POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Principles of Quantum Reinforcement Learning

Supervisor

Prof. Davide GIROLAMI

Candidate

Dott. Michele PACIFICO

2024

Table of Contents

Chapter 1

Introduction

Quantum computation is at the forefront of technological advancements, it exploits the principles of quantum mechanics to perform complex calculations by manipulating quantum bits (qubits).

Unlike classical bits, qubits can exist in superposition states, allowing exponentially fewer computational steps than the most efficient classical algorithm for certain problems.

Quantum logic operations applied to qubits are called quantum gates. Quantum gates are unitary transformations that operate on qubits in a quantum system. A sequence of quantum gates forms a quantum circuit.

Classical reinforcement learning (RL) is a branch of machine learning where an agent learns to make decisions through interaction with an environment. The agent takes actions, receives feedbacks in the form of rewards or penalties, and aims to learn an optimal strategy, called policy, to maximize cumulative rewards over time.

Quantum reinforcement learning is the intersection between reinforcement learning and quantum computing, and arises from the need to address challenges associated with quantum computations, such as the efficient preparation of quantum circuits. Designing optimal quantum circuits by minimizing the number of quantum gates represents a key aspect in quantum computing. In this thesis, it is presented an overview of the employment of quantum data within classical RL algorithms to explore problem-solving in quantum contexts.

In particular Chapter 1 presents an overview of the density matrix quantum formalism that is the most suitable to deal with quantum circuits. It provides also an introduction to the fundamentals of quantum computation, that are: qubits, quantum gates and the universality of quantum gates which is a condition required to perform quantum computations.

Chapter 2 illustrates Reinforcement Learning technique and some methods to solve the optimization problems that arise. These methods are then used to train an agent to design a quantum circuit - employing the smallest as possible number

of gates - which transforms an initial state vector associated to a quantum system into a target state of interest. For example to prepare entangled states of N qubits, such as a |00...0> + b |11...1>, starting from the initial state |00...0>.

Chapter 2

Quantum Mechanics and Quantum Computation

The goal of this chapter is to introduce quantum computation and in particular the quantum circuits model and its main implications. To do so it has been conveniently defined the density matrix formalism.
Quantum mechanics was born in the early 1900s, and made it possible to justify some phenomena, otherwise inexplicable classically, revealing itself as the most complete and accurate description of the world.
In recent years, there has been an increasing interest in building "quantum computers", processing information by controlling atoms and photons. Also, it is interesting to explore the link between this new kind of computing and AI/Machine Learning, for which ever greater computational resources are required. A key aspect to achieving "quantum supremacy" is entanglement, a characteristic property of quantum systems correlated to each other, without a classical analogue. Entanglement can be seen as a new type of physical resource, indispensable for new applications such as quantum teleportation, fast quantum algorithms, and quantum error-correction.

2.1 Density operator

A convenient formulation of quantum mechanics dealing with compound systems is provided by the *density operator* or *density matrix* formalism [1]. Let us consider a system in the state $|\psi_i\rangle$ with probability p_i, then to the *ensemble of pure states* $\{p_i, |\psi_i\rangle\}$ is associated the density operator:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \tag{2.1}$$

Assuming that the time evolution of a closed system is described by a unitary operator U, the **time evolution of the density operator** of the system, which is initially in state $|\psi_i\rangle$, will be:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger \tag{2.2}$$

2.1.1 Measurements in the density operator formalism

Consider performing a measurement described by operator M_m. The probability of obtaining m, given an initial state $|\psi_i\rangle$, is:

$$p(m|i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle = tr(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \tag{2.3}$$

where the equation $tr(A|\psi\rangle\langle\psi|) = \langle\psi|A|\psi\rangle$ has been used. To obtain this last equality, the Grand-Schmidt procedure has been implemented in order to extend $|\psi\rangle$ to an orthonormal basis $|i\rangle$ that has $|\psi\rangle$ as former element:

$$tr(A|\psi\rangle\langle\psi|) = \sum_i \langle i|A|\psi\rangle\langle\psi|i\rangle = \langle\psi|A|\psi\rangle \tag{2.4}$$

where $|\psi\rangle$ is a unit vector and A an arbitrary operator.

Exploiting the *total probability law*:

$$\begin{aligned} p(m) &= \sum_i p(m|i)p_i \\ &= \sum_i p_i tr(M_m^\dagger M_m \rho|\psi_i\rangle\langle\psi_i|) \\ &= tr(M_m^\dagger M_m \rho) \end{aligned} \tag{2.5}$$

After having obtained m from a measurement, the state $|\psi_i\rangle$ is:

$$|\psi_i^m\rangle = \frac{M_m|\psi_i\rangle}{\sqrt{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}}$$

Hence, after a measurement produces an outcome m, there is an ensemble of states $|\psi_i^m\rangle$ with associated probabilities $p(i|m)$. The corresponding density operator is:

$$\rho_m = \sum_i p(i|m)|\psi_i^m\rangle\langle\psi_i^m| = \sum_i p(i|m)\frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}$$

Noticing that $p(i|m) = p(m,i)/p(m) = p(m|i)p_i/p(m)$ and replacing with (3) and (5) we obtain:

$$\rho_m = \sum_i p_i \frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{tr(M_m^\dagger M_m\rho)} = \frac{M_m\rho M_m^\dagger}{tr(M_m^\dagger M_m\rho)}$$

2.1.2 Pure states and mixed states

If a quantum system state $|\psi\rangle$ is perfectly known, then the system is in a pure state. The corresponding density operator is: $\rho = |\psi\rangle\langle\psi|$. Otherwise ρ will be in a mixed state.
There is a simple criterion to establish if a state is pure or mixed:

pure state: $tr(\rho^2) = 1$

mixed state: $tr(\rho^2) < 1$

Consider a quantum system in state ρ_i with probability p_i. Suppose that ρ_i derives from some ensemble $\{p_{ij}, |\psi_{ij}\rangle\}$ of pure states. It is possible to prove that the density operator is:

$$\rho = \sum_i p_i p_{ij} |\psi_{ij}\rangle\langle\psi_{ij}| = \sum_i p_i \rho_i \tag{2.6}$$

ρ is called *mixture* of states ρ_i with probabilities p_i. The concept of *mixture* is widely used in analysis problems like those concerning quantum noise, in which noise causes an uncertainty about the quantum state.
For example, one can imagine having lost track of the outcome of a measurement m, but it is possible to have a state ρ_m with probability $p(m)$, without knowing the value of m. The density operator that describes such a state is then:

$$\rho = \sum_m p(m)\rho_m = \sum_m tr(M_m^\dagger M_m \rho) \frac{M_m \rho M_m^\dagger}{tr(M_m^\dagger M_m \rho)} = \sum_m M_m \rho M_m^\dagger \tag{2.7}$$

2.1.3 Properties of the density operator

An operator ρ is the density operator associated to a generic ensemble $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the following conditions:

Trace condition: the trace of ρ must be equal to one.
Being $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$:

$$tr(\rho) = \sum_i p_i tr(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1$$

Positivity condition: ρ is a positive operator. Considering an arbitrary vector $|\phi\rangle$ in the space of states:

$$\langle\phi|\rho|\phi\rangle = \sum_i p_i \langle\phi|\psi_i\rangle\langle\psi_i|\phi\rangle = \sum_i p_i |\langle\phi|\psi_i\rangle|^2 \geq 0$$

Then, considering an arbitrary operator ρ that satisfies the above conditions, since ρ is positive, it has a spectral decomposition

$$\rho = \sum_j \lambda_j |j\rangle\langle j|$$

where the vectors $|j\rangle$ are orthonormal, and λ_j are the real non-negative eigenvalues of ρ.
From the trace condition follows that $\sum_j \lambda_j = 1$. Thus a system in state $|j\rangle$ with probability λ_j will have ρ as density operator. In other words, $\{\lambda_j|j\rangle\}$ is a set of states that defines the density operator ρ.

