.

Chapter 1

Introduction and motivation

> ***If you think you understand quantum mechanics, you don't understand quantum mechanics.***
>
> RICHARD P. FEYNMAN

Quantum computing (QC) is a rapidly evolving field of computer science focused on the development of computer-based technologies based on quantum theory [4]. Quantum theory is a fundamental theory in physics that describes the behavior of matter and energy at the atomic and subatomic levels. Despite being widely used, quantum theory's complex behavior is difficult to comprehend, and researchers are continuously uncovering its secrets. However, by leveraging the principles of quantum mechanics, certain computations that were once considered impossible can now be performed. Quantum computing is an advanced computing paradigm that uses quantum bits, also known as qubits, to perform computation [4]. Unlike classical bits, which can only exist in one of two states, 0 or 1, qubits can exist in multiple states simultaneously, a phenomenon known as superposition. This property enables quantum computers to perform certain computations faster and more efficiently than classical computers. The unique features of quantum computing, such as superposition and entanglement, have the potential to enable faster and more efficient computations than classical computing. The practical applications of quantum computing are numerous, and its potential impact on various industries is significant. For example, Shor's algorithm [5], a quantum algorithm, can factorize large numbers exponentially faster than the most efficient classical algorithm currently available. This advancement has major implications for cryptography. Similarly, quantum machine learning algorithms, such as quantum support vector machines, can perform certain machine learning tasks exponentially faster than classical machine learning algorithms [6].

The intersection of quantum computing and artificial intelligence (AI) is an exciting area of research, with both technologies considered disruptive technologies that will dramatically affect the way consumers, industries, and businesses operate in the near future. Until recently, the two fields were considered distinct, but recent research has shown that they can help each other solve problems and remove barriers to a new technological reality. One of the challenges in quantum computing is the preparation of highly entangled states with minimal effort. This task can be delegated to AI, which has the potential to improve the preparation protocols for entangled states significantly.

The study of quantum computing began in the 1980s, with the goal of solving complex computational problems more efficiently than classical algorithms [7]. Quantum computing uses qubits, which can exist in multiple states simultaneously, to perform computations. The principles of superposition and entanglement are essential to quantum computing and have led to the development of quantum machine learning algorithms and evolutionary computation using evolutionary algorithms. Quantum machine learning has emerged as a promising research area in the field of quantum computing [8, 9]. It uses quantum algorithms to train complex models for classification, regression, and clustering [9]. A breakthrough example of this technology was demonstrated in the research of Tacchino et al. [10], which presented the first implementation of an artificial neuron on a real quantum computer. In this work, a quantum information-based algorithm was used to implement the quantum computer version, and a small version of the model was experimentally tested on a real processor with the expected results. In addition to quantum machine learning, evolutionary computation using evolutionary algorithms is another area of interest in the field of quantum computing. These algorithms operate on a population of proposed solutions to an optimization problem, selecting an appropriate subset of solutions and evolving these individuals using evolutionary operators. Recently, a hybrid classical/quantum genetic architecture has been developed to design a genetic algorithm with some key evolutionary steps (evolutionary hybrid algorithm) and run it on actual quantum computers [11]. This approach has the potential to accelerate the optimization process, thereby improving the overall performance of the evolutionary algorithm. These emerging technologies are expected to impact various industries and scientific fields significantly, and their potential applications are being actively explored. Further research in these areas is expected to drive the development of more efficient and effective computing technologies with potential applications in various industries and scientific fields.

The use of evolutionary algorithms in quantum computing offers an efficient way to optimize quantum circuits for specific applications. The implementation of such an algorithm allows for the flexible design and evolution of quantum circuits using a range of gates and fitness functions. The main objective of this work is to review and study recent advancements in evolutionary quantum computation for designing quantum circuits with cellular automata and addressing

the problem of maximal entanglement. The motivation behind using evolutionary algorithms for quantum circuit optimization is twofold. Firstly, it enables building quantum circuits using various cellular automata rules with specific features. Secondly, the evolutionary approach of Evolutionary Quantum Computing (EQC) can be used to design and build quantum circuits for maximal entanglement.

The interdisciplinary of this project brings together two cutting-edge fields, Quantum Computing and Artificial Intelligence (AI), enabling the development of a unique professional skill set. Entanglement, a striking type of quantum correlation, plays a vital role in quantum computations and is considered to be the most valuable resource in this field. The preparation of a set of qubits in a highly entangled state is crucial for quantum computations, as it provides the necessary fuel for the computation process. However, the task of preparing such states using quantum gates, the building blocks of quantum circuits, is highly complex and demands a profound understanding of various aspects of quantum dynamics.

While bio-inspired evolutionary algorithms have proven proficient in solving practical classical problems such as pattern recognition, traffic control, and big data sorting [12–14], their potential in quantum computing and quantum machine learning is still being explored [15–19]. Therefore, this project aims to investigate the possibility of using bio-inspired evolutionary algorithms to retrieve a highly entangled quantum state from a given number of qubits, providing insights into the potential integration of bio-inspired evolutionary algorithms in quantum computing.

We will also discuss the key concepts of quantum entanglement and evolutionary algorithms and how they are combined to develop EQC-based evolutionary algorithms for entanglement measurement. A comparative analysis of various EQC-based algorithms for entanglement measurement will also be presented, along with their strengths and limitations. Finally, we will discuss the future prospects of evolutionary algorithms in entanglement measurement and other quantum information processing applications. The project is a collaboration between the OsloMet AI Lab and its sub-divisions, including the Living Technology Lab, the Quantum Computing Lab, and the Stochastic Lab. It will provide valuable insights into the potential of bio-inspired evolutionary algorithms in quantum computing, contributing to the growing field of Quantum AI.

The thesis is structured as follows. This Chapter 1 presents the motivation and introduction to the topic, providing an overview of the goals and objectives of the research. Chapter 2 offers an in-depth background and state-of-the-art review, discussing the relevant theories and experimental considerations for quantum computing and evolutionary algorithms. Chapter 3 outlines the methodology and research plan, describing the proposed approach and steps to

achieve the research objectives. Chapter 4 presents the results of evolving quantum circuits using CA rules, which are analyzed and discussed. Similarly, Chapter 5 presents the results of the evolution of quantum circuits to achieve a maximally entangled circuit. Chapter 6 presents the summary and the outlook of the thesis. Finally, there are three appendices. Appendix A presents each run's results and the circuit's visualization as displayed in the console. Tables are provided for the best fitness scores of Rule 90 and Rule 110 and three randomly generated sets of CA probabilities. Appendix B details the implementation method of the Meyer-Wallach entanglement measure, including entanglement measures for parametrized two-qubit circuits and a three-qubit random circuit generated with a random state vector. Appendix C discusses the implementation method for von Neumann entanglement entropy for two- and three-qubit quantum circuits. Together, these chapters and appendices offer a comprehensive overview of the research, the results obtained, and the implications of this work for the field of quantum computing and evolutionary algorithms.

Chapter 2

Background and state of the art

Quantum computing is a little like fusion power. It's the technology of the future. It always has been and it probably always will be.

SAMUEL ARBESMAN

2.1 Quantum computing: basic concepts

Quantum computing is a novel computational paradigm that exploits quantum mechanics phenomena such as superposition and entanglement to achieve intrinsic and enormous parallelism in computation. This intrinsic parallelism makes quantum algorithms potentially more efficient than classical ones. Qubits are the basic unit for storing and processing information in quantum computers. Unlike classical bits that have a single binary state, a qubit can be in a quantum state that is a complex linear superposition of $|0\rangle$ and $|1\rangle$ before being measured. In a sense, before performing a quantum measurement, a qubit may simultaneously have the values 0 and 1 and, only when it is measured, it "collapses" to one of these two values, corresponding to classical bits. While qubits share some similarities with classical bits, they can assume other states besides $|0\rangle$ and $|1\rangle$ and can form linear combinations of states, known as superpositions [20]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle\,. \tag{2.1}$$

The quantum state of a qubit can be represented by the state vector in Equation 2.1. Here, α and β are complex coefficients that satisfy the normalization condition: $|\alpha|^2 + |\beta|^2 = 1$. The coefficients, α, and β are referred to as probability amplitudes since the probability of

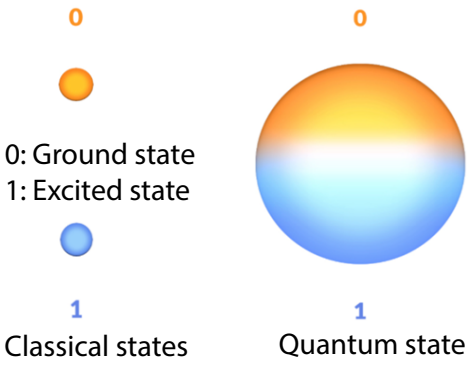


Figure 2.1. The basic unit of information in a QC is the quantum bit (qubit) which can be in any linear combination of ground and excited states. Figure adapted from Ref. [21].

measuring the qubit to be in state $|0\rangle$ is $|\alpha|^2$. Similarly, the probability of measuring the qubit to be in state $|1\rangle$ is $|\beta|^2$. The complex nature of quantum mechanics allows for a qubit to exist in a coherent superposition of the two states, which enables the exploitation of parallelism in quantum computing [20].

The classical bit determines the state 0 or 1 (left side of Figure 2.1) and the classical computers do this all the time when they retrieve the contents of the memory. However, the quantum state of a qubit is in a continuous state between 0 and 1 as shown in the right side of Figure 2.1. We can measure the probability of the state $|0\rangle$ with probability $|\alpha|^2$ or the state $|1\rangle$ with probability $|\beta|^2$, where $|\alpha|^2 + |\beta|^2 = 1$. With vectors, we can describe the more complex states than just $|0\rangle$ and $|1\rangle$. Let us consider a vector:

$$|q_0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}. \tag{2.2}$$

To understand this matrix state, we need to understand vector addition and vector multiplication. In quantum mechanics, the states $|0\rangle$ and $|1\rangle$ form an orthogonal basis for qubits, which allows us to represent a 2D vector as a linear combination of these basis states [22]. Therefore, we can express the state of the qubit in a different form:

$$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle. \tag{2.3}$$

In this representation, the vector $|q_0\rangle$ is called the qubit's state vector, which conveys all the

information we could know about the qubit. From the above expression, we can conclude that this particular state vector is not entirely $|0\rangle$ and not entirely $|1\rangle$. Quantum mechanically, we typically describe such linear combinations as superpositions [22].

The concept of superposition lies at the heart of quantum mechanics and is one of the key differences between classical and quantum systems. Superposition allows quantum systems to exist in multiple states simultaneously, which can result in powerful computational capabilities when leveraged in quantum computing [22].

Alternatively, the qubit can be represented by enabling its visualization in a three-dimensional reference system. This representation is known as the Bloch sphere (named after the Swiss-American physicist Felix Bloch), and it uses the following representation of a qubit to work, derived from a polar form of a complex number [23]:

$$cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{2.4}$$

where polar angle θ and azimuthal angle ϕ are two angles that uniquely identify a qubit. They are used to represent the state of the qubit as an arrow from the origin on the surface of a three-dimensional sphere ($\mathcal{R}^3$) of radius 1. It is shown in Figure 2.2.

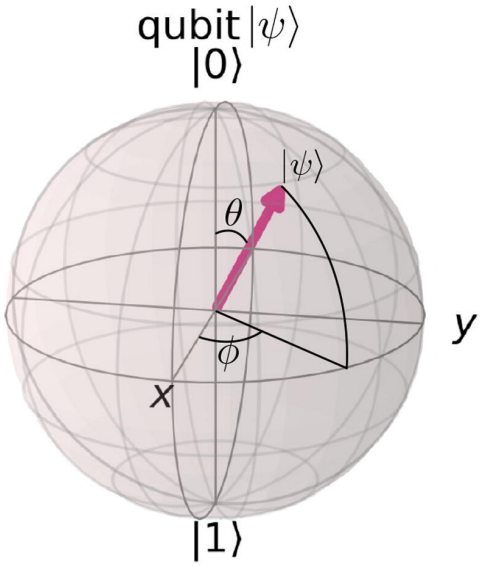


Figure 2.2. The Bloch Sphere representation of a qubit. The sphere illustrates the state of a qubit, with the north and south poles typically representing the two basis states $|0\rangle$ and $|1\rangle$, respectively. Figure taken from Ref. [24].

The evolution of a closed quantum system is described by a special linear operator known as unitary operator U, which operates on the quantum qubits as follows:

$$U|\Psi\rangle = U[\alpha|o\rangle + \beta|1\rangle] = \alpha U|0\rangle + \beta U|1\rangle\,. \tag{2.5}$$

A fundamental aspect of this manipulation of quantum bits or qubits is the use of unitary operators and quantum gates, which enable the precise control of qubit states. Unitary operators perform rotations on the vectors representing quantum states, while quantum gates are the basic building blocks for constructing quantum circuits [22]. Common quantum gates used in quantum computing are the Hadamard gate, Pauli gate, phase gate, and CNOT gate. The Hadamard gate maps the state $|0\rangle$ to the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and maps the state $|1\rangle$ to the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The Pauli gates consist of the Pauli-X, Pauli-Y, and Pauli-Z gates. The phase gates are represented by a diagonal matrix that rotates the qubit phase, such as the S gate and the T gate. The CNOT gate is a two-qubit gate that flips the second qubit's value when the first qubit is in state $|1\rangle$. These gates can be combined to create complex quantum circuits capable of performing quantum algorithms [20, 21].

2.2 Quantum gates

Quantum gates are essential elements of quantum circuits used in quantum algorithms. These gates operate on qubits to perform various tasks in quantum computing, such as quantum teleportation and Shor's algorithm [5]. Quantum gates are designed to take advantage of the two key features of quantum computation, superposition and entanglement. Superposition allows the qubit to be in multiple states simultaneously, while entanglement describes a correlation between two or more qubits such that their states are dependent on each other. These two features allow quantum gates to process and manipulate multiple states at once, which leads to quantum computation being much faster than classical computation in certain tasks. Moreover, unlike classical gates, quantum gates are reversible. This is due to the fact that quantum states are reversible, and the underlying physics of quantum computing is unitary. This property of quantum gates is essential in preserving the information contained in the quantum state, allowing quantum algorithms to run without losing information [20].

2.2.1 The quantum X gate

The Pauli X gate is a single-qubit gate in quantum computing, which is commonly known as the quantum NOT gate. It flips between the state $|0\rangle$ and $|1\rangle$, where σ_x is the Pauli X matrix. The Pauli X matrix can be written as

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \tag{2.6}$$

The Pauli X matrix has the following property:

$$\sigma_x \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \quad \text{and} \quad \sigma_x \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle . \tag{2.7}$$

The X gate is a crucial component in many quantum algorithms. For instance, in the famous Deutsch-Jozsa algorithm [25], the X gate is used to perform a classical NOT operation on the output qubit. Similarly, the X gate is often used to manipulate quantum states in quantum error correction codes, where it acts as a logical bit-flip operator [5, 26]. In summary, the Pauli X gate is a fundamental single-qubit gate in quantum computing. It flips between the states $|0\rangle$ and $|1\rangle$ and acts as a logical NOT gate. The X gate also plays a significant role in many quantum algorithms and quantum error correction codes.

2.2.2 The quantum Y gate

The Y matrix and the Y gate are shown below respectively. The matrix is also known as the Pauli Y matrix as shown in Equation (2.8).

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} . \tag{2.8}$$

It rotates the qubit state by π around the y-axis on the Bloch sphere. It swaps $|0\rangle$ and $|1\rangle$ and so it is a bit flip, namely $\sigma_y |0\rangle = -i |1\rangle$, $\sigma_y |1\rangle = -i |0\rangle$.

2.2.3 The quantum Z gate

The Z gate is one of the most important single-qubit gates in quantum computing and is often used as a phase gate, adding a phase shift to the qubit state, namely $\sigma_y |0\rangle = |0\rangle$, $\sigma_y |1\rangle = -|1\rangle$. The Z gate applies a rotation of π radians around the z-axis of the Bloch sphere. It changes the sign of the qubit state $|1\rangle$ while leaving the qubit state $|0\rangle$ unchanged.

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.9}$$

The Z gate is often used in quantum algorithms to create superpositions of states with different phases. For example, in the quantum Fourier transform (QFT), a series of Hadamard and Z gates are used to create a superposition of states with different phases, which is then used to perform the Fourier transform on a quantum state [20].

2.2.4 Pauli rotation gates

Pauli rotations can be derived from the aforementioned Pauli gates. They are formulated as exponentiated Pauli gates in the following ways:

$$\begin{aligned} R_x(\theta) = e^{-i\theta\sigma_x/2} &= \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}\sigma_x = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \\ R_y(\theta) = e^{-i\theta\sigma_y/2} &= \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}\sigma_y = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \\ R_z(\theta) = e^{-i\theta\sigma_z/2} &= \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}\sigma_z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \end{aligned} \tag{2.10}$$

The action of a gate $R_j(\theta)$ on a state, for $j \in \{x, y, z\}$, is to rotate the j-axis on the Bloch sphere, for an amount of θ radians.

2.2.5 The quantum H gate

The Hadamard gate is indeed one of the most widely used gates in quantum computing, with a wide range of applications, from quantum state preparation to quantum algorithms [27]. The gate, also known as the H gate, is often the first gate applied in a circuit as it prepares the qubit in a superposition state. In terms of the Bloch sphere, the H gate rotates the qubit by π around an axis that is the average of the x and z-axes. The Hadamard gate is especially useful when working with quantum algorithms like the quantum Fourier transform, which creates a superposition of states with different phases. The Hadamard gate or the H gate has the matrix **H** as follows,

$$\mathbf{H} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . \tag{2.11}$$

By matrix multiplication, it is written as,

$$H|0\rangle = \frac{\sqrt{2}}{2}(|0\rangle + |1\rangle) = |+\rangle \quad \text{and} \quad H|1\rangle = \frac{\sqrt{2}}{2}(|0\rangle - |1\rangle) = |-\rangle . \tag{2.12}$$

Also, by linearity,

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

and

$$H|-\rangle = |1\rangle .$$

The Hadamard gate, as well as other quantum gates, has the property that it can invert an operation by applying the gate twice. In other words, applying the Hadamard gate twice will undo the effect of the first Hadamard gate. This property is not unique to the Hadamard gate and is also true for other quantum gates.

2.2.6 The quantum CNOT gate

The CNOT stands for the controlled-not gate and is one of the most important gates in quantum computing. It is a multi-qubit gate as it has two-qubit inputs and two outputs. The first qubit is known as the control qubit, and the second is known as the target qubit. If the control qubit is 1, the target qubits state will be flipped from 1 to 0 or vice versa. Since the quantum gates are reversible, we must have the same number of inputs as outputs. We call the qubits q_1 and q_2 and their states $|\psi_1\rangle$ and $|\psi_2\rangle$. The CNOT gates work in the following way: it takes two inputs $|\psi_1\rangle$ and $|\psi_2\rangle$.

- If $|\psi_1\rangle$ is $|1\rangle$, then the state of q_1 remains $|\psi_1\rangle$ but $|\psi_2\rangle$ becomes $X|\psi_1\rangle$.

- Otherwise the states of q_1 and q_2 are not changed.

The CNOT gate is a fundamental building block in quantum computing, and it has many applications in quantum algorithms, quantum error correction, and quantum communication protocols. In quantum algorithms, the CNOT gate is used to perform entanglement and to create superpositions of quantum states, which are essential for many quantum algorithms. For example, the CNOT gate is used in the famous Shor's algorithm to create an entangled state between two quantum registers, which is used to perform the Fourier transform on a quantum state [5]. The CNOT gate also plays a crucial role in quantum error correction codes, where it is used to encode and decode quantum information in a way that can detect and correct errors in the quantum state. For example, the Steane code, which is a seven-qubit quantum error correction code, uses four CNOT gates to encode a logical qubit in a seven-qubit state [26]. In quantum communication protocols, the CNOT gate is used to perform quantum teleportation, which is a process that allows the transmission of a quantum state from one location to another without physically moving the quantum state [28]. The matrix and the visual representation of the CNOT gate are given by:

$$C_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

It is a permutation matrix that swaps the third and fourth coefficients of $|\psi_1\rangle \bigotimes 2 |\psi_2\rangle$. Here are a few examples of how the CNOT gate can be used to manipulate two-qubit states:

- Entangled States: One of the most important applications of the CNOT gate is in creating entangled states [4]. For example, applying a CNOT gate to a pair of qubits in the state $|00\rangle$ or $|11\rangle$ creates an entangled state $|00\rangle + |11\rangle$ while applying a CNOT gate to a pair of qubits in the state $|01\rangle$ or $|10\rangle$ creates an entangled state $|01\rangle + |10\rangle$.

- Quantum Teleportation: The CNOT gate is also used in quantum teleportation, a quantum communication protocol that allows a quantum state to be transmitted from one location to another without physically sending the qubits. In this protocol, a pair of qubits are entangled using a CNOT gate and then separated. When a third qubit is introduced at a different location, a combination of CNOT and Hadamard gates can be used to "teleport" the state of the third qubit to the entangled pair [29].

2.2.7 The quantum Toffoli gate

The Toffoli gate is a quantum gate used in quantum computing that performs a Boolean function on three qubits, also known as a controlled-NOT gate. It is named after its inventor Tommaso Toffoli. The gate only applies a NOT operation to the target qubit when both control qubits are in the state $|1\rangle$. The Toffoli gate is universal, which means that any classical or quantum computation can be decomposed into a sequence of Toffoli gates along with some one-qubit gates [22].

The mathematical expression and the visual representation of the Toffoli Gate are:

$$|x\rangle\,|y\rangle\,|z\rangle \rightarrow |x\rangle\,|y\rangle\,|z \oplus xy\rangle\,, \tag{2.14}$$

where x, y, and z are the three qubits, $\oplus$ is addition modulo 2, and $|0\rangle$ and $|1\rangle$ are the standard basis states.

The Toffoli gate is an essential gate for classical computer operations and is used in constructing many quantum algorithms, such as Shor's algorithm for integer factorization and Grover's algorithm for unstructured search [30].

2.3 Quantum circuits

A sequence of gates applied to one or more qubits is the definition of a quantum circuit. A quantum circuit involves a collection of one or more qubits that are initialized to state $|0\rangle$. Figure 2.3 shows a simple example of a quantum circuit that can achieve an entangled state between two qubits. The circuit is known as the Bell state circuit because it produces one of the four maximally Bell entangled states (Bell pair[1]), which can be mathematically expressed as it is shown in equation (2.15), from the standard basis kets $|00\rangle$, $|01\rangle$, $|10\rangle$and $|11\rangle$. As we can see, both qubits have been prepared to have an initial state $|0\rangle$. A Hadamard gate has been applied on the first qubit, which brought it in a superposition of the values $|0\rangle$ and $|1\rangle$, and then a CNOT gate brought both qubits into the first Bell entangled state. An interesting fact about entanglement is that the qubits involved remain entangled no matter how far apart we may move them. Therefore, we can easily use Bell pairs to generate correlated random bits at different locations. The four Bell states are denoted by the symbols $|\phi^+\rangle$, $|\phi^-\rangle$, $|\psi^+\rangle$, and $|\psi^-\rangle$ and they are defined as follows:

[1] the state used by John Bell in his demonstration of the inexplicable correlations of entangled state

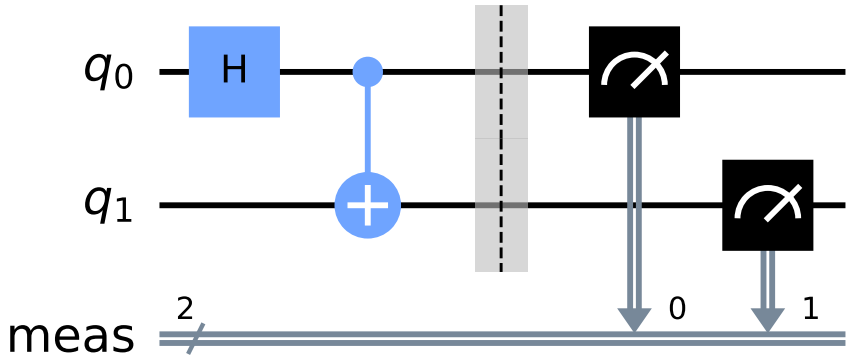


Figure 2.3. Entanglement in the quantum circuit. q_0 and q_1 in the figure are simply labeled as two qubits.

$$
\begin{aligned}
|\Phi^+\rangle &= \frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle \qquad |\Psi^+\rangle = \frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle \\
|\Phi^-\rangle &= \frac{\sqrt{2}}{2}|00\rangle - \frac{\sqrt{2}}{2}|11\rangle \qquad |\Psi^-\rangle = \frac{\sqrt{2}}{2}|01\rangle - \frac{\sqrt{2}}{2}|10\rangle \ .
\end{aligned}
\tag{2.15}
$$

In each Bell state, the two qubits are entangled, which means that their states are linked in a way that classical physics cannot explain. For example, if one qubit is measured and found to be in state $|0\rangle$, the other qubit will be found to be in state $|0\rangle$ or $|1\rangle$ with equal probability, regardless of the distance between them.

Figure 2.4 shows an example of a quantum circuit for performing a multi-qubit (three-qubit) computation. The circuit includes Pauli xyz gate rotations (R_X, R_Y, R_Z), quantum X, Y, Z-gate, a CNOT gate (CX), a Hadamard gate (H), and a Toffoli gate (CCX). The gate which acts on the qubit is represented by a square, and in our case, it is the X gate that flips the state of the qubit from $|0\rangle$ to $|1\rangle$. Each horizontal application of gates for a qubit is called a wire and each wire has a depth which is the number of gates applied to that qubit. In our example, we have one wire with depth 1. The measurement symbol at the end is not considered a gate. The time progress in a quantum circuit goes from the left to the right [27].

2.4 The concept of entanglement in quantum computing

In quantum computing, the concept of entanglement plays a crucial role. Entangled states refer to quantum states that cannot be decomposed into the direct product of two subsystems [31]. When operating on one or more specific qubits, the state of the qubit and the state of the other

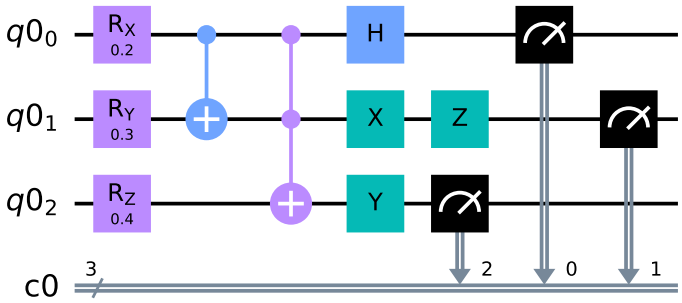


Figure 2.4. Quantum circuit for performing a multi-qubit (three-qubit) computation. The circuit includes Pauli xyz gate rotations (R_X, R_Y, R_Z), quantum X, Y, Z-gate, a CNOT gate (CX), a Hadamard gate (H), and a Toffoli gate (CCX). The qubits are represented by the horizontal lines, and the boxes represent the gates.

entangled qubits change simultaneously. Quantum entanglement, first introduced by Erwin Schrödinger in 1935, is a phenomenon at the quantum level where the quantum states of two or more particles are intrinsically correlated, even when these particles are spatially separated [32]. Qubits are considered entangled if the amplitudes of an n-qubit configuration define a correlation between the individual qubits that cannot be explained classically. If a quantum system is not highly entangled, it can often be simulated efficiently on a classical computer [33, 34].

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.16}$$

Equation (2.16) presents the state $|\psi\rangle$, which is a completely entangled two-qubit state, also known as a Bell state [35]. The normalization factor $\frac{1}{\sqrt{2}}$ ensures that the total probability of the state equals 1. Upon measuring the first qubit, the wave function collapses to one of two possible states, determining the value of the second qubit and demonstrating a correlation between the qubits. In this case, a fully entangled state, such as the one shown in Equation (2.16), can be obtained using a CNOT gate, as illustrated in Figure 2.3. When particles are entangled, the measurement of one particle affects the measurement of the other, exemplifying nonlocality in quantum mechanics. For example, consider the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. If the second qubit has not been measured, the probability of measuring the first qubit as $|0\rangle$ is 1/2. However, if the second qubit has been measured, the probability that the first qubit is measured as $|0\rangle$ is either 1 or 0, depending on whether the second qubit was measured as $|0\rangle$ or $|1\rangle$, respectively. In contrast, the state $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ is not entangled because

$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, which means that any measurement of the first qubit will yield 0 regardless of measurements of the second qubit. Similarly, the second qubit has an equal probability of being measured as 0 or 1, independent of the measurements of the first qubit. Thus, entanglement represents a non-classical correlation between two quantum systems, reflecting the inherent interdependence of their quantum states.

Entanglement is a fundamental property of quantum systems that plays a crucial role in quantum computing. In a two-qubit system, the quantum state can be expressed as a superposition of the four basis states:

$$|\Psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle\,. \tag{2.17}$$

Here, α_{00}, α_{01}, α_{10}, and α_{11} represent the complex amplitudes of the corresponding basis states. For a valid quantum state, the sum of the squares of the absolute values of these amplitudes must be equal to 1:

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{2.18}$$

When measuring both qubits simultaneously, there are four possible outcomes, each with its corresponding probability:

- "00" with probability $|\alpha_{00}|^2$
- "01" with probability $|\alpha_{01}|^2$
- "10" with probability $|\alpha_{10}|^2$
- "11" with probability $|\alpha_{11}|^2$

If we measure only the first qubit, there are two possible outcomes: "0" with probability $|\alpha_{00}|^2 + |\alpha_{10}|^2$ and "1" with probability $|\alpha_{01}|^2 + |\alpha_{11}|^2$. The post-measurement state for each outcome will be a renormalized state, depending on the measured outcome of the first qubit.

For instance, consider the following example state:

$$|\Psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle = (\frac{|0\rangle + |1\rangle}{\sqrt{2}}) \otimes (\frac{|0\rangle + |1\rangle}{\sqrt{2}}). \tag{2.19}$$

This state is a tensor product of two single-qubit states, indicating that the qubits are not entangled. If we measure the first qubit and obtain the outcome "0" with probability $\frac{1}{2}$ and the post-measurement state is given by:

$$|\Psi'\rangle = \frac{\frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle}{\sqrt{(\frac{1}{2})^2 + (\frac{1}{2})^2}} = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle = (\frac{|0\rangle + |1\rangle}{\sqrt{2}}) \otimes |0\rangle\,. \tag{2.20}$$

The post-measurement state remains a tensor product, demonstrating that the qubits are not interacting and exist independently, meaning that the qubits do not feel each other and live separate lives. In contrast, there exist states that are not tensor products, such as Bell states. For example:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.21}$$

This state is indeed a valid quantum state, as it is a superposition of two basis states. Now, let's measure the first qubit. We will have two possible outcomes:

- "0" with probability $\frac{1}{2}$, and the post-measurement state $|\Psi_P\rangle = |00\rangle$
- "1" with probability $\frac{1}{2}$, and the post-measurement state $|\Psi_P\rangle = |11\rangle$

As a result, the consecutive measurement of the second qubit always gives the same result obtained for the first qubit, reflecting the entangled nature of the Bell state [35]. In other words, if the first qubit is measured to be in state "0", the second qubit will also be in state "0" with certainty. Similarly, if the first qubit is measured to be in state "1", the second qubit will also be in state "1".

