UNIVERSIDAD
DE GRANADA

DE NOVO GENOME ASSEMBLY USING QUANTUM ANNEALING

JOSÉ ANTONIO ÁLVAREZ OCETE

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores
Carlos Cano
Antonio Lasanta

FACULTAD DE CIENCIAS
E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 3 de septiembre de 2021

TABLE OF CONTENTS

SUMMARY

In this text, we present a study on how to tackle the de novo genome assembly problem using quantum annealing.

In the first chapter, we develop a quantum mechanics model from a mathematic and quantum computing perspective. The quantum mechanics postulates are explained one by one through an iterative process: we begin by introducing the mathematical concepts required to understand the postulate. After that, the postulate is introduced and explained. Finally, we explore this postulate's implications from a quantum computing perspective. To finalize the chapter we state and prove the no-cloning theorem, one of the most important theorems in quantum computing.

In the second chapter, the gained based on quantum mechanics allows us to study quantum annealing and adiabatic evolution. These are the physical phenomenons on which D-Wave quantum computers are based. Next, Quadratic Unconstrained Binary Optimization (QUBO) problems are explained in detail, providing thorough examples on how to transform well-known NP-hard problems into QUBO models. Lastly, the D-Wave architectures are explained in-depth: how qubits are implemented, how quantum annealing is used to solve problems with these computers, and how we may embed a QUBO model into these systems.

In the third chapter, the de novo genome assembly problem is tackled. The problem is defined and then transformed into a traveling salesman problem. Using the heavy machinery developed in the previous chapter we are able to transform the de novo genome assembly problem into a QUBO, which is then embedded into a quantum annealer. Finally, a series of experiments using D-Wave quantum systems are exposed, using them to solve the genome assembly problem.

Keywords: quantum mechanics model, quantum annealing, quantum optimization algorithms, quadratic unconstrained binary optimization problems, de novo sequencing

1

QUANTUM MECHANICS MODEL

Quantum Mechanics are a mathematical framework in which quantum physics are developed. In this section, we will develop a quantum mechanics model in order to understand quantum computing. The Quantum Postulates will be our guidance. They provide a connection between the physical world and the mathematical formalization. We will provide context and formalization for each postulate, so both the mathematical precision and intuition notions are developed at the same time. This development is based on [1], [2] and [3].

1.1 POSTULATE 1: STATE SPACE

The first postulate sets the environment in which we will operate: The State Space. It will be a Hilbert space associated to a physicial system. Let us rigorously define the necessary concepts using the Bra-ket notation. We will start by revisiting the required linear algebra.

1.1.1 *Bra-ket notation*

Let V be a complex vector space. That is, a vector space over $\mathbb{C}$. We will restrict our study to finite complex vector spaces. If z is a vector in V, we will denote its coordinates either as $z = (z_1, z_2, \ldots, z_n)$ or by column notation:

$$z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$$

Since V is a vector space we have two basic operations: (vector) addition and scalar multiplication.

In quantum mechanics, the usual notation is the Dirac's, also known as *bra-ket* notation. In this context, vectors in a complex vector space are denoted as $|\varphi\rangle$ and are known as *kets*. The only exception to this is the zero vector, which will be denoted as $0 = (0, \dots, 0)$ instead of $|0\rangle$ since $|0\rangle$ will be used as something completely different. A *vector subspace* W of V is a subset of W closed for addition and scalar multiplication.

A *base* of a vector space is a set of vectors $|v_1\rangle, \dots, |v_n\rangle$ such that they are linearly independent and any given vector $|v\rangle$ can be written as a linear combination of them: $|v\rangle = \sum_{i=1}^{n} \alpha_i |v_i\rangle$. The *dimension* of a vector space is the number of elements in any of its bases, which is independent from the chosen base.

Definition 1. Given two complex vector spaces V and W, a *linear operator* is an application $M : V \to W$ that is linear in its inputs:

$$M\Big(\alpha|u\rangle + \beta|w\rangle\Big) = \alpha M(|u\rangle) + \beta M(|w\rangle)$$

If V to W have dimensions n and m respectively, there is a bijection between the operators from V to W and the n by m matrices. Given an operator M, the obtained matrix M' is called the *matrix representation* of the linear operator. Furthermore, $M(|u\rangle) = M' \cdot |u\rangle$, so we usually denote the linear operator and its matrix representation by the same letter, and $M(|u\rangle)$ simply as $M|u\rangle$.