2.1.4 Reduced Density Operator

Let us consider a system made of two subsystems A e B, described by density operator ρ^{AB}. The *reduced density operator* of A is defined as:

$$\rho^A \equiv tr_B(\rho^{AB}) \tag{2.8}$$

with tr_B defined as:

$$tr_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle|a_2|tr(|b_1\rangle\langle b_2|) \tag{2.9}$$

with $|a_1\rangle, |a_1\rangle$ and $|a_1\rangle, |a_1\rangle$ being vectors in spaces A e B respectively, and tr_B is called the *partial trace* over the state B.

2.1.5 Schmidt decomposition and purification

The Schmidt decomposition is a powerful and useful tool in order to describe composite systems:

Theorem 1 *Let us consider the pure state $|\psi\rangle$ of a composed system AB. There are orthonormal states $|i_A\rangle$ referred to system A, and $|i_B\rangle$ for B, such that:*

$$|\psi\rangle = \sum_i \lambda_i |i_A\rangle|i_B\rangle \tag{2.10}$$

where $\lambda_i \in \Re, \lambda_i \geq 0$ are called Schmidt coefficients and satisfy $\sum_i \lambda_i^2 = 1$.

The states $|i_A\rangle$ and $|i_B\rangle$ are called the *Schmidt bases* for A and B, respectively, while the non-zero Schmidt coefficient λ_i are called *Schmidt numbers* for the state $|\psi\rangle$, that can be seen, for compound quantum systems, as a measure of entanglement between two systems.

Another important tool for studying composite systems is *purification*. Purification is a mathematical procedure where a mixed state can be view as part of a pure state in a higher-dimensional Hilbert space. Mathematically, given the quantum state ρ_A, it is possible to introduce another system $|\psi\rangle$ such that $\rho_A = tr_B(|\psi\rangle\langle\psi|)$. We can say that $|\psi\rangle$ purifies ρ_A if

$$tr_B(|\psi\rangle\langle\psi|) = \rho_A \tag{2.11}$$

To prove the above formula we can define $|\psi\rangle$, through the Schmidt decomposition, as:

$$|\psi\rangle = \sum_i \sqrt{p_i}|i_A\rangle|i_B\rangle \tag{2.12}$$

Therefore

$$\begin{aligned} tr_B(|\psi\rangle\langle\psi|) &= \sum_{ij} \sqrt{p_i p_j}|i_A\rangle\langle j_A|tr(|i_B\rangle\langle j_B|) \\ &= \sum_{ij} \sqrt{p_i p_j}|i_A\rangle\langle j_A|\delta_{ij} \\ &= \sum_i p_i|i_A\rangle\langle i_A| \\ &= \rho_A \end{aligned}$$

2.1.6 Postulates of quantum mechanics

Summarizing, it is possible to formulate the quantum mechanics postulates by means of the density matrix:

- **Postulate 1:** A complex vector space with internal product (Hilbert space), known as states space, is associated to any isolated physical system. The system is completely described by its density operator, which is a positive operator with trace equal to one, and that acts on the system state space. For a quantum system in state ρ_i with probability p_i, the corresponding density matrix is $\rho = \sum_i p_i \rho_i$.
- **Postulate 2:** The time evolution of a closed quantum system is described by a *unitary transformation*. Meaning that the state ρ of the system at time t_1 will evolve into a state ρ' of the system a time t_2 through a unitary operator U that depends only on times t_1 and t_2 : $\rho' = U\rho U^\dagger$
- **Postulate 3:** Quantum measurements are described by a set of operators $\{M_m\}$, where indices m are the possible outcomes of a measurement. If the

quantum state in the instant before the measurement is ρ, then the probability that m is measured is given by: $p(m) = tr(M_m M_m^\dagger \rho)$. The state after the measurement will be $M_m \rho M_m^\dagger / tr(M_m M_m^\dagger \rho)$.
The measurement operator satisfies the completeness relation $\sum_m M_m^\dagger M_m = I_d$, where I_d is the identity matrix.

- **Postulate 4:** The space of the states of a composite system is the tensor product of the component spaces.
  Furthermore for n systems, such that the i-th system is in state ρ_i, the joint quantum state composed of n states is: $\sum_{k_1 \dots k_n} p_{k_1 - k_n} \rho_1^{k_1} \otimes \dots \rho_n^{k_n}$.

2.2 Quantum Computation

As for classical computation, we have quantum-type logic circuits, simply called "quantum circuits", which are the basis of the theory of quantum computation.

The language of quantum circuits is fundamental for the description of quantum algorithms. Quantum circuits are composed of an ordered sequence of components, mostly quantum gates, measurements and resets, concepts that will be introduced in the following chapter.

2.2.1 Qubit and entanglement

A bit is the fundamental conceptual basis for classical computation, it can take value 0 or 1.
The Qubit is instead a basic unit of quantum information. It can exist in a continuum of states between 0 and 1, until it is observed. It is as well possible to create linear combinations of states, called superpositions:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.13}$$

with α , $\beta \in \mathbb{C}$ and $|0\rangle$, $|1\rangle$ called *computational basis states*.
The qubit is a two-state quantum system that can exist in any quantum superposition of two independent states. For example a qubit can be in state

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

and when measured can give as output 0 or 1 with the same probability (1/2).
Let us consider a system of two qubits:

$$|\psi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.14}$$

This is an entangled state, in which spin measurements can take value 0 or 1. Suppose to perform spin measurements along $\vec{v}$ axis for both qubits, that is measure the observable $\vec{v} \cdot \vec{\sigma}$ for each particle defined as:

$$\vec{v} \cdot \vec{\sigma} = v_1\sigma_1 + v_2\sigma_2 + v_3\sigma_3 \tag{2.15}$$

which will give as result -1 or +1 for each particle. Independently from the choice of $\vec{v}$, the results of the measurements on the two particles will be always opposite. To prove this let us consider the `eigenstates` $|a\rangle$ e $|b\rangle$ of $\vec{v} \cdot \vec{\sigma}$. Hence there exist $\alpha, \beta, \gamma, \delta$ complex numbers such that:

$$|0\rangle = \alpha|a\rangle + \beta|b\rangle \tag{2.16}$$
$$|1\rangle = \gamma|a\rangle + \delta|b\rangle \tag{2.17}$$

Substituting, one gets:

$$\frac{|01\rangle - |10\rangle}{\sqrt{2}} = (\alpha\delta - \beta\gamma)\frac{|ab\rangle - |ba\rangle}{\sqrt{2}} \tag{2.18}$$