This strong correlation between the measurement outcomes of the two qubits is a key signature of entanglement [4]. Unlike in the previous example of the separable state, where the qubits did not interact with each other and existed independently, entangled states exhibit nonlocal

correlations that cannot be explained by classical physics. These correlations play a crucial role in quantum computing, enabling phenomena such as quantum teleportation and superdense coding [29, 35].

The exponential growth of possible values for a system of three or more qubits underlies the power of quantum computing [4]. The outcome of entanglement leads to a large number of possible states, making it expensive to run a quantum program in a classical computer simulation. For instance, a 15-qubit quantum computer simulation requires 2^{15} floating-point numbers to store the program state at any instant. As a result, testing and debugging quantum programs in simulation is only feasible for toy-sized programs.

The simplest form of entanglement is known as the Bell state, which enables tasks such as quantum cryptography, super-dense coding, teleportation, and entanglement swapping [36]. Suppose we have two quantum systems, Q_1 and Q_2. The two systems are said to be entangled if the values of certain properties of system Q_1 are correlated with the values that those properties will assume for system Q_2. The Bell states are the simplest form of quantum entanglement used to entangle two qubits. In general, suppose we have two qubits Q_0 and Q_1, where Q_0 is initialized to the Hadamard superposition state, and Q_1 is initialized to the state $|0\rangle$.

$$|q_0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} \quad |q_1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} . \tag{2.22}$$

The relationship between q_0 and q_1 by applying the CNOT gate and considering q_0 as the control bit and q_1 as a target bit.

One of the most well-known examples of the use of entanglement in quantum computing is quantum teleportation [28]. In quantum teleportation, the quantum state of a particle is transmitted from one location to another using entanglement without the need for the actual particle to travel. This process relies on the fact that two entangled particles can share the same quantum state, even when separated by large distances. Therefore, entanglement is a valuable resource in quantum computing and is essential for developing many quantum technologies. Researchers are actively exploring ways to understand better and utilize entanglement to advance the field of quantum computing.

2.4.1 Entangled quantum states and their measurements

In classical physics, a macroscopic physical object can be broken down into pieces that can be described and measured as separate components. However, in the case of an n-particle quantum system, it may not always be possible to describe the state of its component pieces. Specifically,

a pure state ρ_P is considered to be separable if there exists a qubit bipartition X and Y such that

$$\rho_p = \rho^X \otimes \rho^Y \,, \tag{2.23}$$

where ρ^X and ρ^Y are the pure states of A and B, respectively. The pure states of A and B (denoted as ρ^X and ρ^Y) are the quantum states of two subsystems within an n-particle quantum system. The quantum system is divided into two parts, labeled A (or X) and B (or Y). The pure states of A and B are the individual quantum states of these subsystems when they are considered separately. A pure state is entangled if it is not separable. Additionally, a mixed state ρ over n qubits is separable if it can be expressed as a probabilistic mixture of separable pure states with respect to a fixed qubit bipartition X and Y.

$$\rho = \sum_{i}^{N} p_i \rho_i^X \otimes \rho_i^Y \,, \tag{2.24}$$

where ρ_i^X and ρ_i^Y are pure states of X and Y respectively, N is the number of pure states overall qubits in the composition, and the probabilities satisfy $p_i \geq 0$ and $\sum_i p_i = 1$. If the state X and Y are not separable, then the mixed state is entangled [36].

2.4.2 Quantities to assess how entangled is a state of qubits

The descriptors, expressibility, and entangling capability have been used to study the capabilities of the parametrized quantum circuit by quantifying its deviation from random circuits to approach the research question of how much generalization is effective enough in a quantum circuit for a given task [37]. The expressibility of a quantum circuit is the ability to generate pure states that are well representative of the Hilbert space. In a single qubit, the expressibility corresponds to the circuit's ability to explore the Bloch sphere. Sim et al. [37] propose to quantify the ability of the quantum circuit to generate a pure state as a representative of Hilbert space by comparing the true distribution of the fidelities corresponding to the parameterized quantum circuit (PQC) to the distribution of fidelities from the ensemble of Haar random states. They propose to approximate the distribution of the fidelities. These authors define fidelities as $F = |\langle \psi_\theta | \psi_\theta \rangle|^2$, of the PQC. The ensemble of the Haar random state can be calculated analytically as $P_{Haar}(F) = (N-1)(1-F)^{N-2}$, where N is the dimension of the Hilbert space [38]. The measure of expressibility (E) is then calculated by taking the Kullback-Leibner divergence (KLD) between the estimated fidelity distribution and that of the Haar-distributed ensemble. The mathematical equation of this form is written as

$$\mathrm{E} = D_{KL}(\hat{P}_{PQC}(F;\Theta)||P_{Haar}(F))\,. \tag{2.25}$$

A smaller value of the KLD means a better way to explore the Hilbert space. The original definition of expressibility is here deviated by [39] by including these characteristics into the measure itself. The authors evaluated the negative logarithmic of the KL divergence between the ensemble of Haar random states and the estimated fidelity distribution of the PQC so that larger expressibility values correspond to a better ability to explore the Hilbert space, and it is correlated on this logarithmic scale. The equation of expressibility is referred to this as expressibility (E'), distinguished by the ' symbol:

$$\mathrm{E}' = -\log_{10} D_{KL}(\hat{P}_{PQC}(F;\Theta)||P_{Haar}(F))\,. \tag{2.26}$$

As for the second descriptor, we take the name introduced by Sim et al. [37] entangling capability. This descriptor is intended to measure (calculate) the entangling capability of a quantum circuit. In order to calculate entangling capability, the Meyer-Wallach entanglement measure (Meyer and Wallach 2002 [40]) is popular because of its scalability and ease of computation. Mathematically it is defined as

$$Q_{MW}(|\psi\rangle) = \frac{4}{n}\sum_{j=1}^{n} D(l_j(0)|\psi\rangle, l_j(1)|\psi\rangle)\,, \tag{2.27}$$

where $l_j(b)$ represents a linear mapping for a system of n qubits that act on a computational basis with $b_j \in \{0,1\} : l_j(b)|b_1....b_n$ Sim et al. [37] approximated the measure of the PQC by sampling and defined the estimate of the entangling capability of the quantum circuit as follows:

$$Q_s = \frac{1}{|S|}\sum_{\theta_i \in S} Q(|\psi_{\theta_i}\rangle)\,, \tag{2.28}$$

where S is the set of sample circuit parameter vectors θ.

Among the various measures to quantify the entanglement of a state, another essential and commonly used measure is entanglement entropy (EE), which is the amount of information required to describe one part of a bipartite quantum system in a pure state when the other part is traced out [41]. In other words, EE measures how much entanglement is present between two

subsystems. For a pure state, the EE can be calculated by

$$S(\rho^X) = -\mathrm{Tr}(\rho^X \log_2 \rho^X), \tag{2.29}$$

where ρ^X is the reduced density matrix of subsystem X, obtained by tracing out the degrees of freedom of subsystem Y. The logarithm is usually taken to base 2, making the unit of EE as bits. EE can also be used to quantify the entanglement of mixed states by computing the EE of each of the pure-state components of the mixed state and taking their weighted average. Other measures, such as Concurrence, Relative Entropy of Entanglement, and Negativity, are widely used to measure entanglement in a mixed state. These measures also play an important role in quantifying the degree of entanglement in multipartite systems.

2.4.3 Pure state entanglement: The Meyer–Wallach (MW) measure

For more than two systems, most entanglement measures require knowledge of the state itself, which involves performing quantum state tomography. For a pure state of n qubits, Equation 2.27 [42] can be simplified as ,

$$Q(|\psi\rangle) = \frac{4}{n} \sum_{j=1}^{n} D(|\hat{u}^k\rangle, |\hat{v}^k\rangle), \tag{2.30}$$

where $|\hat{u}^k\rangle$ and $|\hat{v}^k\rangle$ denote non-normalized vectors belonging to the complex vector space $\mathbf{C^{2n-2}}$, where n represents the number of qubits in the quantum system. These vectors are obtained by projecting the pure state $|\psi\rangle$ of the n-qubit system onto the local basis states of the k^{th} qubit, where $k \in 1, 2, ..., n$. The non-normalized nature of the vectors is indicated by the hat symbol "ˆ. " Specifically, $|\hat{u}^k\rangle$ and $|\hat{v}^k\rangle$ represent the reduced states of the remaining $(n-1)$ qubits when the k^{th} qubit is in the $|0\rangle$ and $|1\rangle$ states, respectively.

$$|\psi\rangle = |0\rangle_k \otimes |\hat{u}^k\rangle + |1\rangle_k \otimes |\hat{v}^k\rangle . \tag{2.31}$$

The function $D(|\hat{u}^k\rangle, |\hat{v}^k\rangle)$ measures a distance between the two vectors $|\hat{u}^k\rangle$ and $|\hat{v}^k\rangle$. It is obtained by taking the generalized cross-product:

$$D(|\hat{u}^k\rangle, |\hat{v}^k\rangle) = \sum_{i<j} |\hat{u}_i^k \hat{v}_j^k - \hat{u}_j^k \hat{v}_i^k|^2. \tag{2.32}$$

Also, the state $|\psi\rangle$ can be written in the form of Schmidt decomposition over the bipartite

division of the k and the other qubits as:

$$|\psi\rangle = |\bar{0}\rangle_k \otimes |\hat{x}^k\rangle + |\bar{1}\rangle_k \otimes |\hat{y}^k\rangle\,, \tag{2.33}$$

where $\langle \hat{x}^k | \hat{y}^k \rangle = 0$, and $||\bar{0}\rangle_k\rangle\,, |\bar{1}\rangle_k$ are related to $||0\rangle_k\rangle\,, |1\rangle_k$ by a logical unitary operator $\mathcal{U}_k$. The purity of the state of qubit k is therefore $Tr[\rho_k^2] = \langle \hat{x}^k | \hat{x}^k \rangle^2 + \langle \hat{y}^k | \hat{y}^k \rangle^2$. The generalized cross product under logical unitaries, $D(|\hat{u}^k\rangle\,, |\hat{v}^k\rangle) = D(|\hat{x}^k\rangle\,, |\hat{y}^k\rangle)$ in relation to the norm of an anti-symmetric tensor $M_k = |\hat{x}^k\rangle \langle \hat{y}^{*k}| - |\hat{y}^k\rangle \langle \hat{x}^{*k}|$ is written as [42]

$$\begin{aligned} D(|\hat{x}^k\rangle\,, |\hat{y}^k\rangle) &= \sum_{i<j} |\hat{u}_i^k \hat{v}_j^k - \hat{u}_j^k \hat{v}_i^k|^2 \\ &= \frac{1}{2} \sum_{i,j} (M_k^\dagger)_{ij} (M_k^\dagger)_{ji} \\ &= \frac{1}{2} Tr[M_k^\dagger M_k] \\ &= \langle \hat{x}^k | \hat{x}^k \rangle \langle \hat{y}^k | \hat{y}^k \rangle \\ &= \frac{1}{2}(1 - Tr[\rho_k^2]). \end{aligned}$$

Therefore,

$$Q(|\psi\rangle) = 2(1 - \frac{1}{n} \sum_{k=0}^{n-1} Tr[\rho_k^2]). \tag{2.34}$$

This equation of the entanglement measures Q_{MW} simplified the physical meaning of the multi-particle entanglement as an average over the entanglements of each qubit with the rest of the system.

The given equation is an expression for the quantum purity of a state $|\psi\rangle$, denoted by $Q(|\psi\rangle)$. The purity is a measure of how pure or mixed a quantum state is, and it is defined as the trace of the square of the density matrix ρ that represents the state.

Here, n is the number of times the state is measured or prepared, and ρ_k is the density matrix of the state after the kth measurement or preparation. To understand how this equation relates to the purity of the state, we can start by considering the purity of a pure state. If the state $|\psi\rangle$ is pure, then its density matrix is given by $\rho = |\psi\rangle \langle \psi|$, and its purity is:

$$\begin{aligned}\mathrm{Tr}[\rho^2] &= \mathrm{Tr}[(|\psi\rangle\langle\psi|)(|\psi\rangle\langle\psi|)] \\ &= \mathrm{Tr}[|\psi\rangle\langle\psi|\psi\rangle\langle\psi|] \\ &= \mathrm{Tr}[|\psi\rangle\langle\psi|] \\ &= 1\end{aligned}$$

Therefore, for a pure state, the expression $1 - \frac{1}{n}\sum_{k=0}^{n-1} Tr[\rho_k^2]$ is equal to zero.

The density matrix can be written as a convex combination of pure states for a mixed state, $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, where p_i are probabilities and $\sum_i p_i = 1$. The purity of this mixed state is:

$$\begin{aligned}\mathrm{Tr}[\rho^2] &= \mathrm{Tr}\left[\left(\sum_i p_i |\psi_i\rangle\langle\psi_i|\right)\left(\sum_j p_j |\psi_j\rangle\langle\psi_j|\right)\right] \\ &= \sum_i p_i^2 \,\mathrm{Tr}[|\psi_i\rangle\langle\psi_i|] + \sum_{i\neq j} p_i p_j \,\mathrm{Tr}[|\psi_i\rangle\langle\psi_j|\,|\psi_j\rangle\langle\psi_i|] \\ &= \sum_i p_i^2 + \sum_{i\neq j} p_i p_j |\langle\psi_i|\psi_j\rangle|^2 \\ &\leq \sum_i p_i^2 + \sum_{i\neq j} p_i p_j \\ &= \left(\sum_i p_i\right)^2 \\ &= 1\end{aligned}$$

Therefore, the purity of a mixed state is always less than or equal to one. Using this result, we can see that the expression $1 - \frac{1}{n}\sum_{k=0}^{n-1} Tr[\rho_k^2]$ is a measure of how mixed the state is. If the state is pure, then this expression is zero, and if the state is entangled/mixed, then this expression is positive.

In this thesis, we will generate quantum circuits with maximum entanglement using an evolutionary algorithm and we will use the measure in Equation (2.34) as a fitness function for the algorithm. The MW measure of entanglement Q is implemented for a two-qubit handmade circuit using Python. Thus the implemented code satisfies the aforementioned properties of the measure of entanglement (whether the state is pure or entangled). See Chapter 3 and Appendix

B.

2.4.4 Challenges of entanglement in quantum computers

Quantum computers rely on highly entangled states for their effectiveness, as any other state is not useful [43]. However, achieving entanglement is challenging, as it depends on various factors, including the environment. Even if maximally entangled states are prepared, stored, or processed, their entanglement can reduce due to environmental interaction [44]. Qubits undergo decoherence, causing them to lose their initial state after a specific time (coherence time). Additionally, applying quantum gates to a qubit introduces small errors to its state. After applying many gates, the resulting measurements may be unreliable, causing the entanglement applications to lose their purpose. This occurs either because the number of errors increases or the coherence time of qubits is insufficient for maintaining their states after running the entire quantum circuit. Therefore, the complexity of a quantum circuit increases the likelihood of de-coherence and errors [27]. Not all entangled states are equally powerful, so detecting and measuring entanglement quality between qubits is necessary. However, in most quantum systems, quantifying entanglement is challenging, despite the existence of several tools for the purpose [45].

Currently, available quantum technologies are referred to as noisy intermediate-scale quantum (NISQ) technologies due to the presence of noise in every phase of quantum computing, such as qubit preparation, circuit construction, and measurement [46]. To mitigate these errors, quantum error correction (QEC) and fault-tolerant (FT) computations can be employed. However, implementing these methods would require a significant increase in the number of qubits, estimated to be in the millions [47]. Given that only recently a quantum computer with over 100 qubits has been developed, it is unlikely that the necessary number of qubits will be achieved in the near future.

Despite entanglement being considered a key feature of quantum computing, recent studies question its significance [48, 49]. Nevertheless, further research and experimentation are necessary to establish more conclusive evidence.

2.5 Quantum computing in python: the Qiskit library

Qiskit, a Python package provided by IBM, can be used to create and simulate quantum circuits. Qiskit provides tools for coding quantum programs and can simulate quantum circuits on a local machine or run them on a real quantum computer in the cloud. To maintain separation between Qiskit and other applications, creating a Python virtual environment with Jupyter is recommended in Anaconda. A Conda environment can be created with a specific version

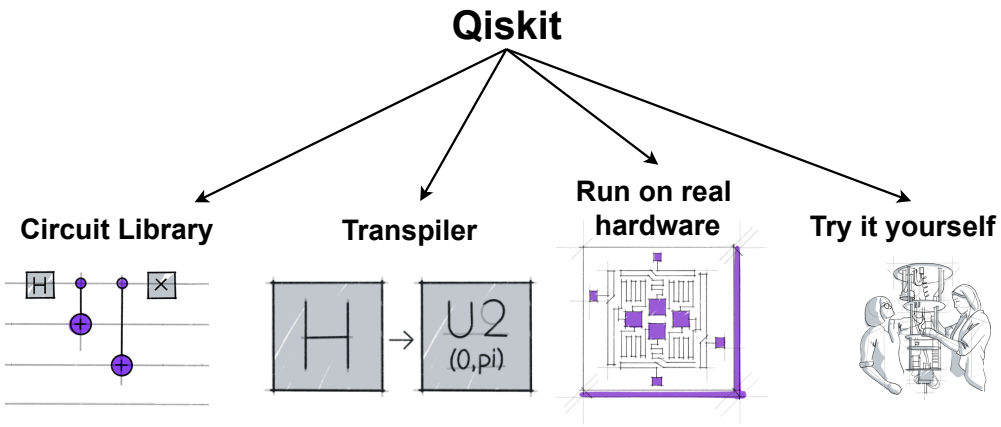


Figure 2.5. Graphical representation showing the various application of the quantum development tool Qiskit. Figure adapted from qiskit.org.

of Python and a set of libraries. To do so, we can first create a minimal environment with only Python by running the command "`conda create -n ENV_NAME python=3`", then activate the new environment with "`conda activate ENV_NAME`" and install the Qiskit package by running the command "`pip install qiskit`".

Qiskit serves as a vital intermediary between quantum algorithms and physical quantum devices. It enables users to work in common programming languages like Python and translates them into a quantum programming language. In the IBM research blog, Qiskit is described as a tool that unites three different levels of users. Firstly, it caters to high-level domain experts in fields such as AI, chemistry, finance, or biology who may not have a background in quantum mechanics. Secondly, it offers developers with some prior knowledge of quantum computing, circuits, and gates an opportunity to experiment with real quantum advantages over classical computing. Finally, it allows quantum mechanics experts to work with control signals both into and out of the qubits and their measurements. Qiskit thus plays a crucial role in bridging the gap between quantum algorithms and physical quantum devices, making quantum computing more accessible to a wider range of users.

Qiskit is a versatile tool for various tasks in quantum computing. The capabilities of Qiskit are illustrated in Figure 2.5, which includes a comprehensive set of quantum gates and pre-built quantum circuits called circuit libraries. Qiskit's transpiler translates codes into optimized quantum circuits using backend gates, enabling users to program any quantum processor. Qiskit can be run on real quantum processors and use the Qiskit run time to compose quantum programs on cloud-based hardware. Additionally, Qiskit allows users to explore the software's capabilities by creating a virtual environment on their local device with a local Python and Anaconda environment or by using the IBM quantum lab and testing there.

2.6 Genetic algorithms and operators: basic concepts

The concept of genetic algorithms was introduced by John Holland in the 1960s, and since then, they have gained significant popularity due to their ability to search large and non-linear spaces [50]. Genetic algorithms are based on the idea of chromosomes as a string of genes, much like the spiral helix model of DNA [51]. In this approach, chromosomes are fixed-length strings of letters or binary numbers, and the optimization process relies on the selection, crossover, and mutation of these genes. In Holland's first implementation of GA, individuals were represented as binary strings, and the fitness of each individual was evaluated to determine which individuals were more likely to survive and produce offspring through recombination and mutation. Offspring were created by recombining parts of the parent's chromosomes, with occasional mutations to introduce new genetic material. This process was repeated over multiple generations until a satisfactory solution was found.

To maintain genetic diversity between generations, it is important to explore a larger space of solutions and avoid getting stuck in local optima. Therefore, ensuring that the next generation is more likely to perform better than the previous one while maintaining genetic diversity is necessary. This can be achieved using different stochastic methods, such as Stochastic Universal Sampling. Genetic algorithms have been extended to many problem domains and have proven to be powerful search algorithms for optimization problems. GA's success lies in its ability to explore large and complex search spaces, including those that are non-linear and non-convex, and to avoid getting trapped in local optima. GA's flexibility has also made it useful in many different applications, such as designing complex engineering systems and predicting protein structures.

In genetic algorithms, there are different parameters that can be adjusted to improve search performance. These parameters include population size, fitness function, crossover rate, mutation rate, and selection method [52].

Population representation

In genetic algorithms, each member of the population (or candidate solution) is represented by a list or string of values. Each value represents a parameter that will be used in the solution. The value could represent the number of iterations of a loop or represent an operator in a sum. Traditionally binary number strings were used to represent a member of the population. This was because of the ease with which binary numbers can be manipulated by computers and the simplicity of carrying out mutations on these binary numbers. To mutate a binary string, a bit can be flipped. This can work in most situations; however, it soon becomes apparent that this cannot use in all situations. If numbers are being represented in binary strings, the most significant bit is at the start of the string; if a bit near this end is flipped, then the number will

change dramatically, whereas if a bit is flipped near the least significant end, then the number will change very little.

The population size, on the other hand, is the number of individuals or chromosomes in the population. A larger population size allows for more diverse solutions to be explored, but it also increases the computation time. On the other hand, a smaller population size may converge faster, but it may also lead to premature convergence and sub-optimal solutions [52].

Fitness function

A GA scheme starts with defining the fitness function (FF). The fitness function(FF) is the name used in GA for the objective function or cost function. It is the target you test your solutions against. It's important to have an appropriate FF to get a good result [53]. The next step is generating many random solutions/individuals called a population. Then we measure each individual in the population against the FF. Then we select a group of individuals based on their performance against the FF. The genes of the individuals in this group undergo genetic operations such as crossover, mutation, and replication. This creates a new population of children based on the genes of the best individuals in their parent generation. Fitness functions are continuous if smaller/larger improvements in an individual's performance are also related to smaller/larger improvements in the fitness [54]. The fitness functions are standardized and normalized, meaning that a fitness value of zero is assigned to the fittest individual, and the fitness value is always between zero and one.

Crossover

In GA, the crossover is a genetic operator that takes two parent solutions and produces offspring by swapping the genetic information between them. The offspring inherits genetic information from both parents, and this is where GA gets its ability to search in multiple directions simultaneously. The crossover operator enhances the diversity of the population and helps the algorithm to converge faster to the optimal solution [51]. It's important to have a good idea of the genetic structure of the solutions. Which clusters of bits are more important? This does not have a general answer [51].

There are several types of crossover methods, including one-point crossover, two-point crossover, and uniform crossover. In the one-point crossover, a single crossover point is selected, and the genetic material from the two parents is exchanged to form two new offspring. Two crossover points are selected in the two-point crossover, and the genetic material between these two points is exchanged to form two new offspring. In the uniform crossover, each gene of the parents is selected with equal probability, and the corresponding genes are exchanged to form the offspring [52].

Crossover is a vital operator in GA and significantly produces better offspring in each generation. The effectiveness of crossover depends on the nature of the problem and the size of the population [55].

Mutation

Both in nature and GA, mutations usually occur less frequently than the crossover. The mutation is, in nature, a random change on a chromosome. In a genetic algorithm scheme, this will be flipping on or several bits. The mutation is a very important operator because its randomness can help the algorithm avoid local extrema. In genetic algorithms, the mutation occurs on a per-gene basis. In the configuration of the genetic algorithm, a value will determine the likelihood of a mutation on a gene within an individual. The mutation is usually applied after or during crossover [55], even though it is possible to mutate the population without crossover. Mutation can vary depending on the precise implementation. The mutation can cause a complete gene change, meaning the gene before the mutation will have no impact on the new gene; you can also have a variance mutation in which the mutation will cause a change in the current gene. A variance mutation on an integer gene would change the value represented by this gene by a set amount concerning the original value.

The mutation rate is the probability that a mutation operation will be applied to a chromosome. The mutation operation involves randomly changing one or more genes in a chromosome and introducing new genetic material to the population. A high mutation rate may increase the diversity of the population, but it may also decrease the quality of the solutions. A low mutation rate may cause the population to converge slowly, but it may also improve the quality of the solutions [56].

Selection method

The selection method determines which individuals in the population are selected to produce offspring for the next generation. Selection methods can be based on fitness, rank, or tournament. Fitness-based selection chooses individuals with high fitness values to produce offspring, while rank-based selection chooses individuals based on their rank in the population. Tournament selection involves randomly selecting a subset of individuals and choosing the best one based on their fitness or rank. The choice of selection method depends on the problem and the population size [51].

2.7 Evolutionary algorithms for quantum computing

Evolutionary computation has long been fascinated by the probability and the possibility of using the concepts of quantum mechanics to enhance the performance of bio-inspired

optimization algorithms. Despite this strong interest in quantum evolutionary computation, there is no breathtaking implementation of a whole evolutionary process, including mutation, crossover, and selection on real quantum processors. Nevertheless, a large community of people is working on quantum-inspired evolutionary algorithms. In this section, most of the approaches in the literature will focus on quantum-inspired evolutionary algorithms. Narayanan and Moore in 1996 [57] published a quantum-inspired evolutionary algorithm where concepts and principles of quantum mechanics are used to inform and inspire more efficient evolutionary computing methods. A quantum-inspired genetic algorithm was used to solve the traveling salesperson problem. Similarly, the pioneering evolutionary algorithms inspired by the concepts and principles of evolutionary quantum computing are proposed in [58, 59]. Both the paper's approaches to the concepts and principles of quantum mechanics inform and inspire more efficient evolutionary computing methods. The improvement on the quantum evolutionary algorithm was made with the quantum swarm evolutionary algorithm [60]. In this paper, a new definition of Q-bit expression called quantum angle was proposed with the employment of an improved particle swarm optimization to update quantum angles automatically. Numerous approaches to the best-known evolutionary algorithms and ideas from quantum computing have been developed. Some of them are the Quantum-behaved Gravitational Search Algorithm (QGSA) [61], Quantum-inspired Immune Evolutionary Algorithm (QIEA) [62], Quantum-inspired Bacterial Foraging Algorithm (QBFA) [63], the Multi-scale Quantum Harmonic Oscillator Algorithm (MQHOA) [64] and its variants [65].

In addition to the quantum-inspired evolutionary algorithms discussed earlier, research has also been carried out on using these algorithms for various optimization problems. One such problem is the maximum clique problem [66], which seeks to find a maximum subset of nodes in an undirected graph. A paper on this problem proposes a hybrid quantum-inspired evolutionary algorithm that uses a binary string of qubits to represent the solution. The algorithm uses a modified version of Grover's search algorithm to calculate each individual's fitness, and it outperforms other algorithms for solving the maximum clique problem.

Another paper proposes a quantum-inspired evolutionary algorithm for the 3-SAT problem [67], which is a well-known NP-hard problem in computer science. The algorithm uses concepts from quantum mechanics, such as superposition, to explore a larger search space and improve the efficiency of the algorithm. The paper's overall goal is to show that quantum-inspired evolutionary algorithms can be used to solve NP-hard problems more efficiently than classical algorithms.

The job shop scheduling problem is another well-known NP-hard problem that researchers have tackled using quantum-inspired evolutionary algorithms [68]. A paper proposes a quantum-inspired approach to the problem and uses a quantum-inspired crossover operator and a

hybrid mutation operator. The algorithm outperformed traditional evolutionary algorithms and demonstrated the potential of quantum-inspired algorithms for real-world optimization problems.

Finally, a paper proposes a quantum-inspired evolutionary algorithm for the multi-objective flow shop scheduling problem, which involves scheduling a set of jobs on a set of machines [69]. The algorithm uses a quantum-inspired initialization scheme, a quantum-inspired crossover operator, and a hybrid mutation operator to outperform traditional evolutionary algorithms.

A hybrid classical/quantum genetic algorithm (HQGA) [11] uses real quantum computers from the IBM Q Experience initiative to evolve the solution to the problems to solve. HQGA introduces completely new operators to define a quantum version of key evolutionary concepts, such as crossover, mutation, and elitism. A 15 qubits real quantum computer was used to prove HQGA converges towards optimal solutions of the problems with a good approximation and with a potential computational advantage with respect to classical evolutionary computation. An adaptive quantum genetic algorithm (QGA) was implemented for optimization problems on the surface of a unit sphere [70]. The algorithm computes the geometric measure of entanglement for multi-qubit pure states. However, these algorithms were implemented on a classical computer as a simulation mentions the possible implementation of the algorithm on a near-term quantum computer. It is possible because the present QGA performs the crossover operation at every iteration on pairs of qubits from an ensemble of qubit pairs.

2.7.1 The overall idea of evolutionary algorithms in quantum computing

Evolutionary Algorithms (EAs) are a type of heuristic algorithm that utilizes the collective learning process of a population of individuals to find a solution for a given problem. The process starts with a diverse population of potential solutions. The best solutions are used to create a new population, which becomes the next generation of solutions. This cycle continues until a satisfactory solution is found. EAs have several key features that distinguish them from other methods. They are population-based, meaning they process a group of potential solutions at the same time. EAs use recombination, which involves combining information from two or more solutions to create a new one. Finally, EAs are stochastic, meaning they use random processes to generate and refine candidate solutions.