We will usually refer to linear operators simply as *operators*.

1.1.2 *Inner product and Hilbert Spaces*

Let us define another operation within the complex vector spaces.

Definition 2. Let V be a complex vector space. An inner product $\langle \cdot | \cdot \rangle : V^2 \to \mathbb{C}$ is a function such that:

1) $\langle \cdot | \cdot \rangle$ is sesquilinear. That is,

   1.1) $\langle \cdot | \cdot \rangle$ is conjugate symmetric: for all u, v in V, $\langle u|v\rangle = \overline{\langle v|u\rangle}$.

   1.2) $\langle \cdot | \cdot \rangle$ is linear on the second variable: for all u, v, w in V and α, β in $\mathbb{C}$:

$$\langle u|\alpha v + \beta w\rangle = \alpha\langle u|v\rangle + \beta\langle u|w\rangle$$

2) $\langle \cdot | \cdot \rangle$ is definite positive. That is, for all u in V, $\langle u|u\rangle \geq 0$ and $\langle u|u\rangle = 0 \iff v = 0$.

Given this properties it can easily be proven that $\langle \cdot | \cdot \rangle$ is also conjugate linear on the first variable. That is, for all u, v, w in V and α, β in $\mathbb{C}$:

$$\langle \alpha u + \beta v|w\rangle = \overline{\alpha}\langle u|w\rangle + \overline{\beta}\langle v|w\rangle$$

We will sometimes denote the inner product $\langle \cdot | \cdot \rangle$ as $(\cdot, \cdot)$ to simplify notation.

Two vectors are said to be *orthonormal* if their inner product is zero. We define the norm of a vector $|v\rangle$ by:

$$\| \, |v\rangle \, \| = \sqrt{\langle v|v\rangle}$$

A *unit vector* is a vector $|v\rangle$ such that $\| \, |v\rangle \, \| = 1$. We also say that $|v\rangle$ is *normalized*, and we can normalize any vector except the zero vector by dividing it by its norm.

A base $|v_1\rangle, \ldots, |v_n\rangle$ is said to be *orthonormal* if every vector is a unit vector and they are pairwise orthogonal. That is, $\langle v_i | v_j \rangle = \delta_{ij}$ where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Definition 3. An *inner product space* is a vector space with an associated inner product. A **Hilbert Space** is an inner product space that is also complete.

Hausdorff's Theorem states that every finite normed space is complete, therefore every finite inner product space over $\mathbb{C}$ is a Hilbert space [4]. Again, by Hausdorff's theorem, we know that every n dimensional Hilbert space is isomorphic to $\mathbb{C}^n$. Thus, $\mathbb{C}^n$ is the canonical n dimensional Hilbert space. Our study will be focused on these spaces.

Let $\alpha = a + i \cdot b \in \mathbb{C}$. We define the *conjugate*, $\bar{\alpha}$, as $\bar{\alpha} = a - i \cdot b$. The canonical inner product in $\mathbb{C}^n$ is:

$$\langle u|v\rangle \; = \sum_{i=1}^{n} \overline{u_i} v_j$$

where $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$, for every u, v in $\mathbb{C}^n$.

1.1.3 *Postulate 1 statement*

The reader should be familiar by now with the notation and the necessary linear algebra to formulate the first postulate.

> **Postulate 1.** Associated to any isolated physical system is a complex vector space with an inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

An important concern with this postulate is that it does not tell us which is the state space of a given system, nor its state vector. Although up to this point we cannot formally assure this, in quantum computing the state space will be fixed: $\mathbb{C}^{2^n}$ for an n-qubits system. Our evolving state vector will be a vector 2^n-vector.

Let us start by modulating a simpler system: a single qubit system.

Definition 4. A state vector of the state space $\mathbb{C}^2$ is called a **qubit**. Thus, $\mathbb{C}^2$ may be called a single qubit state space.