But $(\alpha\delta - \beta\gamma)$ is the determinant of a unitary matrix for a change of basis, then it is equal to a `not observable` global phase factor $e^{i\theta}$. Hence it is possible to write:

$$\frac{|01\rangle - |10\rangle}{\sqrt{2}} = \frac{|ab\rangle - |ba\rangle}{\sqrt{2}} \tag{2.19}$$

Consequently, measuring $\vec{v} \cdot \vec{\sigma}$ on both particles, a result of +1(-1) on the first spin implies a result of -1(+1) on the second one. Therefore, it is always possible to guess the outcome of a measurement of one of the particles in a specific basis if it is known the spin measurement of the other correlated particles in the same basis. A useful geometrical representation for the pure state space of a qubit is the Bloch sphere. This representation arises from the fact that the total probability of a system of being in a pure state $|\psi\rangle$ is $\langle\psi|\psi\rangle$ equal to 1 and then $\||\psi\rangle\|^2 = 1$. This constrain allows to rewrite $|\psi\rangle$ as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.20}$$

with $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$. This allows to uniquely represent all the possible states of ϕ through a point $\vec{a}$ on the surface of a sphere of unitary radius, similarly to the spherical coordinates:

$$\vec{a} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) = (u, v, w) \tag{2.21}$$

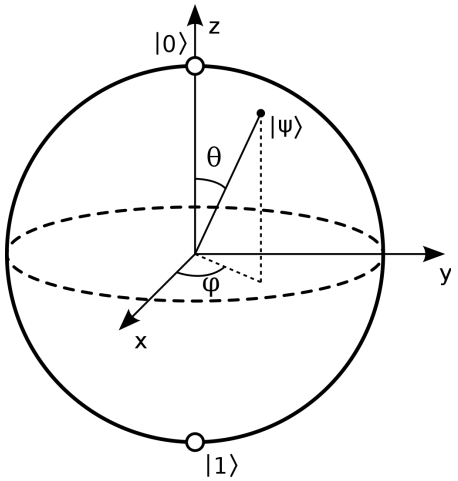

Figure 2.1: Bloch Sphere

2.2.2 Quantum Gates

Quantum logic gates are the building blocks for quantum circuits. An important feature of quantum gates, unlike classical logic gates, is that they are always reversible. This property can be visualized as a rotation around the Bloch sphere. The quantum gates acting on n qubits can be represented mathematically by $2^n \times 2^n$ unitary matrices. They preserve the norm $|a|^2 + |b|^2 = 1$ for all states $|\Phi\rangle = a|0\rangle + b|1\rangle$.
In principle there is an infinite number of gates. Some of the most important quantum gates, operating on a single-bit, are the Pauli gates (X, Y, Z). We can think at these gates as a rotation by π radians around the x, y and z-axis, respectively, of the Bloch sphere. The X-gate is also often called a NOT-gate, referring to its classical analogue.

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.22}$$

Other important gates, operating on a single qubit, are phase shift gates that modify the phase of the quantum state. These gates represent rotations along the z-axis on the Bloch sphere by θ. They have a matrix representation:

$$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \tag{2.23}$$

Some common examples are:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = P(\pi)$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = P(\frac{\pi}{2}) = \sqrt{Z}$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{(i\pi/4)} \end{bmatrix} = P(\frac{\pi}{4}) = \sqrt[4]{Z}$$

Another quantum gate widely used in literature acting on a single qubit is the Hadamard gate. It performs a rotation of π about the axis $(\hat{x}+\hat{z})/\sqrt{2}$ at the Bloch sphere. The Hadamard gate maps the computational basis $|0\rangle$ and $|1\rangle$ into a two states superposition with equal weight:

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

. In the matrix notation:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \tag{2.24}$$

An interesting fact is that $H = (X + Z)/\sqrt{2}$.

Exponentializing the Pauli Matrix we can define:

$$R_x(\theta) = e^{-i\theta\frac{X}{2}} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.25}$$

$$R_y(\theta) = e^{-i\theta\frac{Y}{2}} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.26}$$

$$R_z(\theta) = e^{-i\theta\frac{Z}{2}} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.27}$$

Generalizing in three dimensions, we can define the rotation by θ around the $\hat{n} = (n_x, n_y, n_z)$ axis as:

$$R_{\hat{n}} \equiv exp(-i\theta\hat{n}\cdot\vec{\sigma}/2) = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}(n_xX + n_yY + n_zZ) \tag{2.28}$$

Theorem 2 *Suppose U is a unitary matrix on a single qubit. Then there exist real numbers $\alpha, \beta, \gamma, \delta$ such that:*

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta) \tag{2.29}$$

This theorem implies an important corollary in quantum computation: Suppose U is an arbitrary unitary gate acting on a single quibit. Then $\exists A, B, C \in \mathbb{C}^{2x2}$ unitary operators on a single qubit and $\alpha \in \mathbb{R}$ overall phase factor such that

$$ABC = I, U = e^{i\alpha} AXBXC$$

Controlled operations

Controlled operations, of the type "If A, then B", are very common in classical computation. In general, in a quantum computation, a controlled operation is a two qubit operation in which if the first qubit (control qubit) is set then an arbitrary unitary operation U is applied to the second qubit (target qubit), otherwise the second qubit is left unchanged: $|c\rangle|t\rangle \rightarrow |c\rangle U^c|t\rangle$.

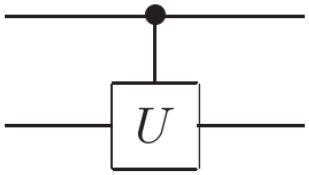

Figure 2.2: Controlled-U operation. The top line represents the control qubit, and the bottom line represents the target qubit. The timeline goes from left to right

One typical controlled operation is the controlled-NOT gate, also called CNOT gate. The action of the CNOT gate is: if the control qubit is set to "1" then the target qubit is flipped.

$$\text{CNOT} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.30}$$

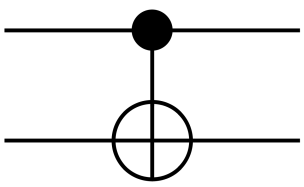

Figure 2.3: CNOT gate in circuit representation

We can express a controlled operation substituting, as we have seen above, an arbitrary unitary operation U with three single qubit operations A, B, C, some NOT-gate X and a phase shift. The basic idea to build a first circuit implementing controlled-U operation is substituting $U = e^{i\alpha}AXBXC$ and the NOT gate with the C-NOT gate: if the controlled qubit is set then $e^{i\alpha}AXBXC = U$ is applied on the control qubit, otherwise the operation $ABC = I$ is applied on the control qubit that keeps its state unchanged.

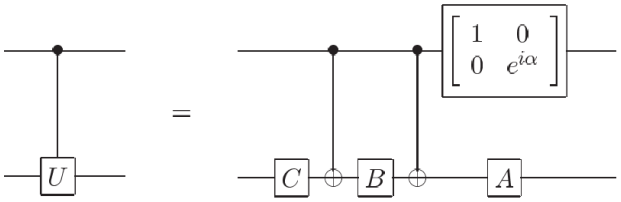

Figure 2.4: Circuit implementing controlled U-operation for single qubit.