The basic EA algorithm can be described using the pseudo-code given in Algorithm 1, which involves initializing a population of random solutions, evaluating their performance, selecting parents, recombining them, mutating the resulting offspring, and evaluating the new candidates. This process repeats until a satisfactory solution is found. The flowchart in Figure 2.6 illustrates the basic steps involved in an EA. The pseudo-code and flowchart are taken and modified

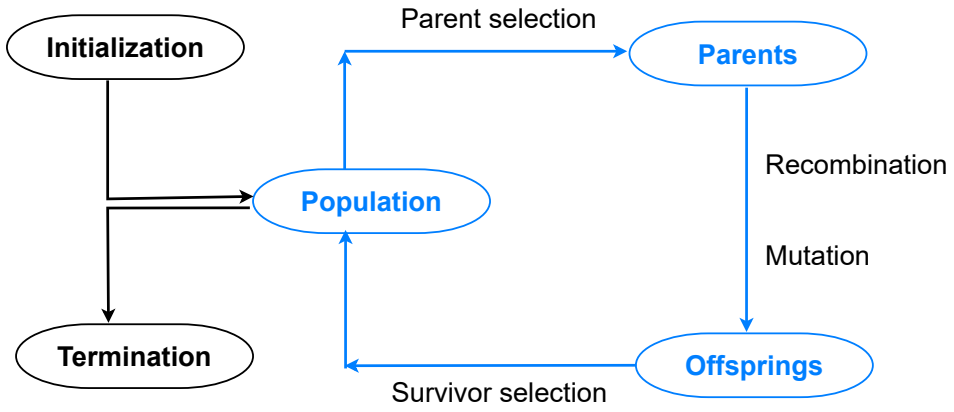


Figure 2.6. The general scheme of an evolutionary algorithm as a flowchart.

from Ref [71]. The algorithm starts with a population of potential solutions, evaluates their performance, and selects the best individuals as parents for the next generation. The parents are then recombined to generate new offspring, which are mutated to introduce variation. The new candidates are then evaluated, and the process repeats until the termination condition is satisfied.

Overall, EAs are a powerful tool for solving optimization problems, and their ability to explore a large search space makes them well-suited for quantum computing applications. Combining the principles of EAs with the capabilities of quantum computers makes it possible to find optimal solutions to complex problems more efficiently than classical algorithms.

Algorithm 1 Basic EA pseudo code scheme

```
Begin
INITIALISE population with random candidate solutions
EVALUATE each candidate
while REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) do
   SELECT parents
   RECOMBINE pairs of parents
   MUTATE the resulting offspring
   EVALUATE new candidates
end whileOD END
```

2.7.2 Cellular automata and quantum cellular automata

The cellular automata (CA) model was first used by John von Neumann [72] showing the possibility of a universal two-dimensional CA. Because of CA's simple, parallel, and homogeneous structure, they are widely used to model physical systems. This, in combination with its use as a mathematical abstraction of parallel computation, makes it a unique combination of physics and theoretical computer science. With the advancement of modern computer technology (quantum computers), there has been a growing interest in the theory of cellular automata by computer scientists and physicists. The articles by Stephen Wolfram [73,

74] Tommaso Toffoli [75] are some of the good starting points to study the modern use of cellular automata. CAs are often used to model physical systems, such as fluid dynamics and crystal growth, and to simulate complex systems, such as social behavior [76] and traffic flow [77].

The study of quantum cellular automata (QCA) is generally attributed to Feynman. A computational scheme based on quantum mechanical laws was first proposed in his famous lecture in 1982 [78]. The general idea is that QCAs are the quantization of classical cellular automata. For example, a computing device based on quantum phenomena would outperform any classical computing device. The elementary components of classical digital circuits in a computer chip are ordinary digital gates. Quantum computers, however, use quantum gates whose output is turned on and off with a certain probability after being measured. The idea of paralleling sequentially operating quantum mechanical machines and mimicking the logical operations of classical digital computation was found in quantum cellular automata. Moreover, the most likely applications of QCA are in the ability to simulate dynamic quantum systems whose dynamics are not computable with classical devices. QCA can be seen as a natural extension of the classical cellular automata model, which has been used to simulate physical and social systems for many years. The key difference is that QCA operates on quantum states, allowing for the potential for massive parallelism and exponential speedup over classical computers. One of the main challenges in building a practical quantum computer is the issue of noise and decoherence. QCA offers a promising approach to mitigating this issue by using the intrinsic robustness of quantum entanglement to perform computations in a fault-tolerant manner.

Cellular or quantum cellular automata are suitable for simulations, physical modeling, and implementation of quantum computers. If one wants to model the general result of the quantum cellular automata in a continuous and isotropic quantum physical phenomenon, then the quantum physical phenomena could be efficiently simulated by a quantum computer. These curious theoretical questions echo the profound questions of axiomatic quantum field theory [79]. Similarly, many investigations of universal quantum cellular automata have facilitated computation, and quantum cellular automata may provide a good paradigm for implementing quantum computers. And that is the reason why quantum cellular automata evolved in the first place [80]. Examples of different applications of cellular automata are presented in [81–85]. One of the earliest breakthroughs of the quantum cellular automata was proof that they could efficiently simulate Turning machines with only constant deceleration [86].

Overall, QCA is a promising area of research with the potential to revolutionize computing and enable the simulation and understanding of complex quantum systems.

2.7.3 Elementary and stochastic cellular automata

Elementary cellular automata (ECA) are a simple yet powerful class of one-dimensional cellular automata first introduced by Stephen Wolfram in the early 1980s [87]. ECAs consist of a grid of cells, each of which can be in one of two possible states, typically labeled 0 and 1. The state of each cell at the next time step is determined by its current state and the states of its neighboring cells, according to a fixed local rule [88]. The local rule is usually defined by a lookup table that specifies the next state of a cell based on the three cell states in its neighborhood: the cell itself and its left and right neighbors. Since each of the three cells can be in one of two states, the neighborhood has $2^3 = 8$ possible combinations of cell states. Consequently, there are $2^8 = 256$ different local rules, each of which can be represented by an 8-bit binary number [89]. Wolfram [87] classified ECAs into four classes based on their behavior. This classification of ECSs has been influential in the study of complex systems, as it highlights the emergence of complex behavior from simple rules. This is a key concept in the field of complexity science [90].

Rule 90 and Rule 110 are two well-known rules in the realm of ECA. Both rules exhibit interesting and distinctive behavior, and their study has contributed significantly to understanding complexity in simple systems [91]. Rule 90 is an interesting ECA rule, which can be represented by the binary number 01011010, or 90 in decimal. The lookup table for Rule 90 is shown in Table 2.1.

Current Pattern	111	110	101	100	011	010	001	000
Next State	0	1	0	1	1	0	1	0

Table 2.1. Lookup table for Rule 90.

Rule 90 generates intricate patterns that display a high degree of symmetry. In fact, it produces a well-known pattern called the Sierpinski triangle, which is a fractal that exhibits self-similarity at different scales. This means that as you zoom in on the pattern, you will continue to see the same triangular structure repeating at smaller and smaller scales. Rule 90 belongs to Wolfram's Class III, which is characterized by chaotic behavior and apparent randomness in the cell states. Despite the seemingly random behavior, Rule 90's patterns have deterministic and highly symmetric properties.

Rule 110 is one of the most famous rules in ECA due to its universal computation capabilities, as proven by Matthew Cook [92]. The rule can be represented by the binary number 01101110, which translates to 110 in decimal. The lookup table for Rule 110 is shown in Table 2.2.

Rule 110 exhibits complex behavior and generates a mix of regular and irregular patterns. It forms localized structures called "particles" that can interact with each other in nontrivial

Current Pattern	111	110	101	100	011	010	001	000
Next State	0	1	1	0	1	1	1	0

Table 2.2. Lookup table for Rule 110.

ways, similar to particles in physical systems. These particles can be used to perform logical operations and transmit information, which is the basis for Rule 110's capability for universal computation [92]. Rule 110 has been proven to be Turing-complete, meaning it can simulate any other cellular automaton or perform any computation that a Turing machine can do, despite its simplicity [92].

In summary, Rule 90 and Rule 110 are two important rules in the study of elementary cellular automata. Rule 90 produces intricate, fractal patterns that exhibit self-similarity and symmetry, while Rule 110 demonstrates the emergence of complex behavior and universal computation from a simple set of rules.

Stochastic cellular automata (SCA) represent a variant of cellular automata that incorporate a probabilistic aspect in the evolution of the system. Contrasting with deterministic cellular automata, where the next state of each cell is uniquely determined by the current state of the cell and its neighbors, SCAs utilize a transition probability matrix to ascertain the next state of each cell based on the current state of the cell and its neighbors [88]. The transition probability matrix in SCAs assigns each potential state transition probability. These probabilities can either be time-invariant or vary over time following a specific rule. Consequently, in SCAs, the state of a cell evolves stochastically, indicating that the state of the cell is determined by a probabilistic function rather than being uniquely determined by the states of its neighbors [93].

SCAs have found applications in various fields to model systems where randomness or noise significantly influences the system's behavior. One such application is presented by Zamith et al. [94], where a novel traffic model efficiently considers different drivers' behavior profiles using a two-stage process involving motion expectation and decision-making. This model implements stochastic rules and employs a Probability Density Function (PDF) of the Beta Distribution, enabling the representation of three distinct driver behaviors by merely adjusting the distribution parameters. This approach is noteworthy for its capacity to model individual drivers' behavior using only the Beta PDF, demonstrating the versatility and applicability of SCAs in capturing the complexities of real-world systems.

2.8 Computational complexity and quantum circuits depth

Computational complexity is a measure of the number of resources, such as time and space, required to solve a computational problem. The complexity of a problem is usually expressed

in terms of the size of the input and is often classified using standard complexity classes, such as P, NP, and NP-hard [95]. The theory of computational complexity categorizes problem difficulty into P (polynomial time) and NP (non-deterministic polynomial time), as well as NP-Hard and NP-Complete, set to describe more intricate problems. These complexity classes provide a framework for understanding the difficulty of computational problems and for exploring the potential advantages of quantum computing.

Evolutionary algorithms (EAs) are a family of optimization techniques inspired by natural evolution, and they are used to solve complex optimization problems. The nature of EAs is problem-dependent and stochastic, meaning that random factors influence their behavior [52]. Due to this stochastic nature, assessing the computational performance of an evolutionary algorithm requires considering the computational effort needed to compute the fitness function or the cost function. This fitness function is a quantitative measure used to evaluate the quality of a candidate solution in the search space.

The finite coherence time of qubits plays a significant role in evaluating the computational performance of quantum algorithms, as qubits tend to change their state spontaneously and are only fully operational for a brief period. Quantum evolutionary concepts leverage quantum superposition and entanglement, exhibiting a logarithmic growth rate in the number of fitness function evaluations needed to identify a highly accurate solution [96]. Additionally, the depth of quantum circuits in this context is constant and does not depend on the problem size [97]. This has a notable impact on the effect of quantum noise on computation. When working with current quantum computers or simulators, the limited number of qubits available necessitates the careful selection of chromosome values in the context of evolutionary algorithms. As future quantum computers with thousands of qubits become available, it will be possible further to explore the complexity of evolutionary algorithms in quantum computing, delving into how chromosomes can affect the computational performance of these algorithms.

The depth of a quantum circuit, which represents the number of layers of quantum gates required for computation, plays a crucial role in determining the circuit's susceptibility to noise and decoherence. Shallow quantum circuits, consisting of fewer layers, are generally preferred over deep ones, as they offer fewer opportunities for errors to accumulate and are more manageable in terms of maintaining coherence. Therefore, in our evolutionary algorithm (EA), it's essential to consider the trade-off between the number of gates and the fitness value proportionately. While higher fitness values might be associated with more quantum gates per circuit, it's important to take into account the complexity of the generated quantum circuit as well. By striving to reduce the depth of a quantum circuit, we can mitigate the effects of noise and decoherence, ultimately improving the reliability of quantum computations. This can be achieved through a combination of clever algorithms, gate decomposition, and optimized gate placement within

the circuit.

The major challenge of quantum computation is the limited number of available qubits. IBM, a leader in the field of quantum computing, achieved a significant milestone in November 2021 by unveiling a 127-qubit quantum processor known as the "Eagle" processor [98]. This breakthrough was made possible through the use of innovative packaging technology, enabling IBM to increase the number of available qubits on a single processor significantly. The Eagle processor followed IBM's previous quantum processors, the 65-qubit "Hummingbird" processor in 2020 and the 27-qubit "Falcon" processor in 2019 [99]. IBM's Eagle processor marks a considerable advancement in quantum computing. The increased qubit count can help researchers and scientists solve more complex problems, potentially leading to significant breakthroughs in materials science, cryptography, and optimization [100]. The new packaging technology employed by IBM to achieve the 127-qubit count involves advanced multi-chip module (MCM) design and cooling techniques, which can help to maintain the delicate quantum states required for computation [98]. This breakthrough demonstrates the potential for continued growth in the number of available qubits in future quantum processors. As the field of quantum computing continues to progress, we can expect further advancements and innovations in the design and capacity of quantum processors, enabling increasingly powerful computational capabilities.

Chapter 3

Methodology and algorithm implementation

> ***A classical computation is like a solo voice - one line of pure tones succeeding each other. A quantum computation is like a symphony - many lines of tones interfering with one another.***
>
> **SETH LLOYD**

In this chapter, we will delve into the step-by-step implementation of the algorithms that have been discussed throughout the thesis. This will include providing an in-depth explanation of each algorithm, along with the rationale behind their design and the specific techniques used to bring them to life. Additionally, we will illustrate how the algorithms interact with each other while the insights into the challenges faced during implementation and the solutions employed to overcome these obstacles are explicitly explained in Section 6.4.

3.1 Overview of the methodology

An evolutionary algorithm based on mutations is often used to generate quantum circuits using quantum gates. In this case, the algorithm optimizes the types of quantum gates and their connectivity. The core of the algorithm is the integer representation, which contains a list of integers and all the functions for generating and mutating the lists. These integers represent the gates and their connections in the circuit. After generating the initial set of quantum chromosomes (which are simply a collection of these lists of integers), the fitness function is used to evaluate each chromosome and identify the current best solution. Once the current

best solution is identified, it can be used to perform several operations. For example, it can be used to reconstruct the quantum states related to the non-best chromosomes, identify the qubits to entangle between the current best chromosome and other chromosomes in the population and compute mutations. These operations enable the implementation of a quantum evolutionary circuit, which is the core component of the evolutionary algorithm. All this information is useful for implementing the quantum evolutionary circuit, the core component of the evolutionary algorithm, aiming to perform genetic evolution on a quantum device, identify a new best current solution, and enable the iteration of the above steps further to improve the quality of the problem's solution.

To address our optimization problem, we evaluated the performance of our proposed approach by running the evolutionary algorithm on a simulator provided by the IBM Q Experience initiative (Qiskit)[1]. However, Qiskit does not offer any means to perform evolutionary algorithms or represent gates in ways other than its own classes. Therefore, we needed to create a string representation of the circuits, the gates, and their connections. To begin, a set of "quantum" chromosomes is assigned. This is the core of the algorithm and the integer representation. It contains a list of integers and all the functions for generating and mutating the lists. After this, fitness function evaluations are performed to identify the current best solution. The fitness function evaluates how well the circuit solves the optimization problem and is typically a function of the circuit's output state. Once the current best solution is identified, it can be used to perform many operations, such as creating a strong correlation between the qubits belonging to the best solution found by the algorithm so far and the remaining quantum chromosomes in the population. This correlation is important for entangling the qubits and improving the performance of the algorithm. Other operations include identifying the qubits to entangle between the current best chromosome and other chromosomes in the population, computing mutations, and so on.

3.2 Main ingredients of implementation

The code implemented for this project was done in Python and resulted in a Python module called QUEVO. The module consists of three classes: `Chromosome`, `Generation` and `Circuit`. The first two classes of the model are part of the genetic algorithm, while the class circuit is part of the Qiskit simulation. The Circuit is handling generating quantum circuits, simulations, and measurements of thus generated circuits.

3.2.1 The chromosome

The `Chromosome` class is the core of the genetic algorithm used for generating quantum circuits. It handles the integer representations of the gates and their connections, as explained

[1] Available at: https://qiskit.org

in Section 3.2.4. The class contains a list of integers and functions for generating and mutating the list. It also handles the initialization of the population and the evolution of the population into a new one. Mutations in the `Chromosome` class can happen in two different ways. The first way is by replacing gates from the pool of gates in the chromosome with a randomly generated new one. The second way is by replacing the chromosome to generate the four best parents. The class is also responsible for checking the chromosome for gates that connect multiple qubits. If the gate has an invalid connection, meaning it is connected to itself through the randomly generated integers, the class randomly generates a valid configuration. Overall, the `Chromosome` class handles the creation of random series, the list of angles needed by some of the gates, mutation of the series, and other list-related functions. It takes a list of the desired gate types as a parameter on construction and automatically creates the tables needed for parsing.

The `Chromosome` class plays a crucial role in the genetic algorithm by providing the genetic makeup of each individual in the population. By generating and mutating chromosomes, the algorithm explores the space of possible quantum circuits in search of an optimal solution. The performance of the algorithm depends on the quality of the mutations and the size of the population, among other factors.

3.2.2 The generation

The `Generation` class is another important class in the QUEVO module, and it is responsible for managing the population of chromosomes. It is a collection of chromosomes that undergo evolution with a specified number of generations. After each evolution step, changes in the chromosomes in the generation occur by selecting a fixed number of chromosomes as elite and allowing the rest of them to evolve further. In each evolution step, the chromosomes are evaluated with the fitness function, and the fittest chromosomes become the parents for the next generation. The rest of the chromosomes are reset for the initial chromosomes. The `Generation` class stores a generation of chromosomes, the fitness associated with each chromosome, methods for running and retrieving fitness for two different fitness functions, and functions for printing. It provides methods for performing selection and mutation to generate a new population of chromosomes. The `Generation` class is essential for the genetic algorithm because it manages the population of chromosomes and implements the selection and mutation operations that drive the evolution of the population. The class also provides methods for evaluating the fitness of the chromosomes, which is a crucial step in determining the parents for the next generation. The algorithm's performance depends on the fitness function's quality and the parameters used in the evolution process, such as the population size, selection method, and mutation rate.

3.2.3 The circuit

The `Circuit` class is responsible for parsing a string and generating a Qiskit quantum circuit, as well as for simulating the circuit, performing measurements, and visualizing the circuit. The class is run on the Qiskit AER simulator and returns the results as a dictionary. The `Circuit` class can be configured to mimic an IBMQ backend using the *aer_sim = Aer.get_backend("aer_simulator")* method. This method configures the simulator to use the user's quantum gates for that backend and the same basis gates and coupling map. Overall, the `Circuit` class is essential for the implementation of the genetic algorithm because it allows the algorithm to generate, simulate, and measure quantum circuits. The class interfaces with Qiskit to create and simulate circuits using the gates specified in the chromosomes. The results of the simulations are used to evaluate the fitness of each chromosome and guide the search for an optimal solution. The performance of the algorithm depends on the quality of the simulations and the accuracy of the measurements. The accuracy of the simulations is affected by the number of qubits in the circuit, the complexity of the gates used, and the noise and other sources of errors associated with the quantum circuits. Therefore, it's important to carefully choose the parameters of the simulations to ensure that the results are reliable and accurate.

3.2.4 Quantum circuit as a series of integers

When solving problems with genetic algorithms, we need to represent the problem as a series of numbers or letters. In the case of designing quantum circuits with the QUEVO module, we need to represent the Qiskit circuit and gate objects numerically. This is achieved by representing each gate as a series of three integers and the circuit as a series of these "triplets." The first integer in the triplet represents the gate type, such as Hadamard or CNOT. The second integer specifies the qubit that the gate targets. The third integer contains information about any control qubits used by the gate. This third integer is ignored when parsing the series of integers for a single-qubit gate. The table that converts an integer to gate type is automatically generated based on the gates specified as a parameter at run-time. This table is used to parse the series of integers into a Qiskit quantum circuit. Representing the circuit as a series of integers is important for the genetic algorithm, as it allows the algorithm to explore the space of possible circuits efficiently. The use of integers also simplifies the implementation of the genetic algorithm, as the chromosome class can handle lists of integers more efficiently than lists of gate objects. Overall, the use of a numerical representation for the circuit and gate objects enables the efficient generation and evolution of quantum circuits using the QUEVO module.

An example: Suppose we want to create a series of integers representing three gates on a three-qubit circuit, with the gate types Hadamard, Pauli X, and CNOT. This would give us a parsing table like Table 3.1. Depending on the circuit we want to design or randomly generate, the

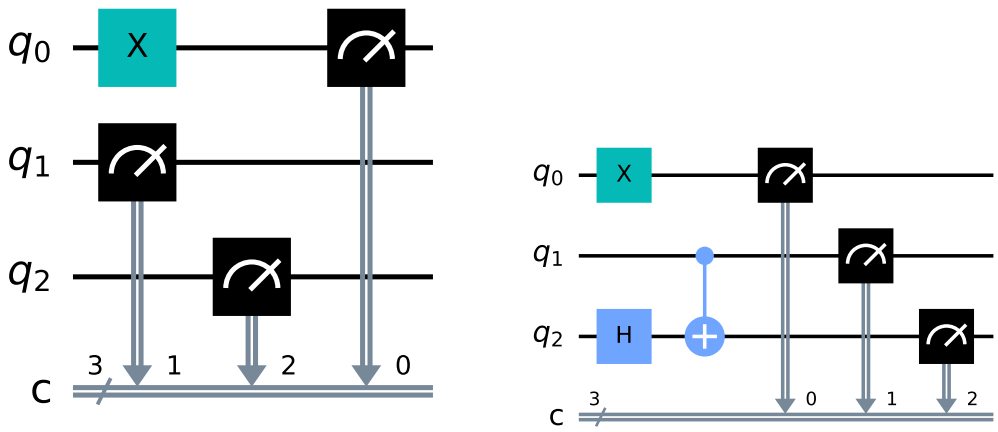


Figure 3.1. (Left) Quantum circuit with three qubits and a X-gate. (Right) Quantum circuit with three qubits, an X-gate, Hadamard-gate, and CNOT-gate.

series of integers might look like this: `[1, 0, 1, 0, 2, 1, 2, 1, 2]`. In this case, the first triplet `1,0,1` represents the first gate. The second triplet `0,2,1` represents gate two, and the third triplet `2,1,2` represents gate three. The parsing table is used to convert the series of integers into a Qiskit quantum circuit, allowing us to visualize and simulate the circuit. The use of integers to represent the gates in the circuit enables the efficient implementation of the genetic algorithm, as the chromosome class can handle lists of integers more efficiently than lists of gate objects.

0	Hadamard
1	Pauli X
2	CNOT

Table 3.1. Translation table for converting integer representations to quantum gates in a three-qubit circuit.

To parse the series of integers into a Qiskit quantum circuit, we use the parsing table to identify the gate types and qubits targeted by each gate. For example, the first triplet of integers in the series `[1, 0, 1]` tells us that the first gate is a Pauli-X gate placed on the 0th qubit (q0). The third integer is ignored since the X gate is a single-qubit gate.

After the first gate is applied, the circuit looks like the one shown in the left panel of Figure 3.1. The second triplet of integers in the series `[0, 2, 1]` tells us that the second gate should be a Hadamard gate placed on the 2nd qubit (q2), with no control qubit. We ignore the last integer in the triplet since the Hadamard gate is a single-qubit gate. The final triplet of integers in the series `[2, 1, 2]` tells us that the circuit contains a CNOT gate with q1 as the target and q2 as the control qubit. The completed circuit is shown in the right panel of Figure 3.1.

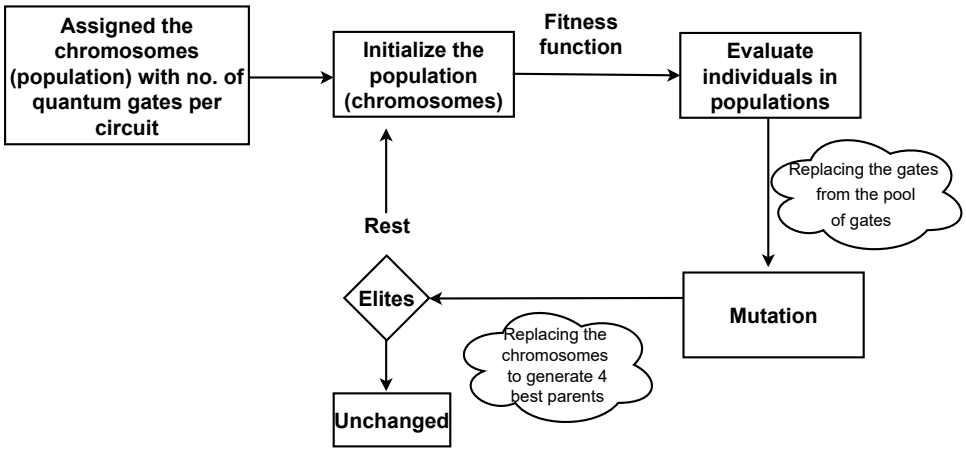


Figure 3.2. Decision tree for the evolutionary algorithm implemented to evolve quantum circuits.

The QUEVO module currently supports several gate types, including Hadamard, Pauli gates (X, Y, and Z), CNOT, Toffoli, swap, RZZ, and RXX. The Toffoli gate, which typically requires four integers to be parsed, can be implemented using the three-qubit setup by assigning the control qubits to the two qubits that are not targeted. The RXX and RZZ gates require an angle as input, which is handled automatically by the module by generating, storing, and updating angles in a separate list.

Overall, the QUEVO module's ability to parse a series of integers into a quantum circuit enables the efficient design and evolution of quantum circuits using the genetic algorithm.

3.3 Genetic algorithms applied to evolve quantum circuits

Figure 3.2 shows the decision tree for the evolutionary algorithm used to evolve the quantum circuit. Initially, a fixed number of chromosomes (population) is assigned with a specified number of quantum gates per circuit, generating the initial set of chromosomes. The assigned population is then initialized, and the genetic algorithm iteratively evolves the current chromosomes into new ones, identifying the chromosome with the highest fitness in each generation as a parent for the next generation.

Each chromosome is evaluated using an appropriate fitness function, and at each evolution step (generation), all chromosomes undergo a probabilistic mutation process. This process involves replacing a gate from the pool of gates in the chromosome with a randomly generated new one or replacing the entire chromosome to generate four new "best" parents. During the mutation process, the four best chromosomes are left unchanged as "elites" while the rest of the chromosomes evolve with a probabilistic selection where each circuit has a probability of becoming a parent that is proportional to its fitness.

In the particular implementation of this evolutionary cycle, we initiated the process with a population of 20 chromosomes, allowing it to continue for a maximum of 500 generations and conducting up to 50 runs. We then focused our efforts on applying this approach to two distinct scenarios: evolving quantum circuits to determine the specific rules of cellular automata and addressing the problems of circuits with maximal entanglement.

Overall, the evolutionary algorithm efficiently optimizes quantum circuits for specific applications, and the QUEVO module's implementation of the algorithm allows for the flexible design and evolution of quantum circuits using a range of gates and fitness functions.

3.3.1 Implementation I: Building quantum cellular automata with specific features

In the first implementation of the QUEVO module, we use the evolutionary algorithm to create quantum circuits for use with cellular automata. Specifically, we create quantum circuits that correspond to the rules of a standard classical one-dimensional cellular automaton (CA) composed of cells with two possible states (0 or 1), which are updated according to one of 256 possible rules matching each of the eight possible neighborhoods ([0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], and [1, 1, 1]) to an update of the middle cell.

We use the evolutionary algorithm to optimize the quantum gate types and their connectivity through mutation to build the quantum circuit associated with each CA rule. We use fitness functions to minimize the difference between the output of the quantum circuit and the target stochastic or deterministic CA for each of the eight possible neighborhoods of an elementary CA.

We investigate different parameter values for the number of gates in each circuit and mutation rates and benchmark our methods with known stochastic and deterministic CA rules as well as randomly generated stochastic rules.

In summary, our implementation demonstrates the effectiveness of the evolutionary algorithm for generating quantum circuits that capture the rules of classical cellular automata, with potential applications in quantum cellular automata and other areas of quantum computing.

3.3.2 Implementation II: Addressing the problem of circuits with maximal entanglement

The second implementation of the QUEVO1 module uses evolutionary algorithms to evolve quantum circuits with high levels of entanglement. This is achieved using the Meyer-

Wallach measure of entanglement as a fitness function, a commonly used measure in quantum information theory. The evolution process involves initializing a population of circuits with a specific number of gates and selecting the circuits with the highest entanglement measure as parents for the next generation. To obtain the best-evolved quantum circuits, genetic operators such as mutation is applied to produce better offspring, i.e., solutions with the highest entanglement measure. Sample code for this measure has been implemented in Python, as shown in Appendix B.

To implement the von Neumann entanglement entropy as a fitness function, modifications have been made to the QUEVO1 module's code to calculate this entropy measure in addition to the Meyer-Wallach measure. The von Neumann entropy is a measure of quantum correlations within a quantum system and can be computed from the eigenvalues of a density matrix. This allows the evolution process to explore the entanglement properties of quantum states represented by state vectors obtained from quantum circuits, providing insights into the complex entanglement structure of quantum systems. The implementation involves generating random density matrices and computing the von Neumann entropy from their eigenvalues, as described in Appendix C.

Collectively, both the Meyer-Wallach measure and the von Neumann entropy can be used as fitness functions to evolve quantum circuits with high levels of entanglement. The implementation involves modifying the QUEVO module's code to calculate these measures and use them to evaluate the entanglement of the evolved quantum circuits.

3.3.3 Mayor Wallach and von Neumann Entanglement Measures as fitness functions

The Implementation II of the thesis explores the use of two entanglement measures, the Mayer-Wallach measure, and the von Neumann entropy, as fitness functions to optimize quantum circuits for maximum entanglement measurement. The *compute_MW _entanglement* method is an implementation of the Mayer-Wallach measure of entanglement within the QUEVO1 module. A state vector is fed as an input in the implementation module, representing the circuit's quantum state. The state vector is then reshaped into a tensor of dimensions 2^n x 2^n, where n is the number of qubits in the circuit. The Mayer-Wallach measure of entanglement is then calculated by computing the reduced density matrix of each qubit. Specifically, for each qubit k, the reduced density matrix is obtained by tracing out all the other qubits from the quantum state. The squared eigenvalues of the reduced density matrix are then computed and summed over all qubits to obtain the entanglement value. The entanglement value is scaled by a factor of $1 - \frac{1}{n}$ to ensure that it lies between 0 and 1. Maximizing the Mayer-Wallach measure of entanglement can lead to the generation of highly entangled quantum circuits. For

more information about the module and instructions on how to use it, please visit the Github repository https://github.com/shailendrabhandari/QUEVO1.

Besides the well-known Mayer-Wallach measure, we also investigate the use of the von Neumann entropy as a fitness function. Our algorithm generates random density matrices using complex state vectors and subsequently calculates the von Neumann entropy from the eigenvalues of the density matrix. The entropy measure is employed as a fitness function to explore the entanglement properties of quantum states, represented by state vectors obtained from quantum circuits. The optimization of quantum states using both Mayer-Wallach and von Neumann entropy measures allows us to measure the entanglement of the quantum circuit generated with the evolutionary algorithm. This comprehensive approach enhances our understanding of the entanglement measure of the quantum circuit using quantum evolutionary algorithms.