Suppose $|0\rangle$, $|1\rangle$ form an orthonormal basis of a 2-dimensional Hilbert space. Then, any state vector in this state space may be described as:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α, β are complex numbers called *amplitudes*. Thus, the condition that $|\varphi\rangle$ is a unit vector, $\langle\varphi|\varphi\rangle = 1$ is equivalent to $|\alpha|^2 + |\beta|^2 = 1$. This is known as the *normalization condition*.

We will always think of $|0\rangle$, $|1\rangle$ as a previously fixed orthonormal base. A linear combination of state vectors $\sum_i a_i|\varphi_i\rangle$ is called a *superposition* of the states $|\varphi_i\rangle$ with amplitudes a_i respectively. For example, the state

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

is a superposition of the states $|0\rangle$ and $|1\rangle$ with amplitudes $1/\sqrt{2}$ and $-1/\sqrt{2}$ respectively.

1.1.4 *Quantum Computation Perspective: The Quantum Bit*

The bit is the minimum measure of information on classical computation and classical information theory. Everything in these fields is built from scratch based on bits. Likewise, quantum computing and quantum information theory are built upon the **qubit**.

We have reached the qubit definition from quantum physics and pure mathematics, describing the qubit as a mathematical object independent of its physical implementation. By describing them as mathematical entities we will be able to explore their properties mathematically without having to worry about the physics underneath. This allows us to construct the quantum computing and quantum information theories independently of the physical implementation.

So, intuitively, what is a qubit? Just like the classical bit, a qubit has a state. For the bit, the two only possible states are either 0 or 1. A qubit can take the states $|0\rangle$ and

$|1\rangle$ -corresponding to the classical states 0 and 1- or it can be in a *linear combination* of them:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where α and β are complex numbers. Thus, we can describe a qubit as a vector in a two-dimensional complex Hilbert space (the canonical $\mathbb{C}^2$), where $|0\rangle$ and $|1\rangle$ form an orthonormal basis called the *computational basis*. $|0\rangle$ and $|1\rangle$ will be called *computational basis states*.

At this point, the reader may ask themselves if a qubit may even physically exist, not just as a mathematical entity. After all, the first postulate states that given a *physical system*, there is an associated state space and vector states that describe the system. However, we define a qubit from state space ($\mathbb{C}^2$) without considering a physical system.

The answer is positive: there are numerous physical systems such that their associated state spaces are ($\mathbb{C}^2$). Thus, modeling a qubit. More intuitive examples are provided in section 1.2.4 and precise physical implementations are briefly discussed in section [TODOref].

1.2 POSTULATE 2: MEASUREMENT

The second postulate describes how states are 'measured', that is, how an outside observer may look inside the system. In classical physics, consider a simple system of a moving particle. 'Measuring' would be recording, for instance, the particle mass and speed at a given time. That is, someone **outside** the system would look **into** the system to record some information. Lastly, in the classical computation model measuring a bit is simply retrieving its content.

In quantum physics, measuring has some unexpected and sometimes counter-intuitive properties. Let us provide some linear algebra context before formulating the second postulate.

1.2.1 *Outer product*

Definition 5. Let V, W be two vector spaces and $|v\rangle \in V, |w\rangle \in W$. We define the *outer product* between $|v\rangle$ and $|w\rangle$, $|w\rangle\langle v|$, as the only linear operator such that for any $|v'\rangle \in V$,

$$(|w\rangle\langle v|)\ |v'\rangle = |w\rangle\langle v|v'\rangle = \langle v|v'\rangle|w\rangle$$

These identities provide a dual interpretation: the already known product of a complex value $\langle v|v'\rangle$ with a vector $|w\rangle$, and the application of the new operator, the outer product $|w\rangle\langle v|$ to the vector $|v'\rangle$. The outer product is defined so that this duality occurs.

Let us consider linear combinations of outer products. By definition, $\sum_i a_i|w_i\rangle\langle v_i|$ is the operator that transforms $|v'\rangle$ into $\sum_i a_i|w_i\rangle\langle v_i|v'\rangle = \sum_i a_i\langle v_i|v'\rangle|w_i\rangle$.