It is possible to expand the previous results to control multiple qubits. Having $n + k$ qubits, and U unitary operation over k qubits, we can define the controlled operation $C^n(U)$ as:

$$C^n(U)|x_1x_2...x_n\rangle|\psi\rangle = |x_1x_2...x_n\rangle U^{X_1X_2...X_n}|\psi\rangle \tag{2.31}$$

For simplicity we assume $k = 1$. We can define V unitary operation such that $V^2 = U$ with U single quibit unitary operator. Defining $V \equiv (1 - i)(I + iX)/2$, where $V^2 = X$, the resulting $C^2(U)$ operation implements the Toffoli gate.

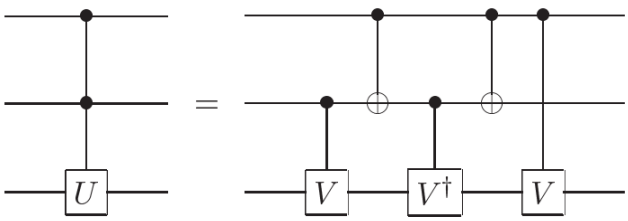

Figure 2.5: Circuits for $C^2(U)$ gate. The special case $V \equiv (1 - i)(I + iX)/2$ corresponds to the Toffoli gate .

The Toffoli gate is a reversable universal logic gate in a classical setting, that flips the third bit if and only if the first two bits are both set to 1. So the previous result shows that it is possible to perform all classical computations on a quantum computer using only one and two qubits operations. This important result does

not mean that Toffoli gate is universal for quantum computation, but rather that classical computation is a subset of quantum computation.

Universal quantum gates

In a classical setting we say that a set of gates is universal if they can be used to compute an arbitrary function. In quantum computation we refer as universal quantum gates to any finite sequence of gates, among a set of quantum gates, through which it is possible to express a unitary matrix with arbitrary accuracy. The goal of quantum computation is to find the best sequence of gates (unitary transformations) that can be performed efficiently, because in principle there is an infinite number of gates and combinations of them.
In practice it is often enough to use the classic reversible gates, such as the Toffoli gate, for universal quantum computation, as a weaker kind of universality. But furthermore it is possible to demonstrate that an arbitrary unitary matrix acting on a system of n qubits may be written as the product of at most n two-level unitary matrices.

Theorem 3 *Two level gates are universal for quantum computation.*
$\forall U \in \mathbf{C}^{3x3}$ *unitary matrix* $\exists U_i \in \mathbf{C}^{3x3} : U_i = U_i' \otimes 1$, $U_i' \in \mathbf{C}^{3x3}$ *unitary matrix with* $i \in \{1,2,3\}$ *such that:*

$$U = U_1^\dagger U_2^\dagger U_3^\dagger$$

or

$$U_3 U_2 U_1 U = I.$$

Proof:

$$U = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\text{if } b = 0 \; U_1 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ if } b \neq 0 \; U_1 \equiv \begin{bmatrix} \frac{a^*}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} & 0 \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Noting that U_1 is a two-level unitary matrix: $U_1 U = \begin{bmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{bmatrix}$

$$\text{if } c_1 = 0 \; U_2 \equiv \begin{bmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ if } c_1 \neq 0 \; U_2 \equiv \begin{bmatrix} \frac{a'^*}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{c'^*}{\sqrt{|a'|^2+|c'|^2}} \\ 0 & 1 & 0 \\ \frac{c'}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{-a'}{\sqrt{|a'|^2+|c'|^2}} \end{bmatrix}$$

$$U_2U_1U = \begin{bmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{bmatrix}$$

Since U,U_1 and U_2 are unitary, it follows that U_2U_1U is unitary $\Longrightarrow d'' = g'' = 0$.
Finally setting $U_3 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & e''^* & f''^* \\ 0 & f''^* & j''^* \end{bmatrix} \Longrightarrow U_3U_2U_1U = I$ which proves the statement.

Similarly for higher dimensions:
Let $U \in \mathbf{C}^{dxd}$ be an arbitrary unitary matrix $\Longrightarrow \exists U_1, U_2, U_3..., U_N$ with $N \leq \frac{d(d-1)}{2}$ such that:

$$\exists U = U_1U_2...U_N.$$

In other words an arbirtary unitary matrix can be written as a product of at most $2^{n-1}(2^n - 1)$ two-level unitary matrices.

2.2.3 Universality of CNOT and single qubit gates

Theorem 4 *Any unitary gate acting on a n-qubit can be implemented with single qubit and CNOT gates, therefore they are universal for quantum computation.*

Thanks to the previous results, it is sufficient to prove the theorem for a two level unitary matrix, which acts not trivially on two qubits. Suppose U is an arbitrary two-level unitary matrix, that acts not trivially on two qubits s and t. For instance, we define unitary matrix U with dimension $d = 8 = 2^3$ as:

$$U = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & b \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix}$$

with $a, b, c, d \in \mathbf{C}$. U can be reduced to a $d = 2 \times 2$ unitary matrix $\tilde{U} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
Note that U acts not trivially only on the states $s = |000\rangle$ and $t = |111\rangle$. Let us define the Gray code connecting s and t, as a sequence of binary numbers, starting from s and ending with t, such that adjacent members differ only by one bit and

thus if s and t differ in n bits, the shortest Gray code will have $n+1$ members. In the example:

$$\begin{array}{cccc} & A & B & C \\ g_1 = & 0 & 0 & 0 \\ g_2 = & 1 & 0 & 0 \\ g_3 = & 1 & 1 & 0 \\ g_4 = & 1 & 1 & 1 \end{array}$$

The idea to implement U is to find the sequence of gates that transforms state $|g_1\rangle$ into state $|g_3\rangle$ through $|g_1\rangle \rightarrow |g_2\rangle \rightarrow |g_3\rangle$, then apply the controlled operation $\tilde{U}$ on the last qubit which is represented by a superposition of states $|g_3\rangle$ and $|g_4\rangle$, and then go back to the initial state performing the inverse operations $|g_3\rangle \rightarrow |g_2\rangle \rightarrow |g_1\rangle$.

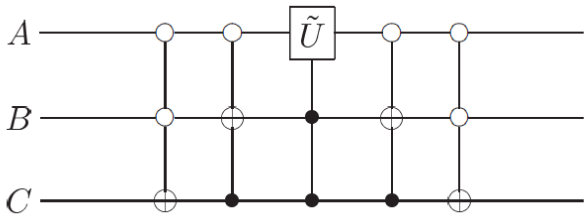

Figure 2.6: Circuits implementing the two-level operation U in the example.