3.4 Reduced density matrix: A tool for understanding purity of the state

3.4.1 Density matrix

The density matrix represents a quantum state that can describe both pure and mixed states. While the state-vector representation is only suitable for describing pure states, the density matrix can also be used to represent mixed states, which are probabilistic mixtures of pure states. A mixed state can be expressed as a sum of outer products of pure states, each term weighted by a probability. In the density matrix formalism, a mixed state is represented by a Hermitian matrix with non-negative eigenvalues that sum up to one. The density matrix formalism provides a unified way to describe both pure and mixed states in quantum mechanics. The density operator representation is an alternative way to express pure quantum states using a matrix formalism. It is defined as:

$$\rho \equiv |\psi\rangle\langle\psi|, \tag{3.1}$$

where $|\psi\rangle$ is the state vector that represents the pure quantum state. The expression $|\psi\rangle\langle\psi|$ represents the outer product of the state $|\psi\rangle$ with itself, which results in a matrix. This matrix is called the pure state's density operator or density matrix.

The density operator has several important properties. It is Hermitian, which means that it is equal to its own conjugate transpose. It has a trace equal to one, which reflects the fact that it represents a pure state. It also satisfies the property that $\rho^2 = \rho$, which reflects the fact that the state is pure.

The term $|\psi\rangle\langle\psi|$ in Equation (3.1) represents the outer product of the state ψ with itself:

$$\rho = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} \begin{bmatrix} \alpha_0^* & \alpha_1^* & \dots & \alpha_N^* \end{bmatrix},$$

$$\rho = \begin{bmatrix} |\alpha_0|^2 & \alpha_0\alpha_1^* & \dots & \alpha_0\alpha_N^* \\ \alpha_1\alpha_0^* & |\alpha_1|^2 & \dots & \alpha_1\alpha_N^* \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_N\alpha_0^* & \alpha_N\alpha_1^* & \dots & |\alpha_N|^2 \end{bmatrix}. \tag{3.2}$$

To illustrate this, let us consider an example of a two-qubit, maximally-entangled pure state $|\psi_{AB}\rangle$, which can be expressed as:

$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.3}$$

Using the density operator representation, we can express this state as:

$$\rho_{AB} = |\psi_{AB}\rangle\langle\psi_{AB}|,$$

$$\rho_{AB} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \right), \tag{3.4}$$

$$\rho_{AB} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix satisfies the properties of a density operator; it is Hermitian, positive semi-definite, and has a trace equal to one. The density operator representation provides a compact and convenient way to represent pure states in a matrix formalism, which can be useful for calculations and quantum information processing tasks.

3.4.2 Reduced density matrix

The reduced density matrix ρ_A describes the state of subsystem A after tracing out the degrees of freedom of subsystem B. It is obtained by performing a partial trace of the density matrix ρ_{AB} over subsystem B:

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}) = \sum_{i=1}^{d_B} \langle i|\rho_{AB}|i\rangle, \tag{3.5}$$

where d_B is the dimension of the Hilbert space of subsystem B, and $|i\rangle$ denotes a basis state of subsystem B.

The reduced density matrix ρ_A inherits some of the properties of the original density matrix ρ_{AB}. For example, it is Hermitian, positive, semi-definite, and has a trace equal to one. However, it is generally not pure, even if the original state ρ_{AB} is pure. This is because entanglement between subsystems cannot be removed by taking a partial trace.

Tr_B in Equation (3.5) is an operation known as the partial trace and is used to extract the state of a subsystem of a composite system from its overall density matrix. The partial trace over subsystem B of a tensor product of two operators $|\xi_u\rangle\langle\xi_v|$ and $|\chi_u\rangle\langle\chi_v|$ is given by:

$$\mathrm{Tr}_B\left(|\xi_u\rangle\langle\xi_v| \otimes |\chi_u\rangle\langle\chi_v|\right) \equiv |\xi_u\rangle\langle\xi_v| \,\mathrm{Tr}\left(|\chi_u\rangle\langle\chi_v|\right). \tag{3.6}$$

$|\xi_u\rangle$ and $|\xi_v\rangle$ are arbitrary states in the subspace of A, and $|\chi_u\rangle$ and $|\chi_v\rangle$ arbitrary states in the subspace of B. Tr is the standard trace operation, which for two arbitrary states $\mathrm{Tr}\left(|\chi_u\rangle\langle\chi_v|\right) = \langle\chi_v|\chi_u\rangle$. Similarly, the reduced density matrix of subsystem B can be obtained by taking the partial trace over subsystem A of the tensor product of two operators:

$$\mathrm{Tr}_A\left(|\xi_u\rangle\langle\xi_v| \otimes |\chi_u\rangle\langle\chi_v|\right) \equiv \mathrm{Tr}\left(|\xi_u\rangle\langle\xi_v|\right) |\chi_u\rangle\langle\chi_v|. \tag{3.7}$$

As an example, let's reconsider the pure entangled state:

$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}\left(|0_A 0_B\rangle + |1_A 1_B\rangle\right). \tag{3.8}$$

This system is then composed of single-qubit subsystem A with basis vectors $\{|\xi_1\rangle, |\xi_2\rangle\} = \{|0_A\rangle, |1_A\rangle\}$, and single-qubit subsystem B with basis vectors $\{|\chi_1\rangle, |\chi_2\rangle\} = \{|0_B\rangle, |1_B\rangle\}$. We know that this system is not separable (i.e., $|\chi_{AB}\rangle \neq |\chi_A\rangle \otimes |\chi_B\rangle$); however, by using the reduced density matrix, we can find a full description for subsystems A and B as follows.

The density matrix of our state $|\psi_{AB}\rangle$ can be expressed in terms of outer products of the basis vectors as:

$$\begin{aligned} \rho_{AB} &= |\psi_{AB}\rangle\langle\psi_{AB}|, \\ &= \frac{1}{2}\left[|0_A 0_B\rangle\langle 0_A 0_B| + |0_A 0_B\rangle\langle 1_A 1_B| + |1_A 1_B\rangle\langle 0_A 0_B| + |1_A 1_B\rangle\langle 1_A 1_B|\right]. \end{aligned} \tag{3.9}$$

Now, to calculate the reduced density matrix for, let's say, subsystem B, we have:

$$\begin{aligned} \rho_B = \mathrm{Tr}_A(\rho_{AB}) = \frac{1}{2}\big[&\mathrm{Tr}_A(|0_A 0_B\rangle\langle 0_A 0_B|) + \mathrm{Tr}_A(|0_A 0_B\rangle\langle 1_A 1_B|) \\ &+\mathrm{Tr}_A(|1_A 1_B\rangle\langle 0_A 0_B|) + \mathrm{Tr}_A(|1_A 1_B\rangle\langle 1_A 1_B|)\big] \end{aligned}$$

$$\begin{aligned} = \frac{1}{2}\big[&\mathrm{Tr}(|0_A\rangle\langle 0_A|)|0_B\rangle\langle 0_B| + \mathrm{Tr}(|0_A\rangle\langle 1_A|)|0_B\rangle\langle 1_B| \\ &+\mathrm{Tr}(|1_A\rangle\langle 0_A|)|1_B\rangle\langle 0_B| + \mathrm{Tr}(|1_A\rangle\langle 1_A|)|1_B\rangle\langle 1_B|\big] \end{aligned}$$

$$\begin{aligned} = \frac{1}{2}\big[&\langle 0_A|0_A\rangle|0_B\rangle\langle 0_B| + \langle 1_A|0_A\rangle|0_B\rangle\langle 1_B| \\ &+\langle 0_A|1_A\rangle|1_B\rangle\langle 0_B| + \langle 1_A|1_A\rangle|1_B\rangle\langle 1_B|\big] \end{aligned}$$

$$= \frac{1}{2}\left[|0_B\rangle\langle 0_B| + |1_B\rangle\langle 1_B|\right] = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

3.4. REDUCED DENSITY MATRIX: A TOOL FOR UNDERSTANDING PURITY OF THE STATE

Starting with the pure entangled state $|\psi_{AB}\rangle$, it is shown that the reduced density matrix of subsystem B is the mixed state ρ_B, which may seem counterintuitive. However, it is essential to note that the reduced density matrix ρ_B describes the statistical outcomes of subsystem B, obtained by averaging out the measurement outcomes of subsystem A. In other words, tracing out subsystem A from the overall density matrix results in the reduced density matrix of subsystem B, which describes the state of subsystem B in terms of its own probability distribution. This is an essential tool for analyzing the properties of entangled states in complex quantum systems, and it allows us to extract meaningful information about the state of individual subsystems even when they are entangled with one another. The information presented about the density matrix and reduced density matrix in this section has been extracted in detail from the References [20, 29, 101], specifically from the section titled "The density matrix and mixed states" [101]. This section provides a comprehensive introduction to density matrix formalism, including its use in describing mixed states and entangled states and its applications in quantum information and computation.

The reduced density matrix of a subsystem can reveal the degree of entanglement between the qubits in that subsystem but not necessarily the degree of entanglement of the entire system. Therefore, calculating the entanglement measure of a quantum circuit by tracing out all qubits is necessary to obtain an accurate assessment of the entanglement of the circuit. The entanglement of a multi-qubit circuit can only be determined by analyzing the joint quantum state of all the qubits and not just the reduced density matrix of individual qubits.

Chapter 4

Building quantum cellular automata with evolutionary algorithms

> ***"The most important application of quantum computing in the future is likely to be a computer simulation of quantum systems because that's an application where we know for sure that quantum systems, in general, cannot be efficiently simulated on a classical computer."***
>
> **-DAVID DEUTSCH**

This chapter describes an experimental study aimed at demonstrating the feasibility and performance of a quantum circuit generation algorithm that was designed as outlined in Chapter 3. The methodology used in the study resulted in the generation of quantum circuits, which were subsequently executed on a quantum simulator to demonstrate its competitive performance. The results of this study were presented at the 15^{th} International Conference and School "Cellular Automata for Research and Industry," and a full paper available online at Ref. [1].

The initial experiment was carried out in collaboration with Shailendra Bhandari, Sebastian Overskott, and Ioannis Adamopoulos and was supervised by Pedro G. Lind, Sergiy Denysov, and Stefano Nichele as part of the research project in the Nordic Center for Sustainable and Trustworthy AI Research (NordSTAR) [102]. In this thesis work, the primary objective was to showcase the efficiency of the proposed quantum algorithm, which requires fewer qubits for implementation, by executing it on a quantum simulator. A more significant number of runs per experiment were performed throughout the thesis work to ensure a comprehensive analysis. The following steps outline the process undertaken in this thesis work:

- **Algorithm refinement**: The proposed algorithm was fine-tuned to produce the desired results, taking into consideration the specific constraints and requirements of the problem at hand.

- **Experiment design**: A series of experiments were re-designed to thoroughly assess the algorithm's performance on a quantum simulator, taking into account various parameters and use-case scenarios.

- **Data analysis**: The data obtained from the experiments was carefully examined to identify trends, patterns, and potential areas for improvement. This analysis enabled us to gain insights into the proposed initial algorithm's efficiency, accuracy, and reliability.

- **Documentation and paper submission**: The findings, insights, and conclusions drawn from the research were documented in detail, forming the basis for a research paper. This paper was submitted during the thesis work, outlining the proposed initial algorithm, its implementation, and the results obtained from the experiments. The text and figures presented in this chapter are transcribed from Ref. [1] and provide compelling evidence for our proposed algorithm's feasibility and competitive performance.

Overall, an evolutionary algorithm is applied to generate specific rules of deterministic and stochastic cellular automata (CA) using the set of five quantum gates known to generate any quantum circuits. A mutation-based evolutionary algorithm is used to build the quantum circuits, which allows the optimization of quantum gate types and their connectivity. A certain number of chromosomes is assigned with the number of quantum gates per circuit, generating the initial number of chromosomes. Then, iteratively, the genetic algorithm evolves the current chromosomes into new ones, identifying the chromosome with the highest fitness in each generation, which will be a parent for the next generation. To assess the performance of the quantum circuits we derived, we measure only the first qubit (q_0) and consider two different fitness functions to compare the measured probability of an initial state computed from the population and the corresponding desired probability. Two different fitness functions are used for this, the sum of absolute difference and the Kullback-Leibler divergence (KL).

4.1 Mutation and fitness function

The first step in the evolutionary iteration of the algorithm is to assign a certain number of chromosomes, each with an appropriate number of quantum gates per circuit. Subsequently, the initial number of chromosomes is generated. The evolutionary cycle begins by identifying the best-fit chromosomes, evolving the current chromosomes into new ones, and inserting the best

Neighbors	Sto. CA Prob.	Rule90	Rule110	Rand. Prob. 1	Rand. Prob. 2	Rand. Prob. 3
[0, 0, 0]	0.394221	0	0	0.6364	0.4778	0.1988
[0, 0, 1]	0.094721	1	1	0.6603	0.5604	0.4701
[0, 1, 0]	0.239492	0	1	0.5261	0.8528	0.9836
[0, 1, 1]	0.408455	1	1	0.1748	0.4818	0.7115
[1, 0, 0]	0	1	0	0.8820	0.3143	0.6616
[1, 0, 1]	0.730203	0	1	0.3371	0.3464	0.1218
[1, 1, 0]	0.915034	1	1	0.0340	0.0678	0.1328
[1, 1, 1]	1	0	0	0.4444	0.9124	0.7306

Table 4.1. The deterministic and stochastic CAs considered in this paper. For the stochastic cases, the values indicate the probability of an update of value 1 for the middle cells in the triad neighborhood. For the deterministic cases, the values indicate the exact update imposed.

one as the first element of the new population. This evolution cycle continues for the desired number of generations, with the best chromosome in each generation becoming the parent for the next.

Every chromosome undergoes a simple mutation process at each evolution step to generate the best-mutated chromosomes. Mutation can occur in one of two ways: by replacing the gates from the pool of gates in the chromosome with a randomly generated new one or by replacing the entire chromosome to generate the four best parents. The mutation process is probabilistic, with the four best chromosomes left unchanged as "elites," while the remaining chromosomes are evolved with probabilistic selection, where the probability of each circuit becoming a parent is proportional to its fitness.

We consider one-dimensional CAs composed of cells with two possible states, 0 or 1 (Boolean CA), which are updated according to one out of 256 possible rules matching each one of the eight neighborhoods ([0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0] and [1, 1, 1]) to an update of the middle cell. Table 4.1 shows the matching in all cases considered in this thesis.

To evaluate the performance of the quantum circuits, we measure only the first qubit (q0) and consider two different fitness functions for comparison. The first function is the sum of the absolute differences between the measured probability Q(ω) of an initial state ω, computed from the chromosome, and the corresponding desired probability $P(\omega)$. The fitness function is given by:

$$F = \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)|. \tag{4.1}$$

This fitness value ranges between 0 and 8. The second fitness function is the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions and has been used in other works [103, 104] as a fitness function:

$$D_{KL}(P||Q) = \sum_{\omega \in \Omega} P(\omega) log\Big(\frac{P(\omega)}{Q(\omega)}\Big). \tag{4.2}$$

In our case, the distributions are discrete, and each one has eight different values related to the eight different initial states of the three qubits. If the distributions are a perfect match, then both fitness functions are zero. There are differences between the sum of absolute differences and the Kullback-Leibler divergence. The former treats each probability as a pure value, while the latter considers deviations of higher probability with a stronger weight than lower probability values.

4.2 Three different "flavors" of quantum cellular automata

In this study, we consider three different types of quantum cellular automata, starting with a stochastic *critical* cellular automaton, followed by more general stochastic CA (random updates) and a few deterministic rules, namely, rule 90 and 110 [73]. The updates for each type of CA are shown in Table 4.1. To ensure the robustness of our framework, we fix a maximum number of chromosomes at $N_c = 20$ and a maximum number of generations at $N_g = 150$. The simulation is repeated for 10 and 20 initial conditions, denoted by $N_{ic} = 10$ and $N_{ic} = 20$, respectively, and randomly chosen.

4.2.1 The quantum cousin of a stochastic critical CA

The authors in [105] have evolved a stochastic cellular automata model to reach criticality, a property of dynamical systems that allows them to do robust computations. For each triad pattern, a probability has been calculated through a genetic algorithm for the central cell to have state 1. These probabilities are shown in Table 4.1, second column.

We start by considering the number of gates used, evolving quantum CAs with 3, 5, 10, 15, and 20 gates and ten runs. The mutation probability is fixed at 10 percent.

The results for both fitness functions are shown in Figure 4.1. It is evident from the figure that the best fitness score per run improves with increasing the number of gates until 15 gates, after which a gradual increase is observed for 20 gates. A similar experiment was run with 20 initial run conditions with more statistics, but the results are qualitatively the same, as shown

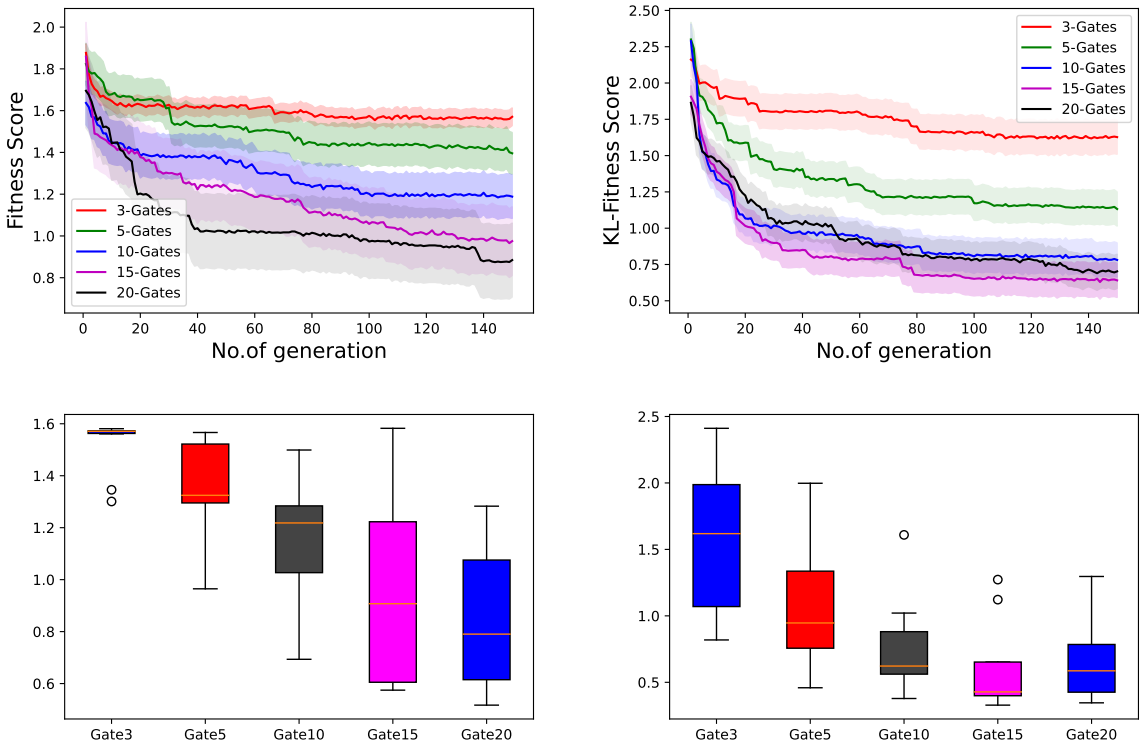


Figure 4.1. (Top) The fitness scores as a function of the number of generations, for **different number of gates**. (Bottom) The number of gates vs. the best fitness scores. In each case, we show the result for (left) the absolute sum of differences, Eq. (4.1), and (right) Kullback-Leibler divergence, Eq. (4.2). The fitness scores of each gate for the box plots are the best fitness scores per run, and the fitness scores for the lower two plots are the average fitness scores for ten runs.

in Figure (Bottom) 4.2. Therefore, for experiments 2 and 3, the number of gates used is 15. Notably, the fitness score is optimal for 15 gates in the case of the KL fitness function, while the fitness score is optimal for 20 gates in the absolute difference of probabilities fitness function with ten runs. However, it is challenging to differentiate the same condition with 20 runs, indicating a gradual improvement in the fitness score for 20 gates, as shown in Figure (Top) 4.2.

Furthermore, the sum of absolute differences performs better for cases with the lowest number of gates, while the KL fitness function is better suited when the number of gates increases. Next, we explore how the fitness changes when changing the mutation probability. The goal is to test the impact of the mutation on the fitness function of the different generations. We fix the number of gates at 15 and select mutation probabilities of 5%, 10%, 20%, 30%, and 50%.

Figure 4.3 (left) shows a comparison of the fitness scores with the number of generations at different mutation rates for the absolute difference of probabilities fitness function, while Figure 4.3 (right) shows a comparison of the fitness scores with the number of generations at different mutation rates for the KL fitness function with ten runs. In the case of the KL divergence fitness function, we observe a gradual improvement in results with an increase in

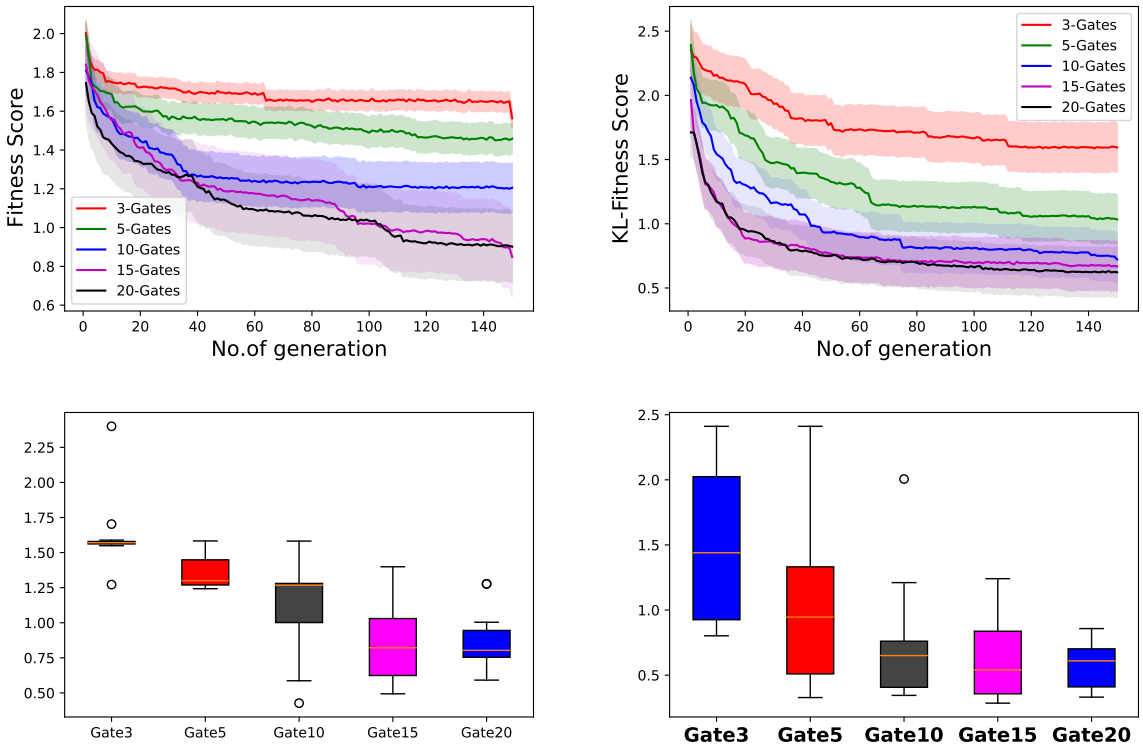


Figure 4.2. (Top) The fitness scores as a function of the number of generations, for **different number of gates**. (Bottom) The number of gates vs. the best fitness scores. In each case, we show the result for (left) the absolute sum of differences, Eq. (4.1), and (right) Kullback-Leibler divergence, Eq. (4.2). The fitness scores of each gate for the box plots are the best fitness scores per run, and the fitness scores for the lower two plots are the average fitness scores for 20 runs.

the percentage of the mutation rate. However, a similar conclusion cannot be drawn for other fitness scores as the results are random.

With 20 runs, the overall fitness scores of all mutation probabilities improve for both fitness functions, as shown in Figure 4.4. While more random initial conditions are possible, computing time increases exponentially with an increase in the number of runs.

4.2.2 The two other flavors: deterministic CA and stochastic CA but non-critical

We conclude our investigation by applying the same framework to deterministic rules as well as to stochastic CAs that do not exhibit critical behavior. For this experiment, we fix the parameters with 15 gates and a 10% mutation probability. The desired probabilities for Rule 90 and Rule 110 (deterministic rules with probabilities of 0 or 1 for the eight neighborhoods) and eight randomly generated probabilities with three repetitions are used as target probabilities. The target probabilities for each condition are shown in Table 4.1, and results are shown in Figure 4.5.

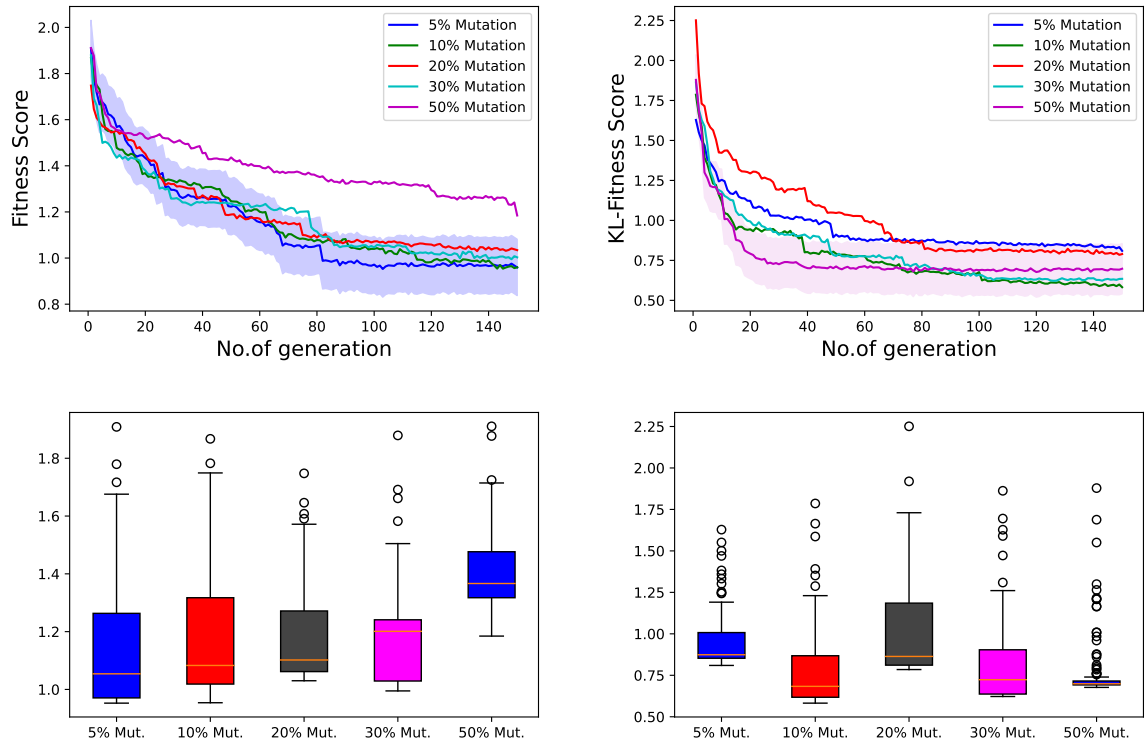


Figure 4.3. (Top) The fitness scores as a function of the number of generations, for **different mutation rates**. (Bottom) The number of gates vs. the best fitness scores. In each case, we show the result for (left) the sum of absolute differences (Eq. (4.1)), and (right) the KL divergence.

The fitness scores for the deterministic CA (Rule 90 and Rule 110) are excellent, as shown in figure Figure 4.5 (left); therefore, it is quite successful in running the quantum circuit for deterministic CA. The results of the stochastic CA with randomly generated probabilities with three repetitions are promising, with the best fitness scores of 0.14 and 0.27 for Random2 and Random3. While the best fitness score for Random1 is 0.685, it still needs further experiments to tune the evolution and reduce the difference further. Figure 4.5 (right) shows the fitness scores for the different sets of probabilities against the number of generations for two deterministic CA (Rule 90 and Rule 110) and the stochastic CA with randomly generated probabilities with three repetitions (Random1, Random2, and Random3) with ten initial run condition. Similarly, Figure 4.6 shows better improvement in the fitness scores for Random1, Random2, and Random3 with 20 initial run conditions. However, the fitness scores remain the same with 20 initial conditions for two deterministic CA (Rule 90 and Rule 110).

Finally, we present the quantum circuits generated by our algorithms. Figure 4.7 shows the circuit generated by a run that scored much better than the average runs for the critical stochastic CA, with a fitness score of 0.3404 using the KL fitness function in Equation (4.2).

Similarly, Figure 4.8 shows the circuit generated by the deterministic CA Rule 110. It has the

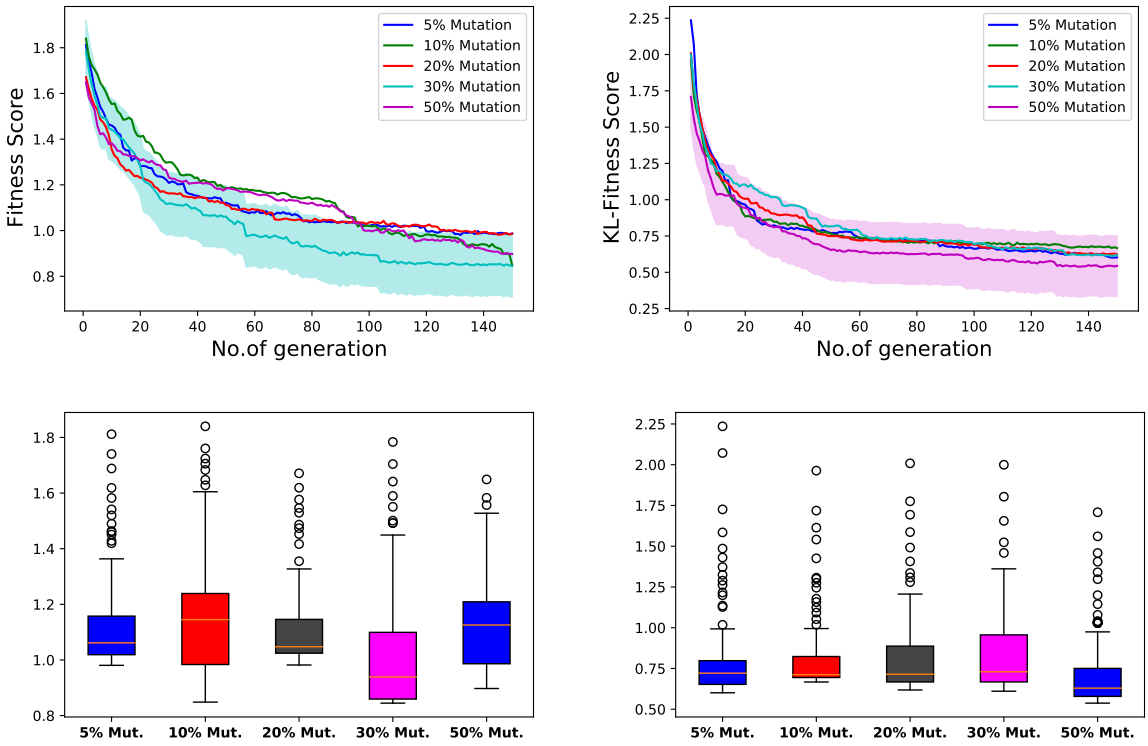


Figure 4.4. (Top) The fitness scores as a function of the number of generations, for **different mutation rates**. (Bottom) Mutation rates vs. the best fitness scores. In each case, we show the result for (left) the sum of absolute differences (Eq. (4.1)), and (right) the KL divergence. The fitness scores of each gate are the average fitness scores of 20 runs.

best fitness score of 0.000921, with 15 gates, 10% mutation probability, and the KL fitness function. The results of the run with their respective outcomes and fitness scores are displayed in the console in Appendix A.

4.3 Discussion and possible extensions

We have presented a simple framework for deriving quantum circuits that reproduce specific CA rules using genetic algorithms. Our quantum-inspired genetic algorithm uses a probabilistic representation based on quantum bits to encode the populations (chromosomes). It employs quantum gates to direct the evolutionary process according to the fitness function. We have shown that the framework can evolve quantum circuits towards several types of CA rules, ranging from deterministic rules to stochastic updates.

Figure 4.9 shows the fitness scores for all the CA rules we used to evolve quantum circuits. The figure shows that the best fitness score for critical CA is 0.3404. For two deterministic CA rules (Rule 90 and Rule 110), the fitness scores are very close to zero, and we consider the fitness improvement as the minimization of the absolute difference between the initial and final states. This is expected for deterministic CA, as the probability values are either 0 or 1. However, in

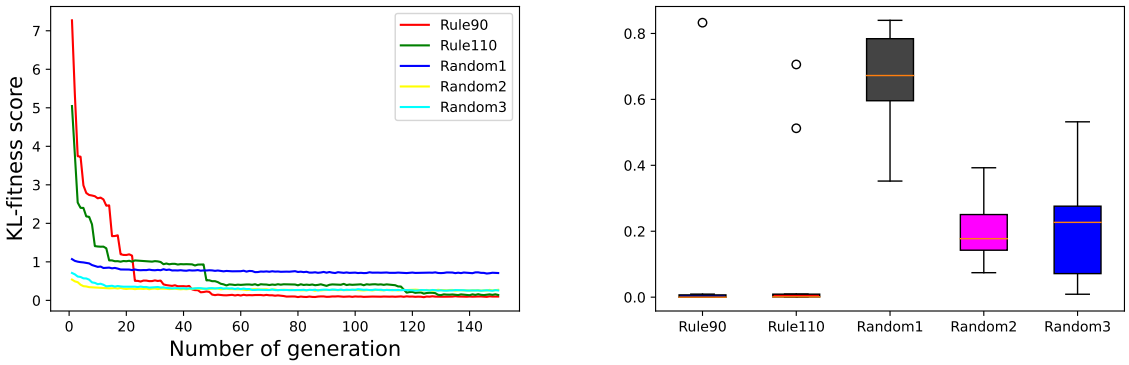


Figure 4.5. (Left) Fitness scores for different sets of probabilities against the number of generations for the KL-fitness function. (Right) Best fitness scores per run for KL-fitness function for different sets of probabilities. The fitness scores of each gate are the average fitness scores of 10 runs.

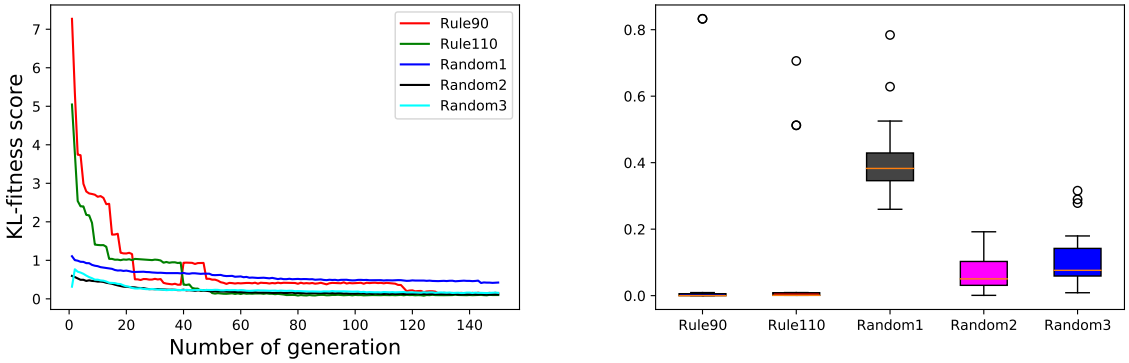


Figure 4.6. (Left) Fitness scores for different sets of probabilities against the number of generations for the KL-fitness function. (Right) Best fitness scores per run for KL-fitness function for different sets of probabilities. The fitness scores of each gate are the average fitness scores of 20 runs.

the case of randomly generated stochastic CA rules, the fitness scores vary for the three different randomly generated probabilities. We can see that Random 2 has the best fitness scores among the three. However, to fully understand why this is the case, we need to run more experiments with more generations and a higher number of runs. Typically, it is difficult to determine the exact reason for variations in fitness scores between different randomly generated probabilities.

One important observation we made during our experiments is that the fitness functions of the parents that survived to the next generation were not always better than their previous generation's fitness functions. In fact, the fitness functions of the parents in the next generation were sometimes worse than those they had in the previous generation. Mutated solutions with better fitness functions would then take their place as parents, but still, these functions could be worse than those that the original parents had. As a result, we did not observe a gradual decrease in the value of the best fitness function from generation to generation, as we expected. Instead, the value of the best fitness function had ups and downs.

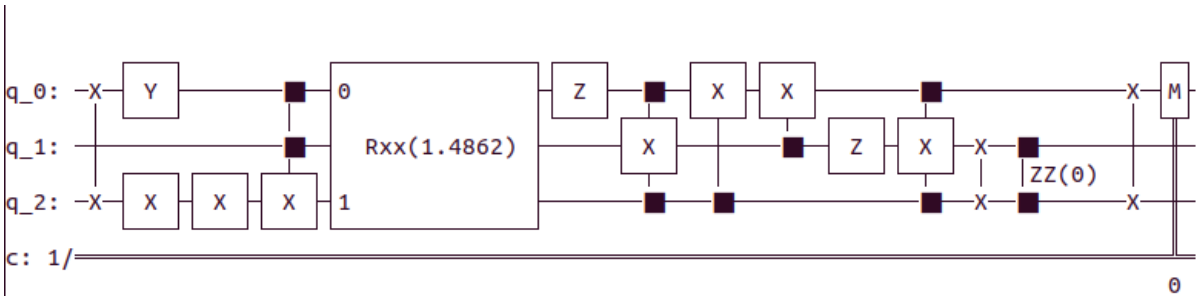


Figure 4.7. Visualization of a circuit created by 15 gates, 10% mutation probability with KL fitness scores. The circuit is the best-produced circuit with a fitness score of 0.3404 for critical stochastic CA.

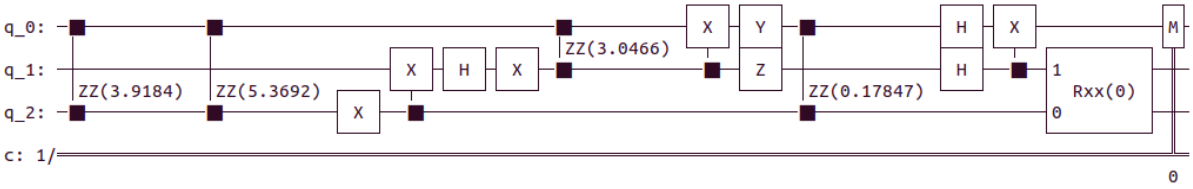


Figure 4.8. Visualization of a circuit created by 15 gates, 10% mutation probability with KL fitness scores for deterministic CA (Rule 110). The circuit is the best-produced circuit with a fitness score of 0.000921.

This phenomenon occurred for both types of fitness functions that we used, and the reason is as follows. Each quantum circuit corresponding to a chromosome or a solution is executed multiple times (1000 in our case) with a simulator. The outcome probabilities that are related to the fitness function can't be the same each time we run the same circuit in another generation. Even when we increased the number of shots in the simulator to stabilize the probability value (10000 and then 50000), we did not succeed in eliminating the slight differences in results we observed between different runs.

Another thing we realized is that, for every set of desired probability outcomes, we can achieve approximately the same fitness function score for all mutation rates and both types of fitness functions (Equation (4.1) and Equation (4.2)), as illustrated in Figure 4.3. One possible reason for this could be that the pool of potential solutions is not very large, given the number of gates we used, even though the types of gates were sufficient to build almost any quantum circuit.

Regarding the two different fitness functions, we cannot say that one performs better than the other. One small observation is that the diagrams corresponding to the KL fitness function have more gentle fluctuations, and they reach their final range of values in the early generations.

Moreover, some parameters could be further tested in future investigations, as well as considering additional genetic operators in evolutionary algorithms, such as crossover. It would be interesting to see how this works, although we believe that such a thing would completely change the topology of our circuit, and therefore, the solutions would be random ones, which is not the purpose of crossover. As for the fitness functions, we could also try to improve them or

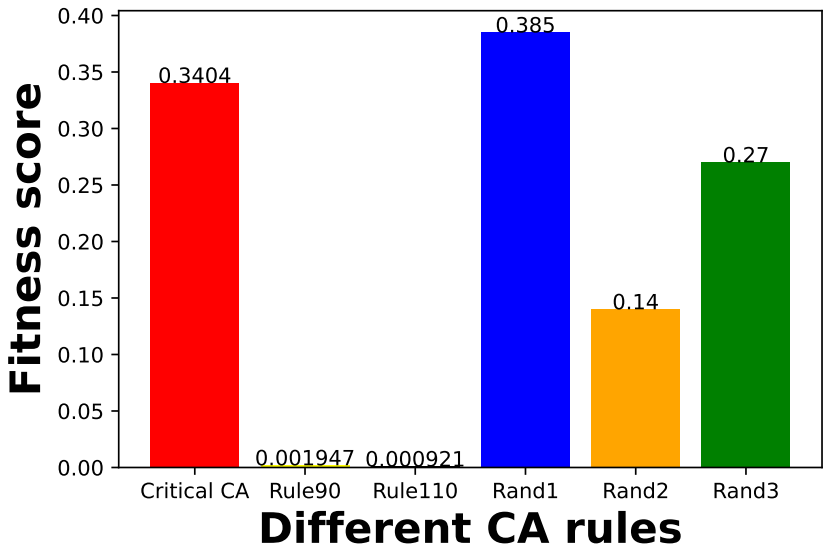


Figure 4.9. Comparison of fitness scores for stochastic CA, deterministic CA (Rule 90 and Rule 110), and the randomly generated stochastic CA repeated three times.

find a more complex one to achieve better results.

In conclusion, we presented a framework for deriving quantum circuits for cellular automata rules using quantum-inspired genetic algorithms. We demonstrated the effectiveness of this framework by evolving quantum circuits for various types of cellular automata, ranging from deterministic to stochastic updates. We showed that the framework could successfully generate quantum circuits with good fitness scores for various types of cellular automata rules. We also observed some interesting phenomena during the experiments, such as the ups and downs of fitness scores for parents and the approximately common fitness score for different mutation rates. Overall, this framework provides a promising direction for generating quantum circuits using CA rules, and further investigations can be done to improve its performance and explore its potential.

Figure 4.9 shows the fitness scores for all the CA rules that we used to evolve quantum circuits. The figure shows that the best fitness score was achieved for the critical CA with a value of 0.3404. For the two deterministic CA rules (Rule 90 and Rule 110), the fitness scores were very close to zero, and we considered the fitness improvement as the minimization of the absolute difference between the initial and final states. This was expected for deterministic CA, as the probability values were either 0 or 1. However, in the case of randomly generated stochastic CA rules, the fitness scores varied for the three different randomly generated probabilities. We observed that the fitness score for Random 2 was the best among the three, but it is difficult to determine why the others did not perform as well. Further experiments with more generations and runs would be necessary to test and identify the factors affecting the results. All the results

of the runs, including the outcomes and the best fitness scores for all the CA rules, are displayed in the console in Appendix A.

Chapter 5

Evolutionary design of quantum circuits for maximal entanglement

> ***Quantum computing is not just about speed; it's about exploring new modes of computation. Quantum Evolutionary Algorithms are a great example of this, as they represent a new paradigm for optimization that goes beyond classical algorithms.***
>
> **ALÁN ASPURU-GUZIK**

This chapter presents the results of an evolutionary algorithm for quantum circuit design that employs the Mayor-Wallach entanglement measure as a fitness function. Our experiments involved three, four, and five-qubit circuits, and the algorithm's performance was evaluated using varying mutation probabilities and numbers of gates. We conducted 50 iterations over 500 generations and presented the results as the mean fitness with standard error of the mean and the best fitness with its standard error. The results are elaborated separately for the three, four, and five-qubit circuits.

Quantum entanglement is a fundamental aspect of quantum computing that enables the generation of correlations between quantum systems. However, designing efficient quantum circuits with maximum entanglement can be challenging due to the exponential increase in the number of parameters with the problem size. Evolutionary algorithms (EAs) have emerged as a promising tool for quantum circuit design [106–108]. This chapter presents our implementation of EAs for quantum circuit design, which aims to maximize entanglement. The QUEVO1 framework employs an evolutionary algorithm to automatically generate quantum circuits that

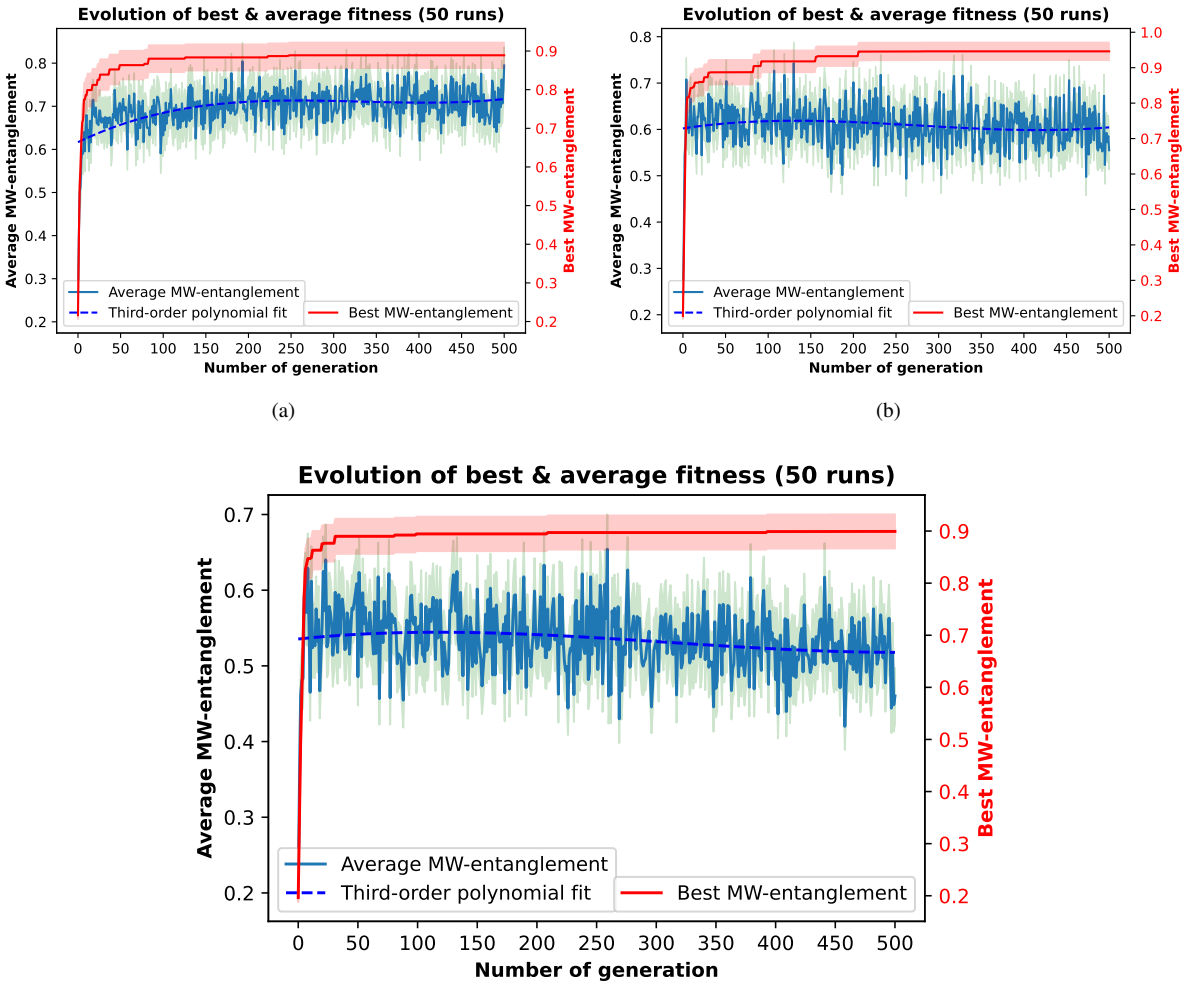
satisfy defined properties specified through a fitness function. The evolution properties can be controlled by tuning parameters such as initial population, number of generations, probability of mutation operator, number of gates in the circuit, and the chosen fitness function.

5.1 Evolutionary algorithm for three-qubit quantum circuit design

5.1.1 Tuning evolutionary parameters for three-qubit circuit design: Mutation rate and gate number selection

Figure 5.1 displays the average and best fitness scores of a three-qubit quantum circuit generated by an evolutionary algorithm over 500 generations. This study investigated the effectiveness of different mutation rates (5%, 10%, and 15%) in generating a quantum circuit having different numbers of gates (3, 5, 8, 10, and 12) with maximum entanglement. The best fitness scores obtained for all mutation rates approached one, indicating that the generated circuits have a high degree of entanglement between the qubits. The fitness function used in the study is the Mayor-Wallach entanglement measure [42], which quantifies the amount of entanglement in a multi-partite quantum system by computing the eigenvalues of the partially transposed density matrix. The fitness value of the entanglement measure ranges between 0 and 1, with 1 representing the maximum entanglement measured.

The results of our research suggest that the implementation of evolutionary algorithms presents a compelling methodology for the construction of three-qubit quantum circuits while maintaining a predefined degree of entanglement measure. The optimal mutation rate was found to be 10%, which struck a balance between exploring different solutions in the search space and exploiting the best solutions. A mutation probability of 1/9, i.e., 11.11%, is needed to replace at least one individual population consisting of nine integer lists (three integer gates for each qubit). Therefore, a 10% mutation probability, which can at least replace one individual population in each generation, can be considered an optimal probability for the mutation rate. The observation is supported by Figure 5.2a, which suggests that when the mutation rate is lower, a balance between exploration and exploitation can be achieved, resulting in a higher average fitness score across all runs. In addition, Figure 5.2a shows the best fitness score obtained for each run over 50 runs is highest for a mutation rate of 10%. This indicates that a lower mutation rate is able to strike a balance between exploration and exploitation, leading to a higher average fitness score across all runs. On the other hand, a mutation rate of 10% may be more effective in discovering the best-performing circuit in each run due to the higher degree of exploration it allows.



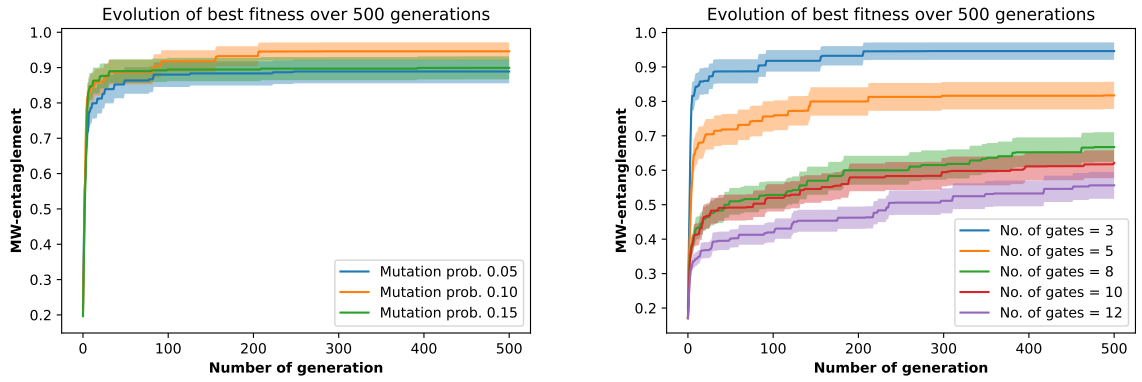
(a)

(b)

(c)

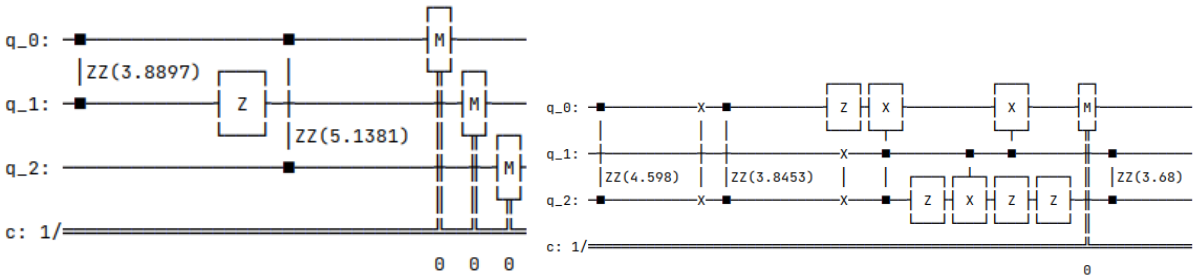
Figure 5.1. Evolutionary optimization of three-qubit quantum circuits with three gates using the Meyer-Wallach entanglement measure as the fitness function. The plot shows the mean fitness (green line) and its shaded standard error, as well as the mean of the best fitness (red line) and its shaded standard error, against the number of generations for different mutation percentages in the evolutionary algorithm: (a) 5%, (b) 10%, and (c) 15%. The blue dashed line represents the third-order polynomial fit to the mean fitness.

In the experiment where the number of gates in the quantum circuit varied from 3 to 12 while the mutation rate was fixed at 10%, it was observed that the fitness score decreased as the number of gates in the circuit increased, as shown in Figure 5.2b. This suggests that the fitness score reduces as the circuit becomes more complex with increasing quantum gates, indicating the decrease in the entangling capability of thus generated quantum circuits. There could be a trade-off between circuit complexity and performance, and there may be an optimal number of gates that provides the best balance between the two factors.



(a) (b)

Figure 5.2. Comparison of best fitness generated for (a) different mutation rates for three gates and (b) different numbers of gates for a constant mutation probability of 10%. The results are averaged over 50 runs, and the error bars represent the standard error of the mean best fitness.



(a) (b)

Figure 5.3. Two examples of three-qubit circuits evolved with a 10% mutation probability and optimized for MW-entanglement fitness scores. (a) A circuit with only three gates achieved a high fitness score of 0.999. (b) A more complex circuit with 12 gates achieved a slightly lower fitness score of 0.999.

5.1.2 Visualizing best-fit three-qubit circuits with state vectors and reduced density matrices

Figure 5.3a shows the visualization of the best-generated three-qubit quantum circuit with three gates with a fitness score of 0.999, providing a concrete example of the potential of evolutionary algorithms for designing highly entangled quantum circuits. The state vector and reduced density matrix for a specific circuit—comprising three qubits and three gates—after 500 iterations, are outlined in A and in Equation (5.2), respectively. In the field of quantum mechanics, the state of any given quantum system can be succinctly represented by what is known as a state vector. This vector comprises complex numbers, each of which corresponds to the probability amplitude of a specific basis state or possible outcome. It is essential to note that the sum of the squared absolute values of these complex numbers is normalized to equal unity in accordance with the principles delineated in the Qiskit textbook [109]. Consequently, by evaluating these state vector elements, we can ascertain the probability distribution of different outcomes that might be observed upon the measurement of the quantum system. In our QUEVO1 module includes the function *get_statevector*(), which retrieves the state vector of a given quantum circuit. This state vector is then utilized by

the *compute_MW_entanglement*() function, function to determine the circuit's entanglement using the Meyer-Wallach entanglement measure method. To accomplish this, the function first converts the state vector into a density matrix through the outer product of the normalized state vector and its complex conjugate. Next, the function computes the partial trace of the density matrix for each qubit, excluding all other qubits in the system. This process yields a reduced density matrix that allows for the calculation of the Meyer-Wallach entanglement measure, which is then added to the overall entanglement sum. Figure 5.3b shows another example of a three-qubit quantum circuit with 12 gates having a fitness value of 0.999. The state vector and reduced density matrix of the circuit shown in Figure 5.3b are given by B and Equation (5.4), respectively. These three-qubit quantum circuits appear to be relatively simple, yet they achieve a very high degree of entanglement between the qubits.

$$\mathbf{A} = \begin{cases} \text{State vector} = (-0.2678 + 0.165i)\,|000\rangle + (-0.199 - 0.125i)\,|001\rangle + \\ \qquad (-0.008 + 0.454i)\,|010\rangle + (0.385 - 0.198i)\,|011\rangle + \\ \qquad (-0.58 - 0.016i)\,|100\rangle + (-0.069 - 0.165i)\,|101\rangle + \\ \qquad (0.163 - 0.2724i)\,|110\rangle + (0.2641 - 0.019i)\,|111\rangle\,, \end{cases} \tag{5.1}$$

$$\textbf{Matrices} = \begin{cases} k_0 = \begin{bmatrix} 0.3294684 + 0.i & 0.145521 - 0.083587i \\ 0.14552 + 0.08358i & 0.23164 + 0.i \end{bmatrix}, \\ k_1 = \begin{bmatrix} 0.215855 + 0.i & -0.06664 - 0.15040i \\ -0.06664 + 0.1504i & 0.34684 + 0.i \end{bmatrix}, \\ k_2 = \begin{bmatrix} 0.4312 + 0.i & 0.0247 - 0.0430i \\ 0.024735 + 0.04306i & 0.12162 + 0.i \end{bmatrix}. \end{cases} \tag{5.2}$$

$$\mathbf{B} = \begin{cases} \text{State vector} = (-0.023 - 0.627i)\,|000\rangle + (-0.169 + 0.194i)\,|001\rangle + \\ \qquad (-0.031 + 0.038i)\,|010\rangle + (-0.398 - 0.081i)\,|011\rangle + \\ \qquad (0.004 - 0.517i)\,|100\rangle + (-0.036 + 0.077i)\,|101\rangle + \\ \qquad (0.043 - 0.150i)\,|110\rangle + (-0.081 + 0.252i)\,|111\rangle\,, \end{cases} \tag{5.3}$$

$$\textbf{Matrices} = \begin{cases} k_0 = \begin{bmatrix} 0.52680 + 0.i & 0.3506 + 0.0941i \\ 0.35062 - 0.0941i & 0.2698 + 0.i \end{bmatrix}, \\ k_1 = \begin{bmatrix} 0.566742 + 0.i & 0.1350 - 0.0893i \\ 0.135014 + 0.0893i & 0.09635 + 0.i \end{bmatrix}, \\ k_2 = \begin{bmatrix} 0.5276 + 0.i & -0.1912 + 0.1145i \\ -0.191289 - 0.i & 0.1450 + 0.i \end{bmatrix}. \end{cases} \tag{5.4}$$

It is important to note that the robustness of these quantum circuits to noise and quantum errors is a critical factor to be analyzed in their practical applications. Therefore, it is essential to consider this aspect when using evolutionary algorithms to optimize these circuits. Future research should be conducted to investigate the resilience of circuits generated through evolutionary algorithms to circuit depth and noise and identify methods for improving their robustness. Additionally, it is worth mentioning that the choice of gate set used in the study could impact the results. Different gate sets may have varying degrees of expressivity, leading to different entanglement and fitness scores in the generated circuits. Future studies could explore the effectiveness of different gate sets so we can better design practical quantum circuits for real-world applications.

In our QUEVO1 module of the evolutionary algorithm, the reduced density matrices for a three-qubit system are computed by tracing over the remaining two qubits in the system. To calculate the reduced density matrix of three qubits, $statevector$ is first reshaped into a 2-dimensional array of shape (2, 2, 2) using $np.reshape()$ function. Then, a loop is initiated to iterate over each qubit in the circuit, denoted by $k = 0, 1, 2$ which calculates the reduced density matrix rho_k_sq for each qubit. To find the reduced density matrix for a single qubit k, the trace of the density matrix is taken over all other qubits except for k. This is accomplished by using the $np.transpose()$ function to rearrange the axes of the density matrix, placing the k^{th} qubit in the first axis, followed by the remaining qubits. The $np.roll()$ function is then employed to shift the axes, ensuring that the k^{th} qubit becomes the last axis, allowing it to be traced out using the $np.trace()$ function. Since there are three qubits in this case, the loop is executed three times, resulting in three reduced density matrices being calculated, one for each qubit. Equations 5.1 and 5.4 show the calculated three reduced matrices for one of the circuits (three-qubits, three and twelve gates), respectively, among 500 generations. Since the output reduced density matrix is 1×2 array because the $reshape()$ function in the code is being used to convert a one-dimensional array into a multi-dimensional array with shape $[2] * n\_qubits$. The use of reduced density matrices to analyze the properties of entangled states in complex quantum systems is discussed in detail in Section. 3.4.

After calculating the reduced density matrix for all qubits, the code computes the entanglement of the circuit using the Mayor Wallach entanglement measure. The reduced density matrix of a subsystem can reveal the degree of entanglement between the qubits in that subsystem but not necessarily the degree of entanglement of the entire system. Therefore, calculating the entanglement measure of a quantum circuit by tracing out all qubits is necessary to obtain an accurate assessment of the entanglement of the circuit.