Generalizing we require at most $2(n-1)$ controlled operations to change the state $|g_1\rangle \rightarrow |g_2\rangle \rightarrow ... \rightarrow |g_m - 1\rangle$ and go backwards $|g_{m-1}\rangle \rightarrow |g_{m-2}\rangle \rightarrow ... \rightarrow |g_1\rangle$, this can be implemented with $O(n)$ qubits and CNOT gates. Considering that the number of qubits necessary to implement $\tilde{U}$ is of order $O(n)$, it follows that the number of qubit to implement U is $O(n^2)$. Consider also, as seen before, that an arbitrary unitary matrix $U \in \mathbf{C}^{n \times n}$ on n qubits can be written as a product of $O(2^{2n}) = O(4^n)$ unitary operations. Finally, taking into account the various contributions, we obtain that to compute an arbitrary unitary operation on n qubits we require $O(n^2 4^n)$ single qubits and CNOT gates. Unfortunately there are not easy error correction methods to implement all of these gates. But there exist some sets of gates that can be used for universal quantum computation.

2.2.4 Universality of Hadamard + phase + CNOT + $\pi/8$ gates

Given U and V two unitary operators acting on the same state space, we define the *error* of approximation in implementing V instead of U as:

$$E(U,V) \equiv max_{|\psi\rangle} ||(U-V)|\psi\rangle||$$

By definition:
$E(U_2U_1, V_2V_1) = ||(U_2U_1 - V_2V_1)|\psi\rangle||$
$= ||(U_2U_1 - V_2U_1)|\psi\rangle + (V_2U_1 - V_2V_1)|\psi\rangle||$
$\leq ||(U_2U_1 - V_2U_1)|\psi\rangle|| + ||(V_2U_1 - V_2V_1)|\psi\rangle||$
$\leq E(U_2, V_2) + E(U_1, V_1)$
Generalizing, by induction, for a sequence of m unitary gates:

$$E(U_mU_{m-1}...U_1, V_mV_{m-1}...V_1) \leq \sum_{j=1}^{m} E(U_j, V_j) \tag{2.32}$$

So the error increases linearly with m.
Any unitary single qubit operation can be approximated to an arbitrary precision by a discrete set of gates. In particular Hadamard gate H, phase gate S, CNOT gate, $\pi/8$ gate are universal for quantum computation. Alternately it is possible to replace $\pi/8$ gate with Toffoli gate to obtain another universal set of gates.
Thanks to the previous results, considering the gate T that is a rotation by $\pi/4$ radians around the $\hat{z}$ axis on the Bloch sphere, and the sequence of gates HTH that is a rotation by $\pi/4$ radians around the $\hat{x}$ axis on the Bloch sphere, we can define:

$$exp(-i\frac{\pi}{8}Z)exp(-i\frac{\pi}{8}X) = [\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}Z][\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}X] \tag{2.33}$$

$$= \cos^2\frac{\pi}{8}I - i[\cos\frac{\pi}{8}(X+Z) + \sin\frac{\pi}{8}Y]\sin\frac{\pi}{8} \tag{2.34}$$

$$= R_{\hat{n}}(\theta). \tag{2.35}$$

That is the rotation of the Bloch sphere by an angle θ, defined as $\cos(\theta/2) \equiv \cos^2\pi/8$, around the axis $\vec{n} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$.
Noting that θ is an irrational multiple of 2π, we can iterate on $R_{\hat{n}}(\theta)$ to approximate to an arbitrary accuracy $\delta > 0$ any rotation $R_{\hat{n}}(\alpha)$.
For any α holds: $HR_{\hat{n}}(\alpha)H = R_{\hat{m}}(\alpha)$, with $\hat{m} = (\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$.
Moreover $\forall\ U \in \mathbf{C}^{2x2}$ unitary operator, $\exists\ \alpha, \beta, \gamma, \delta \in \mathbf{R}$ such that: $U = e^{i\alpha}R_{\hat{n}}(\beta)R_{\hat{m}}(\gamma)R_{\hat{n}}(\delta)$.
Then for any arbitrary unitary operator U and any accuracy $\epsilon > 0$, we have that there exist $n_1, n_2, n_3 \in \mathbf{N}$ such that the error in approximation using a circuit composed only of Hadamard and $\pi/8$ gates is:

$$E(U, R_{\hat{n}}(\theta)^{n_1}HR_{\hat{n}}(\theta)^{n_2}HR_{\hat{n}}(\theta)^{n_3}) < \epsilon \tag{2.36}$$

A "fast" convergence for the number of gates to approximate an arbitrary unitary operation is given thanks to the Solovay-Kitaev theorem.

Theorem 5 (Solovay-Kitaev theorem) *Any quantum circuit containing m CNOTs and single qubit gates can be approximated with accuracy ϵ using $O(m \log^c(m/\epsilon))$ gates from a discrete set, where $c \to 2$.*

Furthermore, it can be proved that there is a unitary transformation U on n qubits that can be approximated by V with $\Omega(2^n \log(1/\epsilon)/\log(n))$ operations such that $E(U, V) \leq \epsilon$. To conclude, combining the previous results, to approximate an arbitrary unitary operation U on n qubits within an accuracy ϵ, we require $O(n^2 4^n \log^c(n^2 4^n/\epsilon))$ gates.
The unitary operation that can be computed efficiently in a quantum circuit model is still an open question.

2.3 Distance Measures

In quantum information, we are interested in distance measures to quantify how close two quantum states are (static measure), and how well information is preserved trough a quantum channel (dynamic measure).
First of all, in classical information, an information source can be seen as a random variable: the probability distribution of its values is computed over all the population. Such probability distributions are useful to define distance measures for quantum information.
The trace distance and fidelity are static measures of the distance between two fixed probability distributions. Given two probability distributions p_x and q_x, we define the trace distance between them as

$$D(p_x, q_x) \equiv \frac{1}{2} \sum_x |p_x - q_x| \tag{2.37}$$

and the fidelity as

$$F(p_x, q_x) \equiv \sum_x \sqrt{p_x q_x} \tag{2.38}$$

To evaluate how well the information is preserved by an evolution we need a dynamic measure, that can be derived from a special case of static trace distance. Given a pair $(X, \tilde{X})$, composed by the random variable X and a copy of it $\tilde{X} = X$, assuming now that X passes through a noisy channel becoming the random variable

Y, we can evaluate how close $(\tilde{X}, X)$ is to (X, Y) as:

$$\begin{aligned} D((\tilde{X}, X), (X, Y)) &= \frac{1}{2} \sum_{xx'} |\delta_{xx'} p(X = x) - p(\tilde{X} = X, Y = x')| \\ &= \frac{1}{2} \sum_{x \neq x'} p(\tilde{X} = X, Y = x') + \frac{1}{2} \sum_{x} |p(X = x) - p(\tilde{X} = X, Y = x')| \\ &= \frac{1}{2} \sum_{x \neq x'} p(\tilde{X} = X, Y = x') + \frac{1}{2} \sum_{x} (p(X = x) - p(\tilde{X} = X, Y = x')) \\ &= \frac{p(\tilde{X} \neq Y) + 1 - p(\tilde{X} = Y)}{2} \\ &= \frac{p(X \neq Y) + p(\tilde{X} \neq Y)}{2} \\ &= p(X \neq Y) \end{aligned}$$