Overall, the effectiveness of evolutionary algorithms in designing three-qubit quantum circuits with a desired level of entanglement has been investigated. Our findings indicate that the choice of mutation rate and the number of gates is crucial in achieving the desired level of entanglement. We found that a mutation rate of 10% struck a balance between exploration and exploitation, leading to a higher average fitness score across all runs. Furthermore, we observed that as the number of gates in the circuit increased, the fitness score decreased, suggesting a trade-off between circuit complexity and performance. By carefully balancing exploration and exploitation through the selection of an appropriate fitness function, initial population, and mutation rate, it may be possible to generate circuits with even higher degrees of entanglement. These findings could have significant implications for the development of quantum computing and could lead to the design of new and more powerful quantum circuits.

5.2 Evolutionary algorithm for four-qubit quantum circuit design

Our research also encompasses the exploration of four-qubit quantum circuit evolution. This time, we focus on evaluating the fitness of quantum circuits comprising four gates and a 10% mutation probability. By investigating four-qubit quantum circuits with diverse gate configurations and mutation rates, we seek to understand how these evolutionary parameters affect the evolution process and the level of entanglement achievable in the circuits. This approach allows us to thoroughly assess the behavior and characteristics of four-qubit quantum circuit designs, using the Mayor-Wallach entanglement measure as the fitness function within the evolutionary algorithms.

Figure 5.4 shows the evolutionary algorithm optimization of four-qubit quantum circuits with four gates and a 10% mutation probability, using the Meyer-Wallach entanglement measure as the fitness function. The plot shows the mean fitness (green line) and its shaded standard error, as well as the mean of the best fitness (red line) and its shaded standard error, against the number of generations. The blue dashed line represents the third-order polynomial fit to the mean fitness. It can be seen from the figure that the mean fitness score is near 0.5, while the mean of the best fitness score is close to 0.9. This observation suggests that the quantum circuits generated with the best fitness score in each generation have a very high degree of entanglement

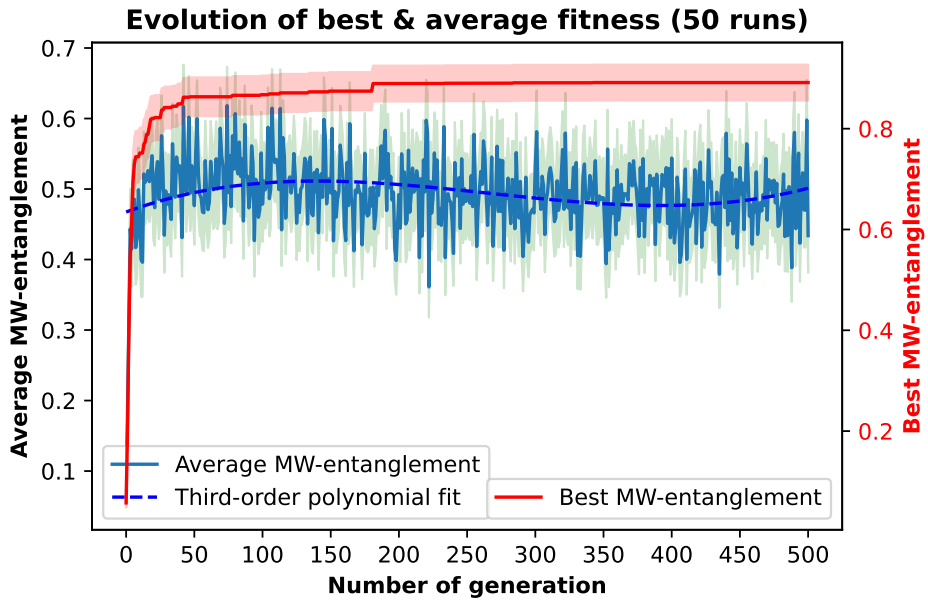


Figure 5.4. Evolutionary algorithm optimization of four-qubit quantum circuits with four gates and 10% mutation probability, using the Meyer-Wallach entanglement measure as the fitness function. The plot shows the mean fitness (green line) and its shaded standard error, as well as the mean of the best fitness (red line) and its shaded standard error, against the number of generations. The blue dashed line represents the third-order polynomial fit to the mean fitness.

among the qubits, emphasizing the effectiveness of the evolutionary algorithm optimization in generating highly entangled quantum circuits.

The experiment with a 10% mutation probability for four-qubit circuits was repeated for circuits with varying numbers of quantum gates. Figure 5.5 displays the best fitness scores for circuits with 4 to 12 gates. The figure shows that the circuit becomes more complex as the number of gates increases and the fitness score decreases, indicating lower entanglement. This finding is consistent with the results obtained for the three-qubit quantum circuit we discussed above, suggesting a potential trade-off between circuit complexity and entanglement capability.

Figure 5.6 represents an example of a four-qubit circuit that was generated using an evolutionary algorithm, consisting of four gates and having a fitness score of 0.999. This demonstrates that evolutionary algorithms hold promise as a tool for designing quantum circuits exhibiting high entanglement levels between qubits.

In order to compute the reduced density matrix (ρ_k) of a generated quantum circuit for a given state vector and a specific number of qubits ($n-$ qubits), we trace out all qubits except for the k^{th} qubit, where k ranges over 0, 1, 2, 3. This results in four 2×2 reduced density matrices, one for each qubit. These density matrices are computed as the squared absolute value of the trace of the partial transpose of the state vector with respect to the k^{th} qubit. An example of a

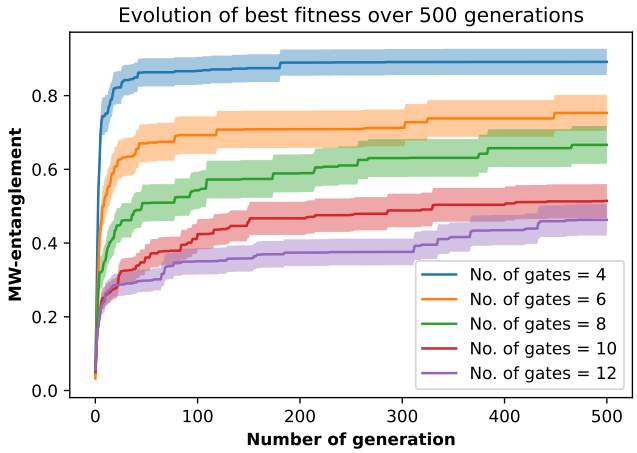


Figure 5.5. Comparison of the best fitness generated for five different numbers of gate sets in a four-qubit circuit using 10% mutation, 20 chromosomes. The results are averaged with 50 runs over 500 generations, and the error bars represent the standard error of the mean best fitness.

state vector and the corresponding reduced density matrices obtained for a four-qubit system is given by C and Equation (5.6), respectively. The use of reduced density matrices to analyze the properties of entangled states in complex quantum systems is discussed in detail in Section. 3.4.

$$\mathbf{C} = \begin{cases} \text{State vector} = (-0.115 + 0.015i)\,|0000\rangle + (-0.187 - 0.181i)\,|0001\rangle + \\ (0.0902 + 0.1135i)\,|0010\rangle + (0.3723 + 0.0052i)\,|0011\rangle + \\ (0.173 + 0.0870i)\,|0100\rangle + (-0.024 + 0.129i)\,|0101\rangle + \\ (0.146 + 0.2812i)\,|0110\rangle + (-0.007 + 0.112i)\,|0111\rangle + \\ (-0.116 + 0.0027i)\,|1000\rangle + (0.2256 + 0.333i)\,|1001\rangle + \\ (-0.070 + 0.064i)\,|1010\rangle + (-0.1037 + 0.280i)\,|1011\rangle + \\ (-0.191 + 0.136i)\,|1100\rangle + (0.039 - 0.284i)\,|1101\rangle + \\ (0.24917 + 0.103i)\,|1110\rangle + (-0.267 + 0.164i)\,|1111\rangle\,, \end{cases} \tag{5.5}$$

$$\mathbf{Matrices} = \begin{cases} \text{Matrices} = k_0 = \begin{bmatrix} 0.19 + 0.i & -0.079 - 0.11i \\ -0.0790 + 0.1031i & 0.357 + 0.i \end{bmatrix}, \\ k_1 = \begin{bmatrix} 0.2693 + 0.i & -0.022 + 0.048i \\ -0.022 - 0.0485i & 0.2368 + 0.i \end{bmatrix}, \\ k_2 = \begin{bmatrix} 0.207 + 0.i & -0.026 - 0.062i \\ -0.025 + 0.0631i & 0.306 + 0.i \end{bmatrix}, \\ k_3 = \begin{bmatrix} 0.117 + 0.i & -0.0168 - 0.110i \\ -0.0164 + 0.110i & 0.46 + 0.i \end{bmatrix}. \end{cases} \tag{5.6}$$

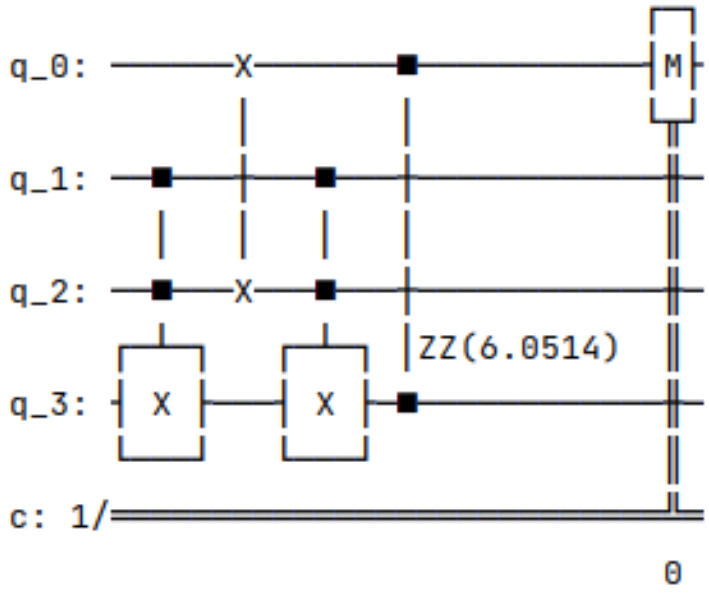


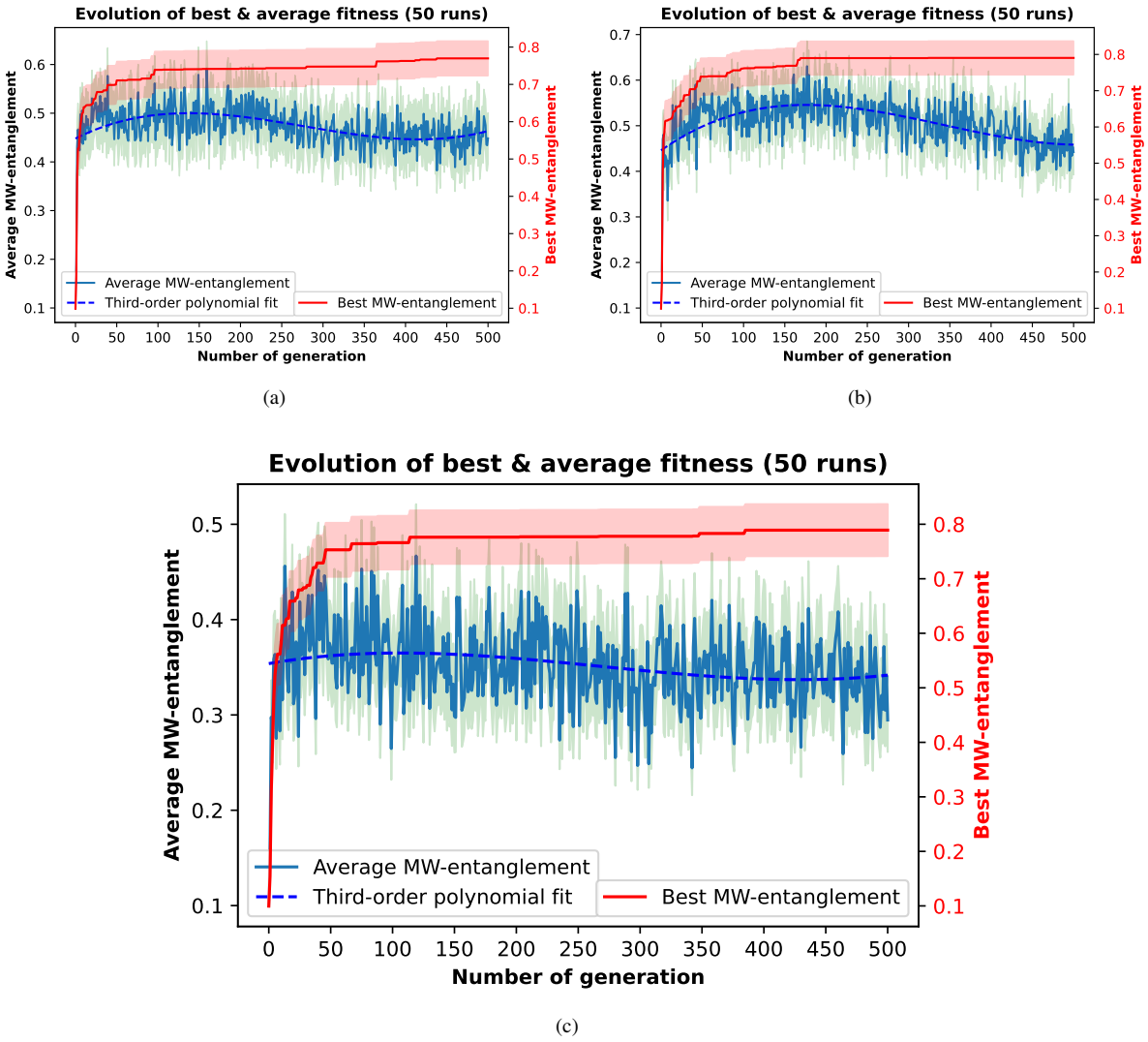
Figure 5.6. Evolutionary generation of four-gate four-qubit circuits with MW-entanglement fitness scores using a 10% mutation probability with a fitness score of 0.999.

5.3 Evolutionary algorithm for five-qubit quantum circuit design

5.3.1 Tuning evolutionary parameters for five-qubit circuit design: Mutation rate and gate number selection

Figure 5.7 displays the results of a study that investigated the effectiveness of an evolutionary algorithm in generating a five-qubit quantum circuit with five gates. The figure includes the average fitness, third-order polynomial fit to the average fitness, and the best fitness values with its standard error. The study explored the impact of different mutation rates (3%, 5%, and 15%) on the generated circuits' average and best fitness scores over 500 generations. The best fitness scores obtained for all mutation rates were close to 0.8, with an expected target fitness value of one, indicating a high degree of entanglement between the qubits. The third-order polynomial fit to the average fitness shows a smooth curve that approximates the general trend of the data.

The results of the study indicate that a mutation rate of 3% was the most effective in optimizing the quantum circuit, as shown in Figure 5.7a, which displays the best and average fitness values obtained. The study also found that increasing the mutation rate beyond 5% decreased the average fitness value, indicating that excessive randomness in the optimization process can negatively impact circuit performance. However, it was observed that the average of the best fitness value was not affected by a higher mutation rate.



(a) (b)

(c)

Figure 5.7. Evolutionary optimization of five-qubit quantum circuits with five gates using the Meyer-Wallach entanglement measure as the fitness function. The plot shows the mean fitness (green line) and its shaded standard error, as well as the mean of the best fitness (red line) and its shaded standard error, against the number of generations for different mutation percentages in the evolutionary algorithm: (a) 3%, (b) 5%, and (c) 15%. The blue dashed line represents the third-order polynomial fit to the mean fitness.

Our study underscores the significance of choosing an appropriate mutation rate to optimize quantum circuits effectively. In order to identify the optimal mutation rate, we analyzed the likelihood of replacing at least one individual in the population, which comprised 15 integer lists (each list representing three integer gates for each of the five qubits). Our investigation revealed that the probability of this event stood at 6.67%. Consequently, we deduced that a mutation rate of 5% (approximating 6.67%) would be an ideal choice, as it would ensure the replacement of at least one individual in the population for each generation.

In the second step of our investigation, we kept the mutation rate constant at 5% (optimal to replace at least one individual in each generation). We varied the number of gates to explore the entanglement capacity of the generated quantum circuit. As shown in Figure 5.8, the results

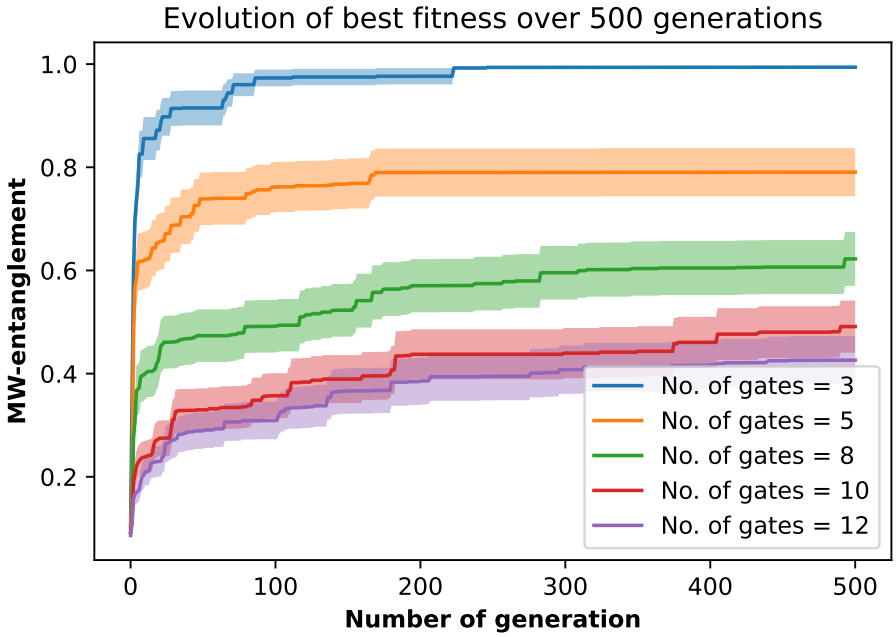


Figure 5.8. Comparison of the best fitness generated for five different numbers of gate sets in a 5-qubit circuit using 5% mutation, 20 chromosomes, and 50 runs over 500 generations. The results are averaged with 50 runs over 500 generations, and the error bars represent the standard error of the mean best fitness.

indicate that the fitness score is maximum or close to one for the three-gate five-qubit circuit. However, as the number of gates increases, the fitness scores decrease continuously. This finding suggests that increasing the number of gates in the quantum circuit (i.e., increasing the circuit depth) reduces the entanglement capability of the generated quantum circuits. The study suggests that a simpler quantum circuit with fewer gates may have better entanglement capability than a more complex circuit with more gates.

5.3.2 Visualizing best-fit five-qubit circuits with state vectors and reduced density matrices

The best-generated five-qubit quantum circuit with five gates, shown in Figure 5.9a with a fitness score of 0.999, provides a concrete example of the potential of evolutionary algorithms for designing highly entangled quantum circuits. Despite its simplicity, with only five gates, the circuit achieves a remarkably high degree of measure of entanglement between the qubits. This high degree of entanglement makes it useful for various applications in quantum computing and quantum information processing, such as quantum teleportation, quantum cryptography, and quantum error correction. This visualization highlights the power of evolutionary algorithms in optimizing quantum circuits for specific tasks, such as generating high-entanglement states. Also, the state vector and the reduced density matrix for one of the five-gate five-qubit circuits among 500 generations are given by D and Equation 5.8 respectively.

$$
\mathbf{D} = \begin{cases}
\text{State vector} = -0.058 + 0.078i\,|00000\rangle + 0.1044 - 0.096i\,|00001\rangle + \\
0.0106 - 0.117i\,|00010\rangle + 0.1254 + 0.116178i\,|00011\rangle + \\
-0.04338 + 0.1052i\,|00100\rangle + 0.049 - 0.1037i\,|00101\rangle + \\
-0.1251 + 0.09i\,|00110\rangle + -0.057 - 0.2138i\,|00111\rangle + \\
0.04810 - 0.117i\,|01000\rangle + -0.04 + 0.1409i\,|01001\rangle + \\
0.1062 + 0.0363i\,|01010\rangle + -0.236 - 0.0192i\,|01011\rangle + \\
-0.12 - 0.044i\,|01100\rangle + -0.07689 - 0.0441i\,|01101\rangle + \\
-0.26996 + 0.3182i\,|01110\rangle + -0.1467 - 0.07i\,|01111\rangle + \\
-0.014209 - 0.01026i\,|10000\rangle + 0.045 - 0.041i\,|10001\rangle + \\
-0.00403 + 0.0663i\,|10010\rangle + 0.1870 - 0.1880i\,|10011\rangle + \\
-0.0376 - 0.0229i\,|10100\rangle + 0.1329 + 0.1326i\,|10101\rangle + \\
-0.0165 + 0.2232i\,|10110\rangle + 0.0215 - 0.2594i\,|10111\rangle + \\
-0.170219 - 0.1936i\,|11000\rangle + 0.0335 - 0.0385i\,|11001\rangle + \\
-0.0058 + 0.101i\,|11010\rangle + -0.104 + 0.2592i\,|11011\rangle + \\
0.0995 + 0.0981i\,|11100\rangle + 0.111 - 0.2246i\,|11101\rangle + \\
-0.07112 - 0.1190i\,|11110\rangle + 0.0146 + 0.0084i\,|11111\rangle,
\end{cases} \tag{5.7}
$$

$$
\textbf{Matrices} = \begin{cases}
k_0 = \begin{bmatrix} 0.262617 + 0.i & 0.06638 + 0.04033i \\ 0.06638 - 0.040376i & 0.249531 + 0.i \end{bmatrix}, \\
k_1 = \begin{bmatrix} 0.171939 + 0. & -0.077024 - 0.02020i \\ -0.0772 + 0.02028i & 0.3533 + 0.i \end{bmatrix}, \\
k_2 = \begin{bmatrix} 0.174949 + 0.i & 0.03162114 + 0.01701i \\ 0.03164 - 0.017061i & 0.34150343 + 0.i \end{bmatrix}, \\
k_3 = \begin{bmatrix} 0.098702 + 0.i & -0.0016 + 0.042269i \\ -0.0016 - 0.042219 & 0.475956 + 0.i \end{bmatrix}, \\
k_4 = \begin{bmatrix} 0.233687 + 0.i & -0.13582 - 0.09953i \\ [-0.13512 + 0.09953i & 0.327048 + 0.i \end{bmatrix}.
\end{cases} \tag{5.8}
$$

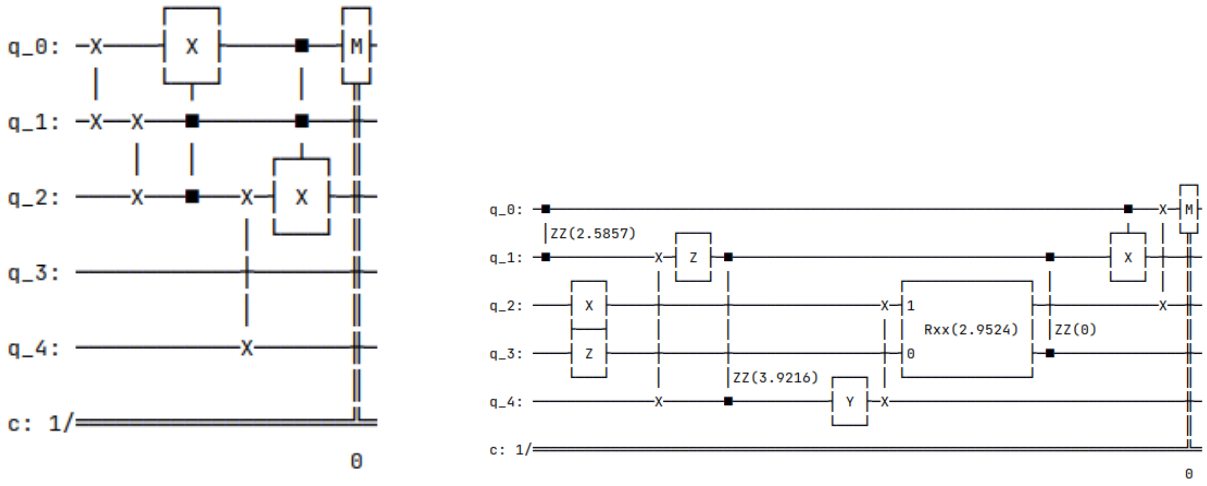
Figure 5.9b is a 12 gates five-qubit quantum circuit with a fitness score of 0.1999. Also, the state vector and the corresponding density matrix of one of the twelve-gate five-qubit circuits among 500 generations are given by E and Equation 5.10 respectively. Increasing the circuit depth, i.e., the number of gates in a circuit, increases the complexity of the quantum circuit and hence decreases the entanglement capability of the circuit. The lower fitness score of the

twelve-gate circuit is evidence of this fact. However, a deeper circuit may be useful for certain quantum algorithms that require a specific gate sequence. Therefore, evolutionary algorithms provide an effective tool for exploring and optimizing the design space of quantum circuits to accomplish certain quantum computational tasks.

$$
\mathbf{E} = \begin{cases}
\text{State vector} = 0.0016 - 0.063i\,|00000\rangle + -0.0461 + 0.195i\,|00001\rangle + \\
\quad - 0.00427 + 0.246i\,|00010\rangle + -0.191 - 0.0738i\,|00011\rangle + \\
\quad - 0.1282 + 0.19i\,|00100\rangle + -0.043 + 0.0471i\,|00101\rangle + \\
\quad 0.01324 - 0.0028i\,|00110\rangle + 0.1088 - 0.117i\,|00111\rangle + \\
\quad 0.08242 + 0.4377i\,|01000\rangle + 0.1089 + 0.097i\,|01001\rangle + \\
\quad 0.14634 - 0.023i\,|01010\rangle + -0.124 - 0.2418i\,|01011\rangle + \\
\quad - 0.14980 - 0.197i\,|01100\rangle + 0.0791 + 0.0092i\,|01101\rangle + \\
\quad 0.149754 + 0.100i\,|01110\rangle + -0.052 + 0.013i\,|01111\rangle + \\
\quad 0.1720 + 0.0166i\,|10000\rangle + -0.115 - 0.065i\,|10001\rangle + \\
\quad 0.0588 + 0.132i\,|10010\rangle + -0.099 + 0.1064i\,|10011\rangle + \\
\quad - 0.0179 - 0.0713i\,|10100\rangle + 0.006 - 0.034i\,|10101\rangle + \\
\quad 0.1627 + 0.1133i\,|10110\rangle + 0.0613 - 0.041i\,|10111\rangle + \\
\quad 0.0063 - 0.09i\,|11000\rangle + 0.1241 - 0.02077i\,|11001\rangle + \\
\quad - 0.2095 - 0.153i\,|11010\rangle + 0.052 - 0.024i\,|11011\rangle + \\
\quad 0.059 - 0.077i\,|11100\rangle + -0.126 - 0.133i\,|11101\rangle + \\
\quad 0.024 - 0.055i\,|11110\rangle + -0.237 + 0.106i\,|11111\rangle\,,
\end{cases} \tag{5.9}
$$

$$
\textbf{Matrices} = \begin{cases}
k_0 = \begin{bmatrix} 0.422 + 0.i & -0.014 + 0.0215i \\ -0.0144 - 0.02127i & 0.1236 + 0.i \end{bmatrix}, \\
k_1 = \begin{bmatrix} 0.14368 + 0.i & -0.07121 - 0.0253i \\ -0.0711 + 0.0253i & 0.40072 + 0.i \end{bmatrix}, \\
k_2 = \begin{bmatrix} 0.43113 + 0.i & -0.089 - 0.04495i \\ -0.089 + 0.0442i & 0.13307 + 0.i \end{bmatrix}, \\
k_3 = \begin{bmatrix} 0.27337 + 0.i & -0.062 + 0.08017i \\ -0.06662 - 0.08017i & 0.2488 + 0.i \end{bmatrix}, \\
k_4 = \begin{bmatrix} 0.38103 + 0.i & -0.01147 + 0.01227i \\ -0.01147 - 0.01219i & 0.14695 + 0.i \end{bmatrix}.
\end{cases} \tag{5.10}
$$

Equation (5.8) and (5.10) present the five reduced density matrices. After obtaining the



(a) (b)

Figure 5.9. Evolutionary generation of five-qubit circuits with MW-entanglement fitness scores using a 5% mutation probability. (a) Five-gate circuit with a fitness score of 0.999 and (b) twelve-gate circuit with a fitness score of 0.1999.

reduced density matrix for each qubit, our implementation computes the entanglement of the circuit using the Mayor-Wallach entanglement measure. It is vital to emphasize that although the reduced density matrix of a subsystem can offer valuable insights into the degree of entanglement among the qubits within that subsystem, it may not fully capture the entanglement of the entire system, as discussed in earlier sections. Consequently, tracing out all qubits to compute the entanglement measure of a quantum circuit is critical for accurately assessing the circuit's entanglement. Evaluating the entanglement of a multi-qubit circuit requires an examination of the joint quantum state of all qubits rather than merely focusing on the reduced density matrix of individual qubits. In Section 3.4, we comprehensively discuss the usefulness of reduced density matrices as an analytical tool for investigating the properties of entangled states in complex quantum systems.

Chapter 6

Summary and outlook

6.1 Summary

The thesis presents two distinct frameworks for generating quantum circuits using evolutionary algorithms. In the first part, we propose a framework for generating quantum circuits that reproduce specific deterministic and stochastic cellular automata rules. An evolutionary algorithm based on mutations has been used to optimize the quantum gate types and their connectivity. We demonstrated the effectiveness of this framework by evolving quantum circuits for various types of cellular automata rules, ranging from deterministic to stochastic updates. We showed that the framework could successfully generate quantum circuits with good fitness scores for various types of cellular automata rules. We also observed some interesting phenomena during the experiments, such as the ups and downs of fitness scores for parents and the approximately common fitness score for different mutation rates. Overall, this framework provides a promising direction for generating quantum circuits using CA rules, and further investigations can be done to improve its performance and explore its potential.

The second part presents an evolutionary algorithm-based approach to designing quantum circuits that maximize entanglement using the Mayor-Wallach entanglement measure as the fitness function. The results show that the algorithm can effectively optimize quantum circuits for entanglement generation with different numbers of qubits and gates. Our findings indicate that the algorithm is proficient in optimizing quantum circuits for entanglement generation across varying numbers of qubits and gates. Specifically, for three-qubit circuits, the ideal mutation rate was determined to be 10%, while for five-qubit circuits, it was 5%. Moreover, our research highlights circuit depth's critical role in establishing a quantum circuit's entanglement capability. As circuit depth increases, signifying a higher number of gates in

the circuit, the complexity of the quantum circuit escalates, subsequently diminishing the entanglement capability. These observations showcase the potential of evolutionary algorithms as valuable tools for quantum circuit design and optimization, offering insights to inform future investigations in this domain. Nonetheless, further research is required to assess the robustness of circuits generated through evolutionary algorithms and to pinpoint techniques for enhancing their resistance to noise. Additionally, distinct gate sets might exhibit varying degrees of expressivity and could yield different outcomes in terms of entanglement and fitness scores for the resulting circuits. Consequently, examining diverse gate sets along with evolutionary parameters constitutes a vital avenue for future research.

6.2 Outlook: Future perspectives and research opportunities

For the first framework, additional genetic operators, such as crossover, could be explored to optimize the performance of the framework further. The use of more gates to build more complex quantum circuits could increase the pool of potential solutions. Further comparison and evaluation of different fitness functions could also be conducted. Additionally, the framework could be extended to address problems in other domains beyond CA.

For the second framework, further investigation could be conducted to explore the impact of different genetic operators, such as crossover, on the performance of the algorithm. The promising results obtained in this study open up exciting opportunities for future research on quantum circuit design using evolutionary algorithms. One future direction is to explore the entanglement capacity of larger quantum circuits with more qubits using our proposed approach. Investigating the optimal mutation rates and the number of gates for these circuits would help in understanding the scalability and performance limitations of the approach.

Future research could explore the use of alternative fitness functions to optimize quantum circuits for specific tasks, such as error correction or state preparation. The findings from such experiments could contribute to the development of hybrid optimization methods that combine different entanglement measures to enhance the entanglement capacity of quantum circuits. For instance, the Meyer-Wallach entanglement measure could be employed in conjunction with the von Neumann entropy [110] and Schmidt [111, 112] measures to optimize quantum circuits across multiple measures. The von Neumann entanglement entropy, defined in Appendix C, has been tested for a pure entangled, pure product, and mixed states. Detailed information regarding its implementation and output for various states can be found in Appendix C. The next step is to combine these functions as a fitness function of evolutionary algorithms, similar to the previous usage of the Meyer-Wallach entanglement measure. This approach may lead to improved performance compared to using a single entanglement measure, as different measures capture distinct aspects of entanglement in quantum systems. The von Neumann entropy is a

widely used entanglement measure that quantifies the degree of entanglement between two subsystems [110]. It is calculated as the negative trace of the reduced density matrix of one subsystem. The Schmidt measure assesses the entanglement degree between two subsystems within a larger system and is defined as the sum of the squared Schmidt coefficients of the state vector [111, 112]. Additionally, more complex quantum circuits with a higher number of qubits and gates could be explored, which would require the development of more advanced evolutionary algorithms. Finally, hybrid approaches combining evolutionary algorithms with other optimization techniques, such as gradient descent, could be explored to enhance the optimization process further. Overall, the results presented in this chapter demonstrate the potential of evolutionary algorithms for quantum circuit design and provide a foundation for further research in this field.

6.3 Leveraging findings: Utilizing von Neumann entropy as the fitness function for future research

As previously mentioned, we implemented von Neumann entropy in our evolutionary algorithm, consisting of a three-qubit system using ten gates and a mutation rate of 10%. The process was conducted for 50 iterations over 500 generations, with a total of 20 chromosomes. The output is displayed in Figure 6.1.

Our investigation focused on the implementation of von Neumann entropy in a three-qubit system. The theoretical maximum entropy for such a system is 2.999, as calculated and demonstrated in Appendix C. Nevertheless, when implementing von Neumann entropy entanglement as a fitness function in the evolutionary algorithm, we obtained a maximum entropy value of 1.05557. This result has significant implications for the entanglement within the system. The quantification of entanglement can be achieved through the computation of von Neumann entropy for the reduced density matrix of any given subsystem. The presence of entanglement between two subsystems is indicated by a non-zero von Neumann entropy value for the reduced density matrix [113, 114]. As a result, our findings indicate that the implementation of von Neumann entropy in the three-qubit system successfully detected the presence of entanglement within the evolved three-qubit quantum circuit. This is consistent with the understanding that the entropy of entanglement is the von Neumann entropy of the reduced density matrix for any of the subsystems, and if it is non-zero, i.e., the subsystem is in a mixed state, it signifies that the two subsystems are entangled [114]. This discovery offers valuable insights for future research in the area of entanglement measurement using evolutionary algorithms.

In summary, our implementation of von Neumann entropy in the three-qubit system successfully identified entanglement within the system, as evidenced by the non-zero entropy

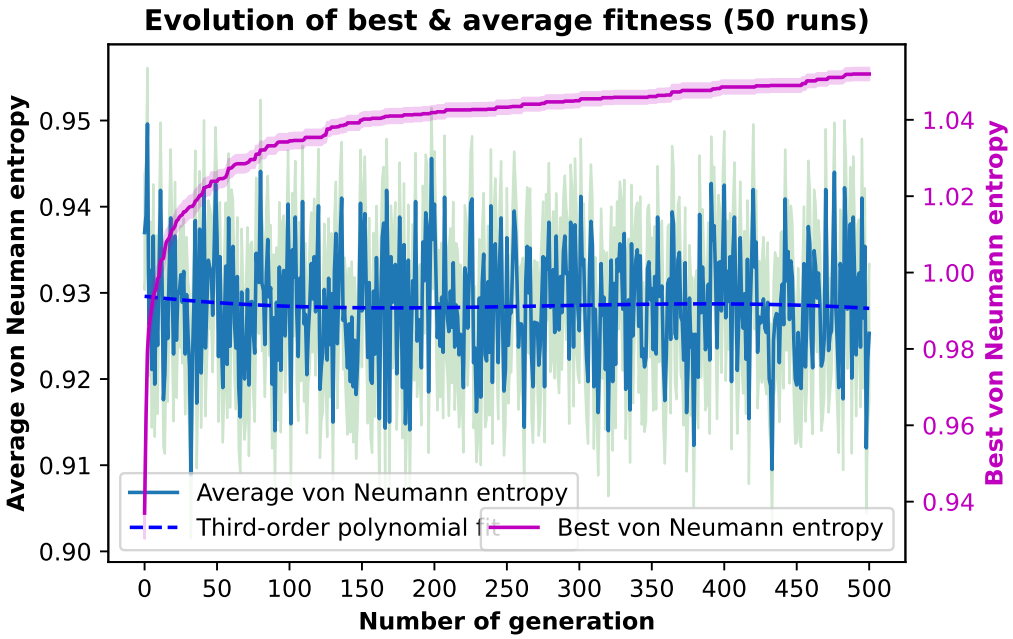


Figure 6.1. Evolutionary optimization of three-qubit quantum circuits with 10% mutation probability and ten gates, using von Neumann entanglement entropy as the fitness function. The plot shows the mean fitness (green line) and its shaded standard error, as well as the mean of the best fitness (magenta line) and its shaded standard error, against the number of generations. The blue dashed line represents the third-order polynomial fit to the mean fitness. The maximum fitness score obtained was 1.05557

values of the reduced density matrix. Further optimization of the evolutionary algorithm parameters may enable the achievement of the maximum possible entropy value for three-qubit quantum circuits. These findings hold significant implications for future research in the fields of quantum computing and entanglement.

6.4 Challenges in implementing evolutionary algorithms for quantum systems

"Evolutionary algorithms rely on problem-specific stochastic processes, and as a result, evaluating their computational performance must account for the computational effort required to compute the fitness function. Consequently, performance comparisons in evolutionary computation involve determining the number of fitness function evaluations needed to achieve a certain level of average accuracy across a specified number of runs for a given problem" [11]. The difficulty of accessing the required computational performance to compute the fitness functions poses practical and fundamental challenges. Practical challenges are related to the size of the parameter space and the amount of data generated. Depending on the parameter selection, data is generated in the simulation, which requires running simulations for various numbers of parameters at each optimization step. For each experimental setup, multiple runs are needed for each set of parameters, increasing computation time. Fundamental challenges arise because

the quantum system produces inherently non-deterministic outputs, with variations in output for each generation. Running simulations multiple times is necessary to mitigate statistical fluctuations, although this further increases computation time.

Another crucial aspect to consider when assessing the computational performance of quantum algorithms is the depth of the associated quantum circuits [11]. In fact, creating shallow-depth quantum circuits is vital for implementing quantum algorithms using current technology. This is because qubits are prone to spontaneous state changes due to quantum decoherence, resulting in a limited operational time frame. Quantum decoherence is the process that causes a quantum system to lose its coherence and transition from a state of superposition or entanglement to a mixture of classical states [115].

During this research project, we encountered challenges that were consistent throughout all stages. We successfully simulated the evolutionary algorithm on an IBM quantum simulator. However, increasing the number of parameters (such as the number of runs, generations, gate types, and mutation probability) would require more time and a more powerful computational device than a personal computer. This experience provided valuable lessons for the later stages of the project. A potential risk during the subsequent stages was the proper definition and implementation of the measure of entanglement in the algorithms to generate it as a function of the number of generations. This was studied and implemented initially on a parametrized quantum circuit, and the obtained results were compared with those from the evolutionary algorithms.

Implementing an evolutionary algorithm on a quantum system requires careful consideration of the parameter space and data generated and proper definition and implementation of measures such as entanglement. However, with careful planning and execution, these challenges can be overcome, and valuable insights can be gained through the use of evolutionary algorithms on quantum systems.

Bibliography

[1] Shailendra Bhandari et al. 'Evolving Quantum Circuits to Implement Stochastic and Deterministic Cellular Automata Rules'. In: *International Conference on Cellular Automata for Research and Industry*. Springer. 2022, pp. 119–129. DOI: https://doi.org/10.1007/978-3-031-14926-9_11.

[2] Shailendra Bhandari et al. *Evolutionary Computing to Solve Optimal Entangle States in Quantum Circuits*. EasyChair Preprint no. 10135. 2023. URL: https://easychair.org/publications/preprint/bgdS.

[3] Shailendra Bhandari et al. 'Building goal-specific quantum circuits with evolutionary algorithms'. Will be submitted for publication in the International Journal of Unconventional Computing. 2023.

[4] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[5] Peter W Shor. 'Scheme for reducing decoherence in quantum computer memory'. In: *Physical Review A* 52.4 (1995), p. 2493.

[6] Seth Lloyd and Christian Weedbrook. 'Quantum machine learning'. In: *Nature* 549.7671 (2018), pp. 195–202.

[7] *Quantum Computing: What, Who, How and When? (2018, June 15). Science Museum Blog.* https://blog.sciencemuseum.org.uk/quantum-computing-what-who-how-and-when/. Accessed: 2021-11-17.

[8] Jacob Biamonte et al. 'Quantum machine learning'. In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474. arXiv: 1611.09347.

[9] Maria Schuld, Ilya Sinayskiy and Francesco Petruccione. 'An introduction to quantum machine learning'. In: *Contemporary Physics* 56.2 (Oct. 2014), pp. 172–185. ISSN: 1366-5812. DOI: 10.1080/00107514.2014.964942.

[10] Francesco Tacchino et al. 'An artificial neuron implemented on an actual quantum processor'. In: *npj Quantum Information* 5.1 (2019), pp. 1–8. ISSN: 20566387. DOI: 10.1038/s41534-019-0140-4.

[11] Giovanni Acampora and Autilia Vitiello. 'Implementing evolutionary optimization on actual quantum processors'. In: *Information Sciences* 575 (2021), pp. 542–562. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.06.049.

[12] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[13] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[14] E. Alba. *Parallel metaheuristics: a new class of algorithms*. John Wiley & Sons, 2005.

[15] J. Romero, J. P. Olson and A. Aspuru-Guzik. 'Quantum autoencoders for efficient compression of quantum data'. In: *Quantum Science and Technology* 3.3 (2018), p. 034004.

[16] R. Biswas and A. Manimegalai. 'Quantum-inspired evolutionary algorithm and its application'. In: *International Journal of Computer Applications* 179.48 (2019), pp. 26–31.

[17] M. Veldstra, K. Vu and F. Groen. 'A new evolutionary algorithm for quantum circuit optimization'. In: *Quantum Information Processing* 16.11 (2017), p. 266.

[18] M. A. Serrano-Andrés. 'Quantum-inspired evolutionary algorithm for chemistry and materials'. In: *The Journal of Chemical Physics* 151.1 (2019), p. 014104.

[19] Victor Manuel Arriola-Rios, Carlos A Cruz-Ramos and Gustavo Arroyo-Figueroa. 'Bio-inspired optimization for quantum computing: a review'. In: *arXiv preprint arXiv:2010.06437* (2020).

[20] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2002.

[21] Farzan Jazaeri et al. *A Review on Quantum Computing: Qubits, Cryogenic Electronics and Cryogenic MOSFET Physics*. 2019. eprint: 1908.02656.

[22] M.A. Nielsen and I.L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[23] Phillip Kaye, Raymond Laflamme, Michele Mosca et al. *An introduction to quantum computing*. Oxford University Press on Demand, 2007.

[24] T Qiskit Team. *Single qubit gates. Qiskit*. https://qiskit.org/textbook/ch-states/single-qubit-gates.html. Accessed: 2021-11-24.

[25] David Deutsch and Richard Jozsa. 'Rapid solution of problems by quantum computation'. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[26] Andrew Steane. 'Multiple particle interference and quantum error correction'. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 452.1954 (1996), pp. 2551–2577.

[27] Robert S. Sutor. *Dancing with Qubits*. Packt Publishing, 2019. ISBN: 978-1-83882-525-6.

[28] C. H. Bennett et al. 'Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels'. In: *Physical Review Letters* 70.13 (1993), pp. 1895–1899. DOI: 10.1103/PhysRevLett.70.1895.

[29] G. Benenti, Giulio Casati and Giuliano Strini. *Principles of Quantum Computation and Information: Volume II: Basic Tools and Special Topics*. World Scientific, Mar. 2007, pp. 1–680. DOI: 10.1142/5838.

[30] Tommaso Toffoli. 'Reversible computing'. In: *International Journal of Theoretical Physics* 21.3 (1982), pp. 165–171.

[31] Kuk-Hyun Han and Jong-Hwan Kim. 'Quantum-inspired evolutionary algorithm for a class of combinatorial optimization'. In: *Evolutionary Computation, IEEE Transactions on* 6 (Jan. 2003), pp. 580–593. DOI: 10.1109/TEVC.2002.804320.

[32] Bertrand Wong. 'ON QUANTUM ENTANGLEMENT'. In: *International Journal of Automatic Control System* 5 (Dec. 2019), pp. 1–7.

[33] Guifré Vidal. 'Efficient Classical Simulation of Slightly Entangled Quantum Computations'. In: *Phys. Rev. Lett.* 91 (14 Oct. 2003), p. 147902. DOI: 10.1103/PhysRevLett.91.147902.

[34] F. Verstraete, V. Murg and J.I. Cirac. 'Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems'. In: *Advances in Physics* 57.2 (Mar. 2008), pp. 143–224. ISSN: 1460-6976. DOI: 10.1080/14789940801912366.

[35] Charles H. Bennett and Stephen J. Wiesner. 'Communication via one-and two-particle operators on Einstein-Podolsky-Rosen states'. In: *Phys. Rev. Lett.* 69 (20 Nov. 1992), pp. 2881–2884. DOI: 10.1103/PhysRevLett.69.2881.

[36] Gary J. Mooney, Charles D. Hill and Lloyd C.L. Hollenberg. 'Entanglement in a 20-Qubit Superconducting Quantum Computer'. In: *Scientific Reports* 9.1 (2019), pp. 1–8. ISSN: 20452322. DOI: 10.1038/s41598-019-49805-7.

[37] Sukin Sim, Peter Johnson and Alán Aspuru-Guzik. 'Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms'. In: (May 2019). URL: https://arxiv.org/pdf/1905.10876.pdf.

[38] Karol Życzkowski and Hans-Jürgen Sommers. 'Average fidelity between random quantum states'. In: *Phys. Rev. A* 71 (3 Mar. 2005), p. 032313. DOI: 10.1103/PhysRevA.71.032313.

[39] Thomas Hubregtsen et al. 'Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability'. In: *Quantum Machine Intelligence* 3.1 (2021). DOI: 10.1007/s42484-021-00038-w.

[40] David A Meyer and Nolan R Wallach. 'Global entanglement in multiparticle systems'. In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4273–4278. DOI: 10.1063/1.1497700.

[41] Mateus Ara'ujo et al. 'Witnessing genuine multipartite entanglement with binary measurements'. In: *Physical Review A* 99.2 (2019), p. 022104.

[42] Gavin K. Brennen. 'An observable measure of entanglement for pure states of multi-qubit systems'. In: (2003). DOI: 10.48550/ARXIV.QUANT-PH/0305094.

[43] Alba Cervera-Lierta, José Ignacio Latorre and Dardo Goyeneche. 'Quantum circuits for maximally entangled states'. In: *Phys. Rev. A* 100 (2 Aug. 2019), p. 022342. DOI: 10.1103/PhysRevA.100.022342.

[44] Chitra Shukla, Anindita Banerjee and Anirban Pathak. 'Protocols and quantum circuits for implementing entanglement concentration in cat state, GHZ-like state and 9 families of 4-qubit entangled states'. In: *Quantum Information Processing* 14 (Mar. 2014). DOI: 10.1007/s11128-015-0948-6.

[45] Nicolai Friis et al. 'Entanglement certification from theory to experiment'. In: *Nature Reviews Physics* 1.1 (Dec. 2018), pp. 72–87. ISSN: 2522-5820. DOI: 10.1038/s42254-018-0003-5.

[46] Jiheon Seong and Joonwoo Bae. *Detecting Entanglement Generating Circuits in Cloud-Based Quantum Computing*. 2021. arXiv: 2110.10528 [quant-ph].

[47] Carmen G Almudever et al. 'The engineering challenges in quantum computing'. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE. 2017, pp. 836–845. URL: 978-3-9815370-9-3/DATE17/?c%202017%20EDAA.

[48] B. P. Lanyon et al. 'Experimental Quantum Computing without Entanglement'. In: *Physical Review Letters* 101.20 (Nov. 2008). ISSN: 1079-7114. DOI: 10.1103/physrevlett.101.200501.

[49] Richard Jozsa and Noah Linden. 'On the role of entanglement in quantum-computational speed-up'. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 459.2036 (Aug. 2003), pp. 2011–2032. ISSN: 1471-2946. DOI: 10.1098/rspa.2002.1097.

[50] Gilson A. Giraldi, Renato Portugal and Ricardo N. Thess. 'Genetic Algorithms and Quantum Computation'. In: *CoRR* cs.NE/0403003 (2004). URL: http://arxiv.org/abs/cs/0403003.

[51] J. H. Holland. 'Genetic algorithms'. In: *Scholarpedia* 7.12 (2012). revision #128222, p. 1482. DOI: 10.4249/scholarpedia.1482.

[52] A E Eiben and J E Smith. *Introduction to evolutionary computing*. Springer, 2003.

[53] M. Lukac and M. Perkowski. 'Evolving quantum circuits using genetic algorithm'. In: *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*. 2002, pp. 177–185. DOI: 10.1109/EH.2002.1029883.

[54] André Leier. 'Evolution of Quantum Algorithms using Genetic Programming'. ENG. PhD thesis. Germany: Dortmund University, July 2004. URL: http://hdl.handle.net/2003/2745.

[55] David Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass: Addison-Wesley Pub. Co, 1989. ISBN: 0201157675.

[56] Darrell Whitley. 'A genetic algorithm tutorial'. In: *Statistics and computing*. Vol. 4. 2. Springer, 1994, pp. 65–85.

[57] Ajit Narayanan and Mark Moore. 'Quantum-inspired genetic algorithms'. In: *Proceedings of IEEE International Conference on Evolutionary Computation* (1996), pp. 61–66. DOI: 10.1109/ICEC.1996.542334.

[58] Kuk-Hyun Han and Jong-Hwan Kim. 'Genetic quantum algorithm and its application to combinatorial optimization problem'. In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*. Vol. 2. 2000, 1354–1360 vol.2. DOI: 10.1109/CEC.2000.870809.

[59] Kuk-Hyun Han and Jong-Hwan Kim. 'Quantum-inspired evolutionary algorithm for a class of combinatorial optimization'. In: *IEEE Transactions on Evolutionary Computation* 6.6 (2002), pp. 580–593. DOI: 10.1109/TEVC.2002.804320.

[60] Yan Wang et al. 'A novel quantum swarm evolutionary algorithm and its applications'. In: *Neurocomputing* 70 (Jan. 2007), pp. 633–640. DOI: 10.1016/j.neucom.2006.10.001.

[61] Mohadeseh Soleimanpour-moghadam and Hossein Nezamabadi-pour. 'An improved quantum behaved gravitational search algorithm'. In: *20th Iranian Conference on Electrical Engineering (ICEE2012)*. 2012, pp. 711–715. DOI: 10.1109/IranianCEE.2012.6292446.

[62] Rui Zhang, Zhiteng Wang and Hongjun Zhang. 'Quantum-Inspired Evolutionary Algorithm for Continuous Space Optimization Based on Multiple Chains Encoding Method of Quantum Bits'. In: *Mathematical Problems in Engineering* 2014 (2014). ISSN: 15635147. DOI: 10.1155/2014/620325.

[63] Fei Li et al. 'Quantum Bacterial Foraging Optimization for Cognitive Radio Spectrum Allocation'. In: *KSII Transactions on Internet and Information Systems* 9.2 (Feb. 2015), pp. 564–582. DOI: 10.3837/tiis.2015.02.005.

[64] Peng Wang et al. 'Multi-scale quantum harmonic oscillator algorithm for global numerical optimization'. In: *Appl. Soft Comput.* 69 (2018), pp. 655–670.

[65] Lei Mu, Peng Wang and Gang Xin. 'Quantum-inspired algorithm with fitness landscape approximation in reduced dimensional spaces for numerical function optimization'. In: *Information Sciences* 527 (2020), pp. 253–278. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.03.035.

[66] Pronaya Prosun Das and Mozammel H. A. Khan. 'Solving Maximum Clique Problem using a novel Quantum-inspired Evolutionary Algorithm'. In: *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)* (2015), pp. 1–6.

[67] Xiaoyue Feng, Enrico Blanzieri and Yanchun Liang. 'Improved Quantum-Inspired Evolutionary Algorithm and Its Application to 3-SAT Problems'. In: *2008 International Conference on Computer Science and Software Engineering*. Vol. 1. 2008, pp. 333–336. DOI: 10.1109/CSSE.2008.1512.

[68] Berend Denkena et al. 'Quantum algorithms for process parallel flexible job shop scheduling'. In: *CIRP Journal of Manufacturing Science and Technology* 33 (2021), pp. 100–114. ISSN: 1755-5817. DOI: https://doi.org/10.1016/j.cirpj.2021.03.006.

[69] Bin-Bin Li and Ling Wang. 'A Hybrid Quantum-Inspired Genetic Algorithm for Multiobjective Flow Shop Scheduling'. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.3 (2007), pp. 576–591. DOI: 10.1109/TSMCB.2006.887946.

[70] R. S. Amal and J. Solomon Ivan. 'A quantum genetic algorithm for optimization problems on the Bloch sphere'. In: *Quantum Information Processing* 21.2 (2022), pp. 1–29. ISSN: 15731332. DOI: 10.1007/s11128-021-03368-7.

[71] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. 2nd. Springer Publishing Company, Incorporated, 2015. ISBN: 3662448734.

[72] A. W. Burks J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1967.

[73] Stephen Wolfram. 'Cellular automata as models of complexity'. In: *Nature (London)* 311.5985 (1984), pp. 419–424. ISSN: 0028-0836.

[74] Stephen Wolfram. 'Cellular automaton fluids 1: Basic theory'. In: *Journal of statistical physics* 45.3-4 (1986), pp. 471–526. ISSN: 0022-4715.

[75] Tommaso Toffoli. 'Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics'. In: *Physica D: Nonlinear Phenomena* 10.1 (1984), pp. 117–127. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(84)90254-9.

[76] M. Shabat, K. Nikolskiy and E. Shabat. 'Cellular automata as models of complex systems'. In: *Applied Mathematics and Computation* 302 (2017), pp. 18–34.

[77] M. Batty. *Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals*. MIT press, 2005.

[78] R.P. Feynman. 'Simulating physics with computers'. In: *International Journal of Theoretical Physics* 21 (1982), pp. 467–488.

[79] Detlev Buchholz. 'Current Trends in Axiomatic Quantum Field Theory'. In: June 1998 (2000), pp. 43–64. DOI: 10.1007/3-540-44482-3_4.

[80] Richard P. Feynman. 'Quantum Mechanical Computers'. In: *Foundations of Physics* 16.6 (1986), pp. 507–531. DOI: 10.1007/BF01886518.

[81] S. C. Benjamin. 'Schemes for parallel quantum computation without local control of qubits'. In: *Phys. Rev. A* 61 (2 Jan. 2000), p. 020301. DOI: 10.1103/PhysRevA.61.020301.

[82] Joseph Fitzsimons and Jason Twamley. 'Globally Controlled Quantum Wires for Perfect Qubit Transport, Mirroring, and Computing'. In: *Phys. Rev. Lett.* 97 (9 Sept. 2006), p. 090502. DOI: 10.1103/PhysRevLett.97.090502.

[83] Daniel Nagaj and Pawel Wocjan. 'Hamiltonian quantum cellular automata in one dimension'. In: *Phys. Rev. A* 78 (Sept. 2008). DOI: 10.1103/PhysRevA.78.032311.

[84] J. Twamley. 'Quantum-cellular-automata quantum computing with endohedral fullerenes'. In: *Phys. Rev. A* 67 (5 May 2003), p. 052318. DOI: 10.1103/PhysRevA.67.052318.

[85] Yaakov S. Weinstein and C. Stephen Hellberg. 'Quantum-cellular-automata pseudorandom maps'. In: *Physical Review A* 69 (2004), p. 062301.

[86] J. Watrous. 'On one-dimensional quantum cellular automata'. In: *Proceedings of IEEE 36th Annual Foundations of Computer Science*. 1995, pp. 528–537. DOI: 10.1109/SFCS.1995.492583.

[87] Stephen Wolfram. 'Statistical mechanics of cellular automata'. In: *Reviews of Modern Physics* 55.3 (1983), pp. 601–644.

[88] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[89] Stephen Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, 1986.

[90] Melanie Mitchell. *Complexity: A Guided Tour*. Oxford University Press, 2009.

[91] Andrew Ilachinski. *Cellular Automata: A Discrete Universe*. World Scientific, 2001. ISBN: 981-02-4623-4.

[92] Matthew Cook. 'Universality in Elementary Cellular Automata'. In: *Complex Systems* 15.1 (2004), pp. 1–40.

[93] Jan M Baetens, Wouter Van der Meeren and Bernard De Baets. 'On the Dynamics of Stochastic Elementary Cellular Automata.' In: *Journal of Cellular Automata* 12 (2016).

[94] Marcelo Zamith et al. 'A new stochastic cellular automata model for traffic flow simulation with drivers' behavior prediction'. In: *Journal of Computational Science* 9 (2015). Computational Science at the Gates of Nature, pp. 51–56. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2015.04.005.

[95] Hamid Izadkhah. 'P, NP, NP-Complete, and NP-Hard Problems'. In: *Problems on Algorithms*. Cham: Springer, 2022. DOI: 10.1007/978-3-031-17043-0_15.

[96] Paramita Basak Upama et al. 'Evolution of Quantum Computing: A Systematic Survey on the Use of Quantum Computing Tools'. In: *arXiv preprint arXiv:2204.01856* (2022).

[97] Anne Broadbent and Elham Kashefi. 'Parallelizing quantum circuits'. In: *Theoretical Computer Science* 410.26 (2009), pp. 2489–2510. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2008.12.046.

[98] IBM Newsroom. *IBM Quantum breaks the 100-qubit processor barrier*. Accessed: 2023-04-19. 2021. URL: https://research.ibm.com/blog/127-qubit-quantum-processor-eagle.

[99] IBM Quantum. *IBM Quantum*. Accessed: 2023-04-19. 2021. URL: https://quantum-computing.ibm.com/composer/docs/iqx/manage/systems/processors.

[100] John Preskill. 'Quantum Computing in the NISQ era and beyond'. In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[101] Abraham Asfaw et al. *Density Matrix*. Qiskit. May 2021. URL: https://qiskit.org/textbook/ch-quantum-hardware/density-matrix.html#4.-The-Reduced-Density-Matrix--.

[102] OsloMet - storbyuniversitetet. *Nordic Center for Sustainable and trustworthy AI Research (NordSTAR)*. URL: https://www.oslomet.no/en/nordstar.

[103] Fernando Martín et al. 'Kullback-Leibler Divergence-Based Differential Evolution Markov Chain Filter for Global Localization of Mobile Robots'. In: *Sensors* 15.9 (2015), pp. 23431–23458. ISSN: 1424-8220. DOI: 10.3390/s150923431.

[104] Simon M. Lucas and Vanessa Volz. 'Tile pattern KL-divergence for analysing and evolving game levels'. In: *Proceedings of the Genetic and Evolutionary Computation Conference* (July 2019). DOI: 10.1145/3321707.3321781.

[105] Sidney Pontes-Filho et al. 'A neuro-inspired general framework for the evolution of stochastic dynamical systems: Cellular automata, random Boolean networks and echo state networks towards criticality'. In: *Cognitive Neurodynamics* 14.5 (2020), pp. 657–674. ISSN: 18714099. DOI: 10.1007/s11571-020-09600-x.

[106] Leo Sünkel et al. *GA4QCO: Genetic Algorithm for Quantum Circuit Optimization*. 2023. DOI: 10.48550/ARXIV.2302.01303.

[107] Jonathon Brown, Mauro Paternostro and Alessandro Ferraro. 'Optimal quantum control via genetic algorithms for quantum state engineering in driven-resonator mediated networks'. In: *Quantum Science and Technology* 8.2 (Jan. 2023), p. 025004. DOI: 10.1088/2058-9565/acb2f2.

[108] Edgar Ballinas and Osvaldo Montiel. 'Hybrid quantum genetic algorithm with adaptive rotation angle for the 0-1 Knapsack problem in the IBM Qiskit simulator'. In: *Soft Computing* (2022). DOI: 10.1007/s00500-022-07460-7.

[109] Abraham Asfaw et.al. *Learn Quantum Computation using Qiskit*. Qiskit, 2020. URL: https://learn.qiskit.org/course/basics/single-systems.

[110] T Mendes-Santos et al. 'Measuring von Neumann entanglement entropies without wave functions'. In: *New Journal of Physics* 22.1 (Jan. 2020), p. 013044. DOI: 10.1088/1367-2630/ab6875.

[111] Jens Eisert and Hans J. Briegel. 'Schmidt measure as a tool for quantifying multiparticle entanglement'. In: *Physical Review A* 64.2 (July 2001). DOI: 10.1103/physreva.64.022306.

[112] J. Eisert and D. Gross. 'Multi-particle entanglement'. In: (2005). DOI: 10.48550/ARXIV.QUANT-PH/0505149.