Hence the trace distance between the probability distributions $(\tilde{X}, X)$ and (X, Y) is equal to the probability that X is not equal to Y: $p(X \neq Y)$. A similar procedure, considering quantum states rather than classical variables, can be used to derive a dynamic measure of how well information has been preserved across a channel.
In the quantum scenario, given two quantum states ρ and σ, we define their trace distance as:

$$D(\rho, \sigma) \equiv \frac{1}{2} tr|\rho - \sigma| \tag{2.39}$$

and the fidelity as:

$$F(\rho, \sigma) \equiv tr\sqrt{\rho^{1/2} \sigma \rho^{1/2}} = \max_{|\psi\rangle, |\phi\rangle} |\langle \psi | \phi \rangle| \tag{2.40}$$

where the last equality is given thanks to the *Uhlmann's theorem*.
These two measures are strictly connected one to the other, indeed they have opposite behaviors: when the trace distance increases, while two states become more distinguishable, the fidelity decreases and vice versa. In many applications these measures lead to equivalent results, for this reason they are both widely used in quantum computation.
A significant result that makes these measures so important, is that no physical process ever increases the distance between two quantum states. This result is known as the *strong convexity property* of the trace distance, or in the same way *strong concavity property* of the fidelity distance.
Another important measure, of the closeness between probability distributions, is the *relative entropy*. Given two probability distributions p_x and q_x on the same

index set x, we define the relative entropy of p_x to q_x as:

$$H(p_x||q_x) \equiv \sum_x p_x \log \frac{p_x}{q_x} \equiv -H(X) - \sum_x p_x \log q_x \tag{2.41}$$

As done for the fidelity and the trace distance we can define a quantum version of the relative entropy. Given two density operators ρ and σ, we can define the quantum relative entropy of ρ to σ as:

$$S(\rho||\sigma) \equiv tr(\rho \log \rho) - tr(\rho \log \sigma) \tag{2.42}$$

Chapter 3

Quantum Reinforcement learning

In the wide world of artificial intelligence, Reinforcement Learning (RL) [2] represents one of the most promising fields of innovation. One of the possible applications of reinforcement learning is to improve computing techniques within quantum computers.

The scope of this chapter is to give an overview on the intersection between quantum computers and reinforcement learning.

3.1 Reinforcement Learning

Reinforcement Learning is a Machine Learning technique that, unlike the other kinds of machine learning methods (supervised and unsupervised learning) is based on the idea that, as humans learn from experience to make decisions, a machine learns through iterative interactions with the environment and receives positive or negative feedbacks based on its actions.

At its core, this paradigm revolves around the concept of an agent navigating an environment, learning optimal actions through a system of trial and error, and being rewarded or penalized based on the consequences of its decisions.

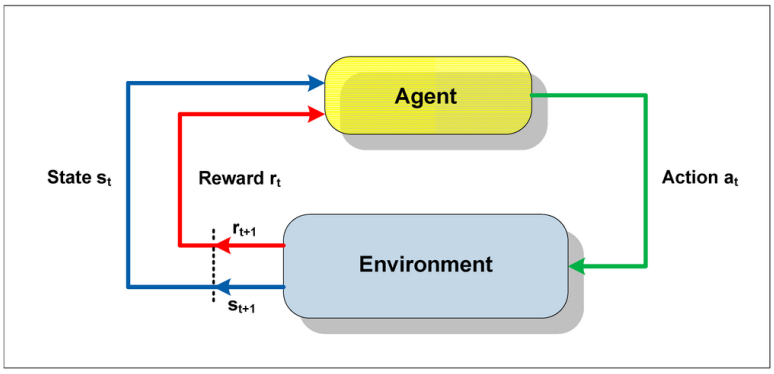

Figure 3.1: Agent-Environment interaction loop.

Figure 3.1 illustrates the RL problem and highlights the interaction between the environment and the agent, underlining the importance of mapping environmental situations into optimal actions to maximize the reward signal.

The RL problem can be formally defined through the following elements:

- **Agent:** is a physical system in the environment that interacts with it through actions, observations and rewards. The actions performed by the agent can influence the state of the environment.

- **Environment:** denoted as ϵ, is the context in which the agent operates, representing the external system with which the agent interacts. It provides to the agent a state or observation s_t at each time step t. The environment can be classified as *completely observable* or *partially observable* depending on the amount of information that the agent can receive from it. If the agent can completely know the state of the environment it is called full. Otherwise, if some information is omitted, it is called partial.

- **Action:** is the decision that the agent can make at each time step t. The set of all possible actions is called action space A. The agent chooses an action a_t at each time step.

- **State:** s_t represents the current observation of the environment to make decisions about the viable actions at the next time step $t + 1$.

- **Reward:** is a scalar that determines how good an action taken by the agent at each time step is. It takes a central role in defining the agent goal: to maximize the cumulative reward in one episode. An episode is a complete sequence of interactions, which begins when the agent is in an initial state

and continues until a predetermined termination condition occurs.

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t \tag{3.1}$$

- **Policy**: π is the strategy or behavior of the agent, mapping states into actions. Finding the policy that maximizes the cumulative reward is the agent's goal and can be represented as a mathematical function or a decision rule $\pi : S \times A \to [0,1]$,$\pi(a, s) = Pr(A_t = a \mid S_t = s)$.
- **Value Function**: is a function that provides the expected cumulative future reward that can be obtained from a certain state-action pair, given the policy π.
- **History:** the term denotes the chronological sequence of observations, actions, and rewards encountered by an agent during its engagement with an environment over a period. The history up to time t is often denoted as $H_t = (O_1, R_1, A_1, ..., O_t, A_t, R_t)$..

In a fully observable environment scenario, reinforcement learning can be formulated as a Finite Markov Decision Process (MDP). MDP is a discrete-time stochastic control process that extends, in the field of optimization problems, the concept of Markov chains. Markov chains are memory-less stochastic processes in which the state of the system at time $t + 1$ depends only on the state at time t, so its evolution is independent of the history of states, this statement represents the *Markov condition* and can be expressed mathematically as:

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0) = P(X_{t+1} = j \mid X_t = i). \tag{3.2}$$

In other words, the probability of moving from one state to another depends only on the current state and the action taken, not on the complete sequence of previous states and actions.
The environment of an RL algorithm is typically stated in the form of a Markov Decision Process (MDP) with a 4-tuple (S, A, P_a, R_a) where:

- S: is the set of states.
- A: is the set of actions.
- P_a: $P_a(s, s') = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the state transition function, and represents the probability of moving from state s to s', given the action a at time t.
- $R_a(s, s')$: is the immediate reward function, and represents the numerical values to state-action pairs assigned after the transition from state s to state s', taking the action a

The goal in a MDP, as for the RL, is therefore to find the optimal policy π^* which maximizes the expected value of cumulative rewards over time, as defined for RL. The agent achieves this by exploring and learning from its interaction with the environment over time. Reinforcement Learning (RL) algorithms, such as Q-learning, Policy Iteration, and Value Iteration, are often used to learn the optimal policy in an MDP.
Therefore Reinforcement Learning is a hard non-linear, non convex optimization problem, where the aim is to optimize the policy $\pi(s, a) = Pr(a = a|s = s)$. In particular, a commonly used method method used for policy evaluation and policy optimization is *dynamic programming*.
Dynamic programming refers to a category of algorithms used to solve optimization problems that relies on dividing the complex problem into more manageable sub-problems and iteratively solving them. Two main approaches are evaluation iteration and policy optimization iteration, both based on the Bellman equation [3].