[113] J Eisert, M Cramer and M B Plenio. 'Area laws for the entanglement entropy - a review'. eng. In: *arXiv.org* (2010). ISSN: 2331-8422.

[114] Wikipedia. *Entropy of entanglement — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-April-2023]. 2023. URL: https://en.wikipedia.org/wiki/Entropy_of_entanglement.

[115] 'Introducing Decoherence'. In: *Decoherence and the Quantum-To-Classical Transition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–12. ISBN: 978-3-540-35775-9. DOI: 10.1007/978-3-540-35775-9_1.

[116] Roberto Longo and Feng Xu. 'Von Neumann Entropy in QFT'. In: *Communications in Mathematical Physics* 381.3 (Feb. 2020), pp. 1031–1054. DOI: 10.1007/s00220-020-03702-7.

[117] Charles H. Bennett et al. 'Mixed-state entanglement and quantum error correction'. In: *Phys. Rev. A* 54 (5 Nov. 1996), pp. 3824–3851. DOI: 10.1103/PhysRevA.54.3824.

Appendix A

The results of the run with their respective outcomes and the best fitness scores for all three CA rules

In this appendix, we present the run's results and the circuit's visualization as displayed in the console after each run. Each table is for the best fitness scores of Rule 90 and 110 and three randomly generated sets of CA probabilities. The desired outcome is the known probabilities values, whereas the actual outcome is the generated probabilities values from the evolutionary algorithms.

```
For Rule 90
Best fitness found: 0.0
Best chromosome found: [1, 0, 0, 5, 1, 0, 0, 1, 0, 1, 1, 0, 6,
1, 1, 8, 1, 0, 4, 2, 0, 7, 0, 1, 8, 1, 2, 0, 1, 2, 6, 1, 1, 4,
2, 0, 1, 2, 0, 5, 1, 0, 8, 1, 0]
Initial State | Desired outcome | Actual outcome  | Difference
[0, 0, 0]           0.0000           0.0000           0.0000
[0, 0, 1]           1.0000           1.0000           0.0000
[0, 1, 0]           0.0000           0.0000           0.0000
[0, 1, 1]           1.0000           1.0000           0.0000
[1, 0, 0]           1.0000           1.0000           0.0000
[1, 0, 1]           0.0000           0.0000           0.0000
[1, 1, 0]           1.0000           1.0000           0.0000
[1, 1, 1]           0.0000           0.0000           0.0000
```

```
Total difference: 0.0


For Rule 110
Best fitness found: 0.0009210340371976182
Best chromosome found: [1, 1, 1, 0, 0, 1, 3, 2, 1, 0, 1, 0,
0, 0, 2, 7, 0, 0, 8, 1, 2, 7, 1, 2, 2, 1, 0, 8, 1, 2, 1, 1,
2, 5, 2, 0, 1, 0, 1, 6, 0, 2, 4, 1, 0]
Initial State | Desired outcome | Actual outcome  | Difference
[0, 0, 0]           0.0000           0.0000           0.0000
[0, 0, 1]           1.0000           1.0000           0.0000
[0, 1, 0]           1.0000           1.0000           0.0000
[0, 1, 1]           1.0000           0.0000           1.0000
[1, 0, 0]           0.0000           0.0000           0.0000
[1, 0, 1]           1.0000           1.0000           0.0000
[1, 1, 0]           1.0000           1.0000           0.0000
[1, 1, 1]           0.0000           0.0000           0.0000
Total difference: 1.0


For randomly generated1
Best fitness found: 0.6858593674799021
Best chromosome found: [3, 2, 1, 4, 1, 0, 8, 2, 2, 5, 1, 0, 6,
0, 2, 1, 2, 0, 5, 2, 0, 4, 1, 0, 4, 2, 1, 7, 1, 1, 7, 2, 1, 3,
0, 2, 7, 2, 1, 5, 0, 1, 5, 0, 1]
Initial State | Desired outcome | Actual outcome  | Difference
[0, 0, 0]           0.6364           0.7412           0.1048
[0, 0, 1]           0.6603           0.3398           0.3205
[0, 1, 0]           0.5261           0.6729           0.1467
[0, 1, 1]           0.1748           0.2568           0.0821
[1, 0, 0]           0.8820           0.7690           0.1129
[1, 0, 1]           0.3371           0.2388           0.0984
[1, 1, 0]           0.0340           0.2261           0.1921
[1, 1, 1]           0.4444           0.7773           0.3330
Total difference: 1.3904479642206446


For randomly generated 2
Best fitness found: 0.14418153629199126
Best chromosome found: [3, 2, 0, 6, 0, 1, 4, 1, 0, 8, 0, 1,
```

```
7, 1, 2, 5, 2, 1, 7, 0, 2, 7, 0, 1, 0, 2, 1, 2, 2, 2, 6, 0,
0, 3, 1, 0, 0, 0, 2, 4, 2, 0, 1, 0, 0]
Initial State | Desired outcome | Actual outcome  | Difference
[0, 0, 0]           0.4778           0.5991           0.1213
[0, 0, 1]           0.5604           0.6006           0.0402
[0, 1, 0]           0.8528           0.7271           0.1258
[0, 1, 1]           0.4818           0.2378           0.2440
[1, 0, 0]           0.3143           0.4258           0.1115
[1, 0, 1]           0.3464           0.3940           0.0477
[1, 1, 0]           0.0678           0.2710           0.2032
[1, 1, 1]           0.9124           0.7495           0.1629
Total difference: 1.0564862781447797


For randomly generated 3
Best fitness found: 0.27398475351302753
Best chromosome found: [6, 1, 0, 2, 0, 0, 4, 2, 1, 6, 2, 2, 2,
2, 0, 7, 1, 2, 1, 1, 1, 4, 1, 0, 1, 2, 1, 2, 1, 1, 7, 2, 0, 3, 0,
2, 1, 1, 2, 6, 0, 0, 7, 1, 0]

Initial State | Desired outcome | Actual outcome  | Difference
[0, 0, 0]           0.1988           0.2466           0.0478
[0, 0, 1]           0.4701           0.7559           0.2857
[0, 1, 0]           0.9836           0.7427           0.2410
[0, 1, 1]           0.7115           0.2305           0.4810
[1, 0, 0]           0.6616           0.7324           0.0708
[1, 0, 1]           0.1218           0.2505           0.1286
[1, 1, 0]           0.1328           0.2466           0.1138
[1, 1, 1]           0.7306           0.7554           0.0247
Total difference: 1.3935279792290451
```

Appendix B

Implementation of Meyer-Wallach measure of entanglement

The Meyer-Wallach (MW) entanglement measure, denoted as Q, is a global measure of multi-particle entanglement for pure states. This measure is useful in quantifying the degree of entanglement between particles, which is a crucial aspect of quantum information processing. In the case of two-particle pure states, the measures of entanglement are functions of the eigenvalues of the reduced density matrix and the sum of the k smallest eigenvalues known as the entanglement monotones.

To apply the MW measure of entanglement in a dynamic setting, one can track the evolution of measured entanglement during specific quantum computations with evolutionary algorithms. This measure can be implemented as a fitness function of the genetic algorithms, where the genes are the quantum gates and the chromosomes are the quantum circuits. They evolve with a specific number of generations and runs. The best quantum circuits can be determined based on the fitness measure (entanglement measure) of the evolutionary algorithms.

The mathematical expression for the MW entanglement measure is taken from the paper Ref. [42]. This expression is used to calculate the entanglement of the evolved quantum circuits during the genetic algorithm runs. Implementing the MW measure in this dynamic setting allows for identifying highly entangled states that can be used in various quantum information processing tasks.

In [1]:

```
import numpy as np
import numpy as np
from scipy.linalg import norm
import qutip
```

Mathematical expression of Meyer-wallach entanglement measure

For more than two systems, most entanglement measures require knowledge of the state itself, which involves performing quantum state tomography. For a pure state of n qubits the **Meyer-Wallach entanglement measure** (*Meyer and Wallach 2002*), is popular because of its scalability and ease of computation. Mathematically it is defined as,

$$Q(|\psi\rangle) = \frac{4}{n}\sum_{j=1}^{n} D(|\hat{u}^k\rangle, |\hat{v}^k\rangle),$$

where $|\hat{u}^k\rangle$ and $|\hat{v}^k\rangle$ are vectors in $\mathbf{C^{2n-2}}$ which are non-normalized (indicated by the $\hat{}$) and obtained by projecting on state $|\psi\rangle$ with local projectors on the k^{th} qubit,

$$|\psi\rangle = |0_k\rangle \otimes |\hat{u}^k\rangle + |1_k\rangle \otimes |\hat{v}^k\rangle.$$

The function $D(|\hat{u}^k\rangle, |\hat{v}^k\rangle)$ measures a distance between the two vectors $|\hat{u}^k\rangle$ and $|\hat{v}^k\rangle$. It is obtained by taking the generalized cross product:

$$D(|\hat{u}^k\rangle, |\hat{v}^k\rangle) = \sum_{i<j} |\hat{u}_i^k \hat{v}_j^k - \hat{u}_j^k \hat{v}_i^k|^2.$$

Also, the state $|\psi\rangle$ can be written in the form of Schmidt decomposition over the bipartite division of the k and the other qubits as:

$$|\psi\rangle = |\bar{0}_k\rangle \otimes \hat{x}^k\rangle + |\bar{1}_k\rangle \otimes |\hat{y}^k\rangle,$$

where $\langle \hat{x}^k | \hat{y}^k \rangle = 0$, and $\{|\bar{0}\rangle_k\rangle, |\bar{1}_k\rangle\}$ are related to $\{|0_k\rangle, |1_k\rangle$ by a logical unitary operator $\mathcal{U}_k$. The purity of the state of qubit k is therefore $Tr[\rho_k^2] = \langle \hat{x}^k | \hat{x}^{k\,2} \rangle + \langle \hat{y}^k | \hat{y}^{k\,2} \rangle$. The generalized cross product under logical unitaries, $D(|\hat{u}^k\rangle, |\hat{v}^k\rangle) = D(|\hat{x}^k\rangle, \hat{y}^k\rangle)$ in relation to the norm of an antisymmetric tensor $M_k = |\hat{x}^k\rangle\langle\hat{y}^{*k}| - |\hat{y}^k\rangle\langle\hat{x}^{*k}|$ is written as

$$\begin{aligned} D(|\hat{x}^k\rangle, |\hat{y}^k\rangle) &= \sum_{i<j} |\hat{u}_i^k \hat{v}_j^k - \hat{u}_j^k \hat{v}_i^k|^2 \\ &= \frac{1}{2}\sum_{i,j} (M_k^\dagger)_{ij} (M_k^\dagger)_{ji} \\ &= \frac{1}{2} Tr[M_k^\dagger M_k] \end{aligned}$$

$$\begin{aligned} &= \langle \hat{x}^k | \hat{x}^k \rangle \langle \hat{y}^k | \hat{y}^k \rangle \\ &= \frac{1}{2}(1 - Tr[\rho_k^2]). \end{aligned}$$

Therefore,

$$Q(|\psi\rangle) = 2(1 - \frac{1}{n}\sum_{k=0}^{n-1} Tr[\rho_k^2]).$$

This equation of the entanglement measure Q simplified the physical meaning of the multi-particle entanglement as an average over the entanglements of each qubit with the rest of the system. In addition the measure Q is linearly related to the average purity of each qubit and it is easy to check that it satisfies the following properties:

1. $0 \leq Q(|\psi\rangle) \leq 1$**.** $Q(|\psi\rangle) = 0$ **iff** $|\psi\rangle$ **is a product state and** $Q(|\psi\rangle) = 1$ **for entangled state.**

2. $Q(|\psi\rangle)$ **is invariant under local unitaries.**

In [2]:

```
def compute_Q_ptrace(ket, N):
    """computes the Meyer-Wallach measure, which is a
    measure of global entanglement in a quantum state,"""
    ket = qutip.Qobj(ket, dims=[[2]*(N), [1]*(N)]).unit()
    #This line converts the input ket into a qutip.Qobj
    #object, specifying its dimensions and ensuring
    #the state is normalized. Qutip is a Python library
    #for simulating quantum systems
    print('KET=  ', ket)
    entanglement_sum = 0
    for k in range(N):
        print('value of n', k, 'PTrace: ',ket.ptrace([k])**2 )
        rho_k_sq = ket.ptrace([k])**2
        #This line computes the squared reduced density
        #matrix for the qubit k by taking the partial trace
        #over all other qubits and then squaring the result."""
        entanglement_sum += rho_k_sq.tr()
    #The entanglement sum is then updated by adding the
    #trace of the rho_k_sq
    #density matrix, which is computed using the .tr() method'''

    Q = 2*(1 - (1/N)*entanglement_sum)
    return Q
```

In [3]:

```
if __name__ == "__main__":

   ##Example Circuit 1:: bell state (enatngled quantum circuit)

    n_qubits = 2
    test_state = np.zeros(2**n_qubits)
    #This line creates a numpy array of zeros with a size of
    #2^n_qubits (4 elements in this case, since n_qubits = 2).
    test_state[0] = 1
    test_state[-1] = 1  #[1, 0, 0, 1], |ψ⟩ = 1/√2(|00⟩ + |11⟩)
    #This state is one of the famous entangled states known as
    #the Bell state
    test_state /= np.linalg.norm(test_state)#normalizes the test_state
    print('test state:',test_state.shape)

    print('Circuit 1 (Q=1):')
    Q_value = compute_Q_ptrace(ket=test_state, N=n_qubits)
    print('Q = {}\n'.format(Q_value))
```

```
test state: (4,)
Circuit 1 (Q=1):
KET=   Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type
= ket
Qobj data =
[[0.70710678]
 [0.        ]
 [0.        ]
 [0.70710678]]
value of n 0 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[0.25 0.  ]
 [0.   0.25]]
value of n 1 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[0.25 0.  ]
 [0.   0.25]]
Q = 0.9999999999999996
```

In [4]:

```
# Example Circuit 2: product state (Not entangled quantum circuit)
if __name__ == "__main__":
    n_qubits = 2
    test_state = np.zeros(2**n_qubits)
    test_state[0] = 1
    test_state[1] = 1 ##[1,1,0,0] |ψ⟩ = 1/√2(|00⟩ + |01⟩)
    #or |ψ⟩ = |0⟩(1/√2(|0⟩ + |1⟩))
    test_state /= np.linalg.norm(test_state)

    print('Test #2 (Q=0):')
    Q_value = compute_Q_ptrace(ket=test_state, N=n_qubits)
    print('Q = {}\n'.format(Q_value))
```

```
Test #2 (Q=0):
KET=   Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type
= ket
Qobj data =
[[0.70710678]
 [0.70710678]
 [0.        ]
 [0.        ]]
value of n 0 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[1. 0.]
 [0. 0.]]
value of n 1 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[0.5 0.5]
 [0.5 0.5]]
Q = -8.881784197001252e-16
```

In [5]:

```
def random_state(n):
    """Generates a random n-qubit state vector"""
    state = np.random.randn(2**n) + 1j*np.random.randn(2**n)
    ''' It creates two arrays of size 2**n with normally distributed
    random numbers, one for the real part and one for the imaginary
    part. It then combines them into a single complex array state.'''
    state /= norm(state)
    """This line normalizes the state vector by dividing it by its
    L2 norm, making the total probability of the quantum state
    equal to 1."""
    print(state)
    return state

if __name__ == "__main__":

    n_qubits = 3
    test_state = random_state(n_qubits)
    Q_value = compute_Q_ptrace(ket=test_state, N=n_qubits)
    print('Q = {}\n'.format(Q_value))
```

```
[-0.24496367+0.17231202j -0.12595903-0.39401576j -0.50351139+0.01691
287j
 -0.02195416+0.12632063j -0.36533203+0.25121647j -0.14863191+0.10995
043j
  0.04358352-0.27481279j  0.38247483+0.12030081j]
KET=   Quantum object: dims = [[2, 2, 2], [1, 1, 1]], shape = (8,
1), type = ket
Qobj data =
[[-0.24496367+0.17231202j]
 [-0.12595903-0.39401576j]
 [-0.50351139+0.01691287j]
 [-0.02195416+0.12632063j]
 [-0.36533203+0.25121647j]
 [-0.14863191+0.10995043j]
 [ 0.04358352-0.27481279j]
 [ 0.38247483+0.12030081j]]
value of n 0 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[0.29008424+0.j         0.08838691-0.01567831j]
 [0.08838691+0.01567831j 0.22796149+0.j        ]]
value of n 1 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[ 0.25174394+0.j         -0.0493315 -0.08757172j]
 [-0.0493315 +0.08757172j  0.26860298+0.j        ]]
value of n 2 PTrace:  Quantum object: dims = [[2], [2]], shape = (2,
2), type = oper, isherm = True
Qobj data =
[[0.40946332+0.j         0.04168311-0.16251371j]
 [0.04168311+0.16251371j 0.17444893+0.j        ]]
Q = 0.918463396567369
```

Appendix C

Implementation of von Neumann entanglement entropy

The entropy of entanglement, also referred to as entanglement entropy, is a measure of the quantum entanglement between two subsystems that constitute a composite quantum system. Specifically, for a pure bipartite quantum state of the mixed system, it is feasible to derive a reduced density matrix that characterizes the state of one of the subsystems [20]. The entropy of entanglement is defined as the von Neumann entropy of the reduced density matrix for either of the subsystems [113, 114]. The von Neumann entropy quantifies the uncertainty or randomness in a quantum state and measures the degree of entanglement between the subsystems [20]. If the entropy of entanglement is non-zero, it implies that the subsystem is in a mixed state, and consequently, the two subsystems are entangled [113, 114]. Therefore, the entropy of entanglement offers a useful method for quantifying the degree of entanglement between the subsystems of a composite quantum system.

The von Neumann entropy of entanglement is mathematically defined as the von Neumann entropy of the reduced density matrix for a subsystem within a composite quantum system [116]. Given a bipartite pure state $|\psi\rangle$ representing the composite quantum system, the reduced density matrix ρ_A for subsystem A can be obtained by tracing out the degrees of freedom associated with subsystem B [113]. The von Neumann entropy of entanglement, denoted as $S(\rho_A)$, is subsequently defined as:

$$S(\rho_A) = -\mathrm{Tr}(\rho_A \log_2 \rho_A) \tag{C.1}$$

Here, Tr represents the trace operation, and $\log_2$ denotes the base-2 logarithm. The von Neumann entropy of entanglement provides a measure of the uncertainty or randomness in the state of subsystem A. Thus, it serves as an indicator of the degree of entanglement between the subsystems.

For pure states, a clear distinction exists between entangled and unentangled states. A pure state is entangled, or nonlocal, if and only if its state vector Υ cannot be expressed as a product $\Upsilon_A \otimes \Upsilon_B$ of pure states of its constituent parts. Entangled states cannot be created from unentangled states through any sequence of local actions, even with the assistance of classical communication.

Quantitatively, the entanglement of a pure state is measured by its entropy of entanglement, $E(\Upsilon) = S(\rho_A) = S(\rho_B)$. The von Neumann entropy, $S(\rho) = -\mathrm{Tr}(\rho \log_2 \rho)$, and the reduced density matrices, $\rho_A = \mathrm{Tr}_B(|\Upsilon\rangle\langle\Upsilon|)$ and $\rho_B = \mathrm{Tr}_A(|\Upsilon\rangle\langle\Upsilon|)$, are utilized for this measurement. The entanglement measure, E, ranges from zero for a product state to $\log_2 N$ for a maximally entangled state of two N-state particles [117].

In the case of a composite quantum system being in a mixed state, the von Neumann entropy of entanglement is defined as the minimum entropy of entanglement across all possible pure-state decompositions of the mixed state. This approach enables the quantification of entanglement for mixed states by considering the least entangled representation of the composite system.

```
def random_density_matrix(n_qubits: int) -> np.ndarray:
    """Generates a random density matrix of given number of qubits."""
    dim = 2 ** n_qubits
    A = np.random.rand(dim, dim) + 1j * np.random.rand(dim, dim)
    B = A @ A.conj().T
    trace = np.trace(B)
    rho = B / trace
    return rho

def compute_von_neumann_entropy(rho):
    """Computes the von Neumann entropy of the input density matrix."""
    eigenvalues, _ = np.linalg.eig(rho)
    non_zero_eigenvalues = eigenvalues[eigenvalues > 0]
    entropy = -np.sum(np.nan_to_num(non_zero_eigenvalues * np.log2(
       non_zero_eigenvalues + 1e-15)))
    return entropy
```

This code generates a random density matrix for an n-qubit system and calculates its von Neumann entropy. The *random_density_matrix* function creates a random density matrix by generating a random complex-valued matrix, computing its product with its conjugate transpose,

and normalizing the resulting matrix to have trace 1. The *compute_von_neumann_entropy* function calculates the von Neumann entropy of the input density matrix by first finding its eigenvalues, filtering out the non-zero ones, and then computing the sum of the products of the non-zero eigenvalues and their binary logarithms, multiplied by -1. The resulting von Neumann entropy is displayed as the real part of the calculated entropy value.

C.1 Two-qubit test state

Below are a few examples of calculating the von Neumann entropy of entanglement using the *compute_von_neumann_entropy* function. The examples include a pure entangled state, a pure product state, and a mixed state for a two-qubit system.

C.1.1 Pure product state

```
    # Test #2: pure product state (entropy should be 0)
    n_qubits = 2
    test_state = np.zeros(2**n_qubits, dtype=np.complex128)
    test_state[0] = 1
    rho_product = np.outer(test_state, np.conj(test_state))

    print('Test\#2(entropy=0):')
    entropy_value = compute_von_neumann_entropy(rho=rho_product)
    print('Entropy={}\n'.format(entropy_value))

    Output:
    Test #2 (entropy=0):
    Entropy = (-1.6017132519074577e-15-0j)
```

The test initializes a *test_state* vector as a two-qubit quantum state in the product state, where only one computational basis state has a non-zero amplitude. A density matrix for this pure product state is constructed by taking the outer product of the state vector with its conjugate. The von Neumann entropy of this pure product state should be zero, as the state is not entangled. However, the output of the code shows a very small non-zero value of $(-1.6017132519074577e-15-0j)$ for the entropy. This is a numerical error resulting from the calculation, as the value is close to zero but not exactly zero due to the finite precision of the computation.

C.1.2 Mixed state

```
def random_density_matrix(dim: int) -> np.ndarray:
    """Generates a random density matrix of given dimension."""
    A = np.random.rand(dim, dim) + 1j * np.random.rand(dim, dim)
    B = A @ A.conj().T
```

```
    trace = np.trace(B)
    rho = B / trace
    return rho

def compute_von_neumann_entropy(rho):
    """Computes the von Neumann entropy of the input density matrix."""
    eigenvalues, _ = np.linalg.eig(rho)
    non_zero_eigenvalues = eigenvalues[eigenvalues > 0]
    entropy = -np.sum(np.nan_to_num(non_zero_eigenvalues * np.log2(
        non_zero_eigenvalues + 1e-15)))
    return entropy

# Define dimensions for a two-qubit system (2^2 = 4)
dim = 4
# Generate random density matrix for the two-qubit mixed state
rho_two_qubit = random_density_matrix(dim)

# Calculate von Neumann entropy
entropy = compute_von_neumann_entropy(rho_two_qubit)
# Get the real part of the entropy
real_entropy = np.real(entropy)
print("Two-qubit mixed state density matrix:")
print(rho_two_qubit)
print("\nVon Neumann entropy:", real_entropy)

Output:
Two-qubit mixed state density matrix:
[[0.30660326+0.j         0.2547622 -0.04703429j 0.18418772+0.04200613j
  0.16672403-0.02468697j]
 [0.2547622 +0.04703429j 0.28133384+0.j         0.17067776+0.05341597j
  0.12818799+0.05938023j]
 [0.18418772-0.04200613j 0.17067776-0.05341597j 0.23471324+0.j
  0.13288514+0.00988177j]
 [0.16672403+0.02468697j 0.12818799-0.05938023j 0.13288514-0.00988177j
  0.17734965+0.j        ]]

Von Neumann entropy: 0.9222928882000523
```

This code generates a random density matrix for a two-qubit mixed state and calculates its von Neumann entropy. The output contains two main pieces of information: the density matrix for the two-qubit mixed state and the von Neumann entropy of that density matrix. The density matrix is a 4x4 Hermitian matrix representing the mixed state of a two-qubit quantum system. Each element in this matrix is a complex number, and the matrix is constructed to be positive and semi-definite and have a trace equal to 1. The density matrix provides a complete description of the quantum state of the two-qubit system.

C.2 Three-qubit mixed state

```
import numpy as np

def random_density_matrix(dim: int) -> np.ndarray:
    """Generates a random density matrix of given dimension."""
    A = np.random.rand(dim, dim) + 1j * np.random.rand(dim, dim)
    B = A @ A.conj().T
    trace = np.trace(B)
    rho = B / trace
    return rho
def compute_von_neumann_entropy(rho):
    """Computes the von Neumann entropy of the input density matrix."""
    eigenvalues, _ = np.linalg.eig(rho)
    non_zero_eigenvalues = eigenvalues[eigenvalues > 0]
    entropy = -np.sum(np.nan_to_num(non_zero_eigenvalues * np.log2(
        non_zero_eigenvalues + 1e-15)))
    return entropy
# Define dimensions for a three-qubit system (2^3 = 8)
dim = 8
# Generate random density matrix for the three-qubit mixed state
rho_three_qubit = random_density_matrix(dim)
# Calculate von Neumann entropy
entropy = compute_von_neumann_entropy(rho_three_qubit)
# Get the real part of the entropy
real_entropy = np.real(entropy)

print("Three-qubit␣mixed␣state␣density␣matrix:")
print(rho_three_qubit)
print("\nVon␣Neumann␣entropy:", real_entropy)
Output:
Three-qubit mixed state density matrix:
[[0.14731967-1.61454879e-18j 0.11937881+2.48062970e-02j
  0.07025618-3.29363604e-03j 0.10172821-2.13333937e-03j
  0.08843102+5.09360960e-03j 0.10917251-9.57494419e-03j
  0.09271517+1.21473993e-03j 0.10180099-1.53868062e-02j]
 [0.11937881-2.48062970e-02j 0.16561003+3.05804976e-19j
  0.0825548 +6.57024178e-03j 0.12310625-2.04885447e-02j
  0.09946375+1.10912216e-02j 0.09389617-1.20236264e-02j
  0.08714316-1.65475877e-02j 0.10245255-1.91425637e-02j]
 [0.07025618+3.29363604e-03j 0.0825548 -6.57024178e-03j
  0.08136502+2.22554545e-20j 0.07549187-5.67884686e-03j
  0.06701584+7.69882675e-03j 0.06156673+8.20938929e-04j
  0.05141939-1.63325025e-02j 0.07833285-1.15902309e-02j]
 [0.10172821+2.13333937e-03j 0.12310625+2.04885447e-02j
  0.07549187+5.67884686e-03j 0.14839835-1.91257346e-19j
  0.09297389+5.95854827e-03j 0.11159714-6.08137371e-03j
  0.08499026+2.00447476e-03j 0.08874421+9.34281574e-03j]
```

```
[0.08843102-5.09360960e-03j 0.09946375-1.10912216e-02j
 0.06701584-7.69882675e-03j 0.09297389-5.95854827e-03j
 0.10894843+1.07194170e-18j 0.0919237 +1.01681135e-02j
 0.06901164-1.21757814e-02j 0.08610277+1.07624781e-04j]
[0.10917251+9.57494419e-03j 0.09389617+1.20236264e-02j
 0.06156673-8.20938929e-04j 0.11159714+6.08137371e-03j
 0.0919237 -1.01681135e-02j 0.13917135+6.18591324e-19j
 0.08020232-2.36567494e-03j 0.08697022-1.44319724e-03j]
[0.09271517-1.21473993e-03j 0.08714316+1.65475877e-02j
 0.05141939+1.63325025e-02j 0.08499026-2.00447476e-03j
 0.06901164+1.21757814e-02j 0.08020232+2.36567494e-03j
 0.08728072-8.87828462e-20j 0.07373215+1.91865941e-02j]
[0.10180099+1.53868062e-02j 0.10245255+1.91425637e-02j
 0.07833285+1.15902309e-02j 0.08874421-9.34281574e-03j
 0.08610277-1.07624781e-04j 0.08697022+1.44319724e-03j
 0.07373215-1.91865941e-02j 0.12190644-1.24004475e-19j]]

Von Neumann entropy: 1.3293233692992161
```

The above code generates a random density matrix for a three-qubit mixed state and calculates its von Neumann entropy. The code consists of two main parts: a helper function *random_density_matrix* and the main script.

1. $"random\_density\_matrix(dim : int) -> np.ndarray"$: This is a helper function that takes an integer dim as input and returns a random density matrix of the given dimension. The function creates a complex-valued matrix A by adding a random real matrix and a random imaginary matrix. It then constructs the density matrix by multiplying A with its conjugate transpose $(A.conj().T)$ to obtain a Hermitian matrix B. The density matrix rho is obtained by normalizing B such that its trace (the sum of the diagonal elements) is equal to 1.

2. The main script:

   - First, it defines the dimensions for a three-qubit system by setting $dim = 8(2^3 = 8)$.

   - It generates a random density matrix *rho_three_qubit* for the three-qubit mixed state using the $random\_density\_matrix$ function.

   - Next, the script calculates the von Neumann entropy of the density matrix using the $compute\_von\_neumann_entropy$ function, which was defined in a previous question.

- Since the computed entropy could be a complex number due to numerical inaccuracies, the script extracts the real part of the entropy using $np.real(entropy)$.

- Finally, the script prints the density matrix and the real part of the von Neumann entropy.

```
# Create a maximally mixed state density matrix for a three-qubit system
rho_max_mixed = np.eye(8) / 8
# Calculate the von Neumann entropy for the maximally mixed state
max_entropy = compute_von_neumann_entropy(rho_max_mixed)
print("Max von Neumann entropy for a three-qubit mixed state:", max_entropy
    )

Output:
Max von Neumann entropy for a three-qubit mixed state: 2.99999
```

The above code generates a maximally mixed state density matrix for a three-qubit system, where each qubit is in an equal superposition of its basis states. The density matrix is an 8x8 matrix with all diagonal elements equal to 1/8 and off-diagonal elements equal to 0. Then, the von Neumann entropy is calculated using the *compute_von_neumann_entropy* function. Since the state is maximally mixed, there is no distinguishable information about the system, and therefore the entropy is at its maximum value of 3 bits (for a three-qubit system). However, due to numerical precision, the calculated value of the entropy is not exactly 3, but a very close approximation of it, which is 2.99999.

Therefore, a value of 1.32 for the von Neumann entropy of a three-qubit mixed state means that the state is not fully entangled and has less overall entanglement than a maximally mixed state. It is possible that the state is only partially entangled, meaning that only some of the qubits are entangled with each other while the others are separable.