3.1.1 Q-learning algorithm

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. "Model-free" means that does not require a model of the environment [4].
For any finite Markov decision process, Q-learning finds an optimal policy maximizing the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning converges an optimal policy for any given finite Markov decision process, given infinite exploration time and a partially random policy [5].

To account for the reward associated with an action/state pair, a reward matrix Q is employed, where each row corresponds to a state and each column to an action.
The element $Q(s, a)$ represents the goodness of the pair (state s/action a) and is defined through the *Bellman equation*:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \max_a [Q_\pi(s_{t+1}, a)] \tag{3.3}$$

R is the immediate reward for s_t and a_t, $\gamma \in (0,1)$ is the discount factor that usually is taken near to 1 to favor the next reward, it multiplies the maximum value of reward associated with all the possible actions that can be made at $t + 1$ given that s_t is chosen at t.
Starting from the initial model based on the policy π, which could be random, the Bellman equation is used to enhance the reward matrix Q for each possible state/action pair in an iterative manner until the initial policy π converges to the

optimal policy π^*.

Other techniques used for optimization problems are:

- Differential Programming: It refers to an approach that includes differential calculus in solving optimization problems. In reinforcement learning the calculation of derivatives can be used to update the parameters of a model in order to minimize or maximize an objective function.
- Monte Carlo: this methodology is based on the sampling of complete episodes. The agent performs several interactions within the environment, acquires experiences, and estimates state or action values based on the total rewards obtained.
- Temporal difference: it is a model free method - so it does not require any model of the system - and combines Monte Carlo methods and Dynamic programming. The agent learns by bootstrapping the estimation of the current value of the objective function and updating it by considering the difference between the current estimates and the new information just obtained.
- Exploration/Exploitation: it balances the exploration of the environment (state space) and the exploitation of the knowledge acquired from the previous steps.
- Policy iteration: it is an algorithm whose aim is to find all optimal policies through an iteration loop where first a policy evaluation and then a policy improvement are performed.
- Gradient descent: it is a minimization algorithm based on finding the optimal parameters that minimize the objective function, by computing its gradient.

These techniques are really effective for a fully observable environment, whereas in a partially observable environment a local minimum can be chosen in place of the optimal minimum, hence the initial state distribution plays a crucial role.

3.2 Quantum Reinforcement Learning setting

There are three possible ways to combine quantum and RL frameworks: employing quantum data and classical algorithms, classical data and quantum algorithms or quantum data and algorithms. The last one is almost an unexplored field, hence this work focuses on the former one. Quantum reinforcement learning is the intersection between reinforcement learning and quantum computing, and arises from the need to address challenges associated with quantum computations, such

as the efficient preparation of quantum circuits.
In this thesis, it is presented an overview of the employment of quantum data within classical RL algorithms to explore problem-solving in quantum contexts. This involves modeling complex quantum environments and seeking optimal policies within quantum settings.
State preparation is a very general and fundamental problem in quantum computing. We want to train a agent to design a quantum circuit, which transforms an initial state vector associated to a quantum system into a target state of interest, minimizing the number of quantum gates used. For example to prepare entangled states of N qubits, such as $a|00...0\rangle + b|11...1\rangle$, starting from the initial state $|00...0\rangle$. The control problem can be centered in designing optimal quantum circuits by minimizing the number of quantum gates that can be useful for several reasons in quantum computation to improve reliability, reduce costs and make the implementation of quantum algorithms in experimental and computational reality more feasible. In particular, some of reasons why it is important to minimize the number of quantum gates are explained in the following:

- **Decoherence and errors**, as stated by one of the *DiVincenzo's criteria* , a set of five criteria proposed by David P. DiVincenzo in 2000 [6], representing the features that a quantum computer should possess to be universally usable and practical, minimizing the number of quantum gates is crucial to reduce the probability of errors and maintaining quantum coherence. Indeed each quantum gate, represented by a unitary transformation, adds noise and interference to the circuit.

- **Computational complexity**. Gates increase computational complexity and require more physical resources. The practical implementation of quantum gates often involves significant computational costs and requires the application of complex error correction techniques. Reducing the number of quantum gates helps reduce hardware and software resource requirements, thus making the implementation of quantum algorithms more accessible and feasible. This optimization not only improves computational efficiency, but also reduces the complexity of the necessary error correction strategies, helping to make quantum systems more robust and reliable.

- **Quantum Parallelism**: The intrinsically parallel nature of quantum computation allows many calculations to be performed simultaneously through quantum superposition. However, an excessive number of quantum gates can increase computational complexity, undermining the advantage of parallelism.

We then proceed to deal with a preparation setting for a quantum target state where the policy is focused on minimizing the gates used [7]. Subsequently, other techniques are introduced for the preparation of target states, in which the number

of gates is less crucial, for example in the context of *quantum control*. Here the goal is to prepare states with the highest possible level of fidelity, maximizing the performance of the quantum system, minimizing unwanted effects and ensuring that the system reaches certain states or performs specific desired operations.

3.3 Quantum State preparation minimizing the number of quantum gates

A general N qubit quantum state, as discussed in the previous chapter, can be represented as a superposition of all computational basis states:

$$|\psi\rangle = \sum_{(q_1,q_2,...,q_N)\in\{0,1\}} a_{q_1,q_2,...,q_N}|q_1\rangle \otimes |q_2\rangle \otimes ... \otimes |q_N\rangle \tag{3.4}$$

where $a_{q_1,...,q_N} \in \mathbb{C}$ are the amplitudes for each state, each qubit $q_i \in \{0,1\}$, and the normalization condition is $|| \sum_{(q_1,q_2,...,q_N)\in\{0,1\}} a_{q_1,q_2,...,q_N}||^2 = 1$.
In this setting, amplitudes represent the states of the agent, and in order to use only real numbers, they can be expressed as a couple of real numbers composed by the real and imaginary part, by:

$$\mathbf{s} = [\Re(a_{00...0}), \Re(a_{00...1}), ..., \Re(a_{11...1}), \Im(a_{00...0}), \Im(a_{00...1}), ..., \Im(a_{11...1})] \tag{3.5}$$

It must be considered that the number of terms increases exponentially with the number of qubits N, so that the number of coefficients is 2^{N+1} (taking into account that each coefficient is written as couple of real numbers).
The action consists then in choosing a quantum gate to update the amplitudes from a set of gates. The application of a quantum gate on the state of N qubits, on the k-st qubit, is given by:

$$|\psi'\rangle = (I \otimes \ldots \otimes U \otimes \ldots \otimes I)|\psi\rangle \tag{3.6}$$

where U is the unitary matrix $\in \mathbf{C}^{2^N x 2^N}$ and is in the k-st position of the tensor product.
Hence according to (2.3):

$$|\psi'\rangle = \sum_{(q_1,q_2,...,q_N)\in\{0,1\}} a_{q_1,q_2,...,q_N} U_k|q_1\rangle \otimes |q_2\rangle \otimes \ldots \otimes |q'_k\rangle \otimes \ldots \otimes |q_N\rangle \tag{3.7}$$

For example in case of a quantum state of N=2 qubits, U is defined as:

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$$

and the updates of the amplitudes are computed as follows:
For the first qubit:

$$(U \otimes I) \sum_{(q_1,q_2)\in\{0,1\}} a_{q_1,q_2}|q_1\rangle \otimes |q_2\rangle \tag{3.8}$$

$$\begin{bmatrix} u_{00} & 0 & u_{01} & 0 \\ 0 & u_{00} & 0 & u_{01} \\ u_{10} & 0 & u_{11} & 0 \\ 0 & u_{10} & 0 & u_{11} \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} = \begin{bmatrix} u_{00}a_{00} + u_{01}a_{10} \\ u_{00}a_{01} + u_{01}a_{11} \\ u_{10}a_{00} + u_{11}a_{10} \\ u_{10}a_{01} + u_{11}a_{11} \end{bmatrix} \tag{3.9}$$

While for the second qubit:

$$(I \otimes U) \sum_{(q_1,q_2)\in\{0,1\}} a_{q_1,q_2}|q_1\rangle \otimes |q_2\rangle \tag{3.10}$$

$$\begin{bmatrix} u_{00} & u_{01} & 0 & 0 \\ u_{10} & u_{11} & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} = \begin{bmatrix} u_{00}a_{00} + u_{01}a_{01} \\ u_{10}a_{00} + u_{11}a_{01} \\ u_{00}a_{10} + u_{01}a_{11} \\ u_{10}a_{10} + u_{11}a_{11} \end{bmatrix} \tag{3.11}$$

Then to find the optimal policy the *Q-Learning Alogorithm* is used. In this case the discount factor γ of the cumulative reward is set to 0 or near 0. This reflects the fact that, unlike what was said previously, the agent will place greater importance on immediate rewards rather than on long-term ones. Thus the agent will be more oriented towards maximizing immediate rewards rather than considering future benefits. This is consistent with the fact that we want to reach the target state using the smallest possible number of actions. For the same reason a fully *greedy* approach was chosen, which consists in always choosing the action corresponding to the maximum value of q obtained from the Bellman equation. This choice limits the exploration of the environment and tends to maximize the immediate reward. Alternatively, an *espilon-greedy* approach can be used, in which the agent chooses the optimal action (the one with the maximum value according to its Q function) with a probability $1 - \epsilon$, or choose a random action (explore) with a probability ϵ.

$$a = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg max_a Q(s,a) & \text{with probability } 1-\epsilon \end{cases} \tag{3.12}$$

Given a two-qubit initial state of the type $|\phi\rangle =| 00\rangle$, this setup works well to reach as a target state one of the four Bell states, where the set of gates employed in the algorithm are $CNOT$ plus the rotation operators of an angle $\theta \in \{\pi, \frac{2\pi}{3}, \frac{\pi}{2}, \frac{\pi}{3}, \frac{\pi}{4}\}$ around the three axis $\hat{x}, \hat{y}, \hat{z}$ [7]. At each step, to check if the desired final state is

reached, the fidelity F between the target vector and the final vector is computed: $F(\psi', \psi(t)) = \mid \langle \psi' \mid \psi(t) \rangle \mid^2$. Where $F \in [0,1]$. However $F = 1$ in general is not achievable through a finite set of gates, hence a *tolerance* parameter ζ is defined. A state ψ' is then considered a final state if its fidelity $F \in [1-\zeta,1]$.
Simulation results can be tested on a quantum computer through IBM quantum lab, a cloud service that give access to a real quantum device [8]. Ideally the final state of the simulation should match the one of the real device, but since the second one is affected by noise, a distance measure (fidelity or trace distance) must be used. Therefore it is necessary to reconstruct the quantum state (or the density matrix). To do so, a possible approach is *quantum state tomography* [9] that consists in estimating the density matrix from a set of measured quantities. But the density matrix ρ obtained in this way does not respect the *trace* and *positivity conditions*. To overcome this problem a maximum likelihood technique is adopted. It requires numerical optimization but it is able to produce density matrices that are always non-negative definite. The basic idea is to use the likelihood function:

$$L(\vec{X} \mid \rho) = \prod_{i}^{n} f_\rho(x_i) \tag{3.13}$$

where f can be considered a normal distribution assuming that the noise is Gaussian, and ρ is the density matrix that can be expressed through a tunable matrix T as:

$$\rho = \frac{T^T T}{tr(T^T T)} \tag{3.14}$$

Moreover noise modeling can be included within the algorithm to improve the fidelity and mitigate the effect of noise.

Another promising approach, which has ample room for improvement, for the quantum state preparation is the *Difference-Driven Reinforcement Learning* [10]. The basic idea is to use an *adaptive ϵ-greedy action selection strategy* and a *weighted diversity dynamic reward function*. This approach leads to an increase in convergence speed and an improvement in fidelity for the preparation of a two-qubits system.
In this setting the total reward is divided in internal and external reward $r_t = r_t^{ext} + r_t^{int}$.
The internal reward is set "by considering the differences between the current quantum state, the real next quantum state and the predicted next quantum state and the target quantum state at the same time, and according to this difference, the possible reward for the agent to perform a quantum operation is dynamically set"[10]. The external reward is instead manually set.
The adaptive *epsilon* is updated by introducing a λ parameter as follows:

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) e^{-\frac{\lambda}{C}} \tag{3.15}$$

where C represents the updated cardinality that depends on the task.

3.4 Conclusion

The Machine Learning sector can be referred to as "experimental", since attempting various techniques to see which one works better usually precedes the theoretical analysis. Very complex algorithms often lead to an effective solution to a problem without exploring and fully understanding the logical/theoretical process behind it. Quantum Mechanics as well is a very abstract matter whose mechanisms are beyond everyday experience and thinking processes. The combination of Machine Learning and Quantum Mechanics therefore leads to the deployment of very abstract techniques, that require trial and error approaches as it has been discussed in this thesis.

Bibliography

[1] M. Nielsen and I. Chuang. *Quantum Coputation and Quantum Information*. Cambridge: Cambridge University Press, 2010 (cit. on p. 3).

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge: The MIT Press, 1998 (cit. on p. 21).

[3] Richard Bellman. *Dynamic Programming*. Princeton: Princeton University Press, 1957 (cit. on p. 24).

[4] Shengbo Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Singapore: Springer Verlag, 2023 (cit. on p. 24).

[5] Francisco S. Melo. «Convergence of Q-learning: a simple proof». In: () (cit. on p. 24).

[6] David P. DiVincenzo. «The Physical Implementation of Quantum Computation». In: *https://arxiv.org/abs/quant-ph/00020773* (2000) (cit. on p. 26).

[7] Francesco Montagna. *Quantum-Reinforcement-Learning*. `https://github.com/francescomontagna/Quantum-Reinforcement-Learning`. 2022 (cit. on pp. 26, 28).

[8] *IBM Quantum*. `https://quantum-computing.ibm.com/` (cit. on p. 29).

[9] Daniel F. V. James et al. «Measurement of qubits». In: *Phys. Rev. A.* 64.5 (2001) (cit. on p. 29).

[10] Wenjie Liu et al. «A Quantum States Preparation Method Based on Difference-Driven Reinforcement Learning». In: *SPIN, 2023. 13(03): p. 2350013* () (cit. on p. 29).