The most important result concerning outer products is the *completeness relation*:

Proposition 1 (Completeness relation)**.** *Let $|i\rangle$ be any orthonormal basis of a finite vector space V. Then:*

$$\sum_i |i\rangle\langle i| = I$$

Proof. Let $|v\rangle \in H$. $|v\rangle$ can be expressed as $\sum_i v_i|i\rangle$ for some complex numbers v_i. Notice that $\langle i|v\rangle = v_i$. Therefor:

$$|v\rangle = \sum_i v_i|i\rangle = \sum_i \langle i|v\rangle|i\rangle = \sum_i |i\rangle\langle i|v\rangle = \left(\sum_i |i\rangle\langle i|\right)|v\rangle$$

Since $|v\rangle$ was arbitrary, this proves that $\sum_i |i\rangle\langle i| = I$. □

Corollary 1 (Cauchy-Schwarz inequality)**.** *For any two vectors $|v\rangle, |w\rangle$ in a Hilbert space,*

$$|\langle v|w\rangle|^2 \leq \langle v|v\rangle\langle w|w\rangle$$

where the equality occurs if and only if $|v\rangle$ and $|w\rangle$ are linearly dependant

Proof. We provide a proof supposed that our Hilbert space is finite.

Let $|v\rangle, |w\rangle$ be two vectors of a finite Hilbert space H. Since H is finite, using the Gram-Schmidt procedure we may obtain a basis $|i\rangle$ where the first vector is $|w\rangle/\sqrt{\langle w|w\rangle}$. Using the completeness relation $\sum_i |i\rangle\langle i| = I$:

$$\langle v|v\rangle\langle w|w\rangle = \langle v|I|v\rangle\langle w|w\rangle = \langle v|\sum_i(|i\rangle\langle i|)|v\rangle\langle w|w\rangle = \sum_i \langle v|i\rangle\langle i|v\rangle\langle w|w\rangle$$

The sum of a list of positive numbers is obviously greater than its first element, so:

$$\sum_i \langle v|i\rangle\langle i|v\rangle\langle w|w\rangle \geq \frac{\langle v|w\rangle\langle w|v\rangle}{\langle w|w\rangle}\langle w|w\rangle = \langle v|w\rangle\langle w|v\rangle = |\langle w|v\rangle|^2$$

Lastly, the equality occurs if and only if $\langle v|i\rangle = 0$ for every $|i\rangle \neq |w\rangle/\sqrt{\langle w|w\rangle}$. But since $|i\rangle$ is a base, this means $|v\rangle$ and $|w\rangle$ are linearly dependent. □

1.2.2 *Unitary and Hermitian operators*

Another way of looking at the inner product is the *adjoint*.

Definition 6. Let A be an operator between $\mathbb{C}^n$ and $\mathbb{C}^m$, finite dimensional Hilbert spaces. That is, $A \in \mathcal{M}_{n\times m}(\mathbb{C})$. Then, its *adjoint* or *conjugate transpose* $A^\dagger$ is defined by:

$$(A^\dagger)_{ij} = \bar{A}_{ji}$$

If $|v\rangle$ is a vector, we can compute its adjoint by seeing it as a matrix. By convention, we will denote $|v\rangle^\dagger = \langle v|$. Adjoints of vector are usually called *bras*, making given sense to the *bra-ket* notation since $\langle v| \cdot |v\rangle = \langle v|v\rangle$, where $\cdot$ denotes the dot product.

Some useful algebraic identities associated to adjoints are:

- Given an operator $A \in \mathcal{M}_n(\mathbb{C})$, $A^\dagger$ is the only operator such that for any two vectors $|u\rangle, |v\rangle \in \mathbb{C}^n$: $(|u\rangle, A|v\rangle) = (A^\dagger|u\rangle, |v\rangle)$
- For any two operators A, B, $(AB)^\dagger = B^\dagger A^\dagger$.
- As a corollary, for any vector $|v\rangle$ and for any operator A, $(A|v\rangle)^\dagger = \langle v|A^\dagger$.

Definition 7. An operator A is said to be *normal* if $AA^\dagger = A^\dagger A$.

Characterization of normal operators is provided in Theorem 1. There are two particular cases of normal operators that will be of special interest to us:

Definition 8. An operator A is said to be *Hermitian* if its adjoint is itself: $A^\dagger = A$.

Definition 9. A matrix U is said to be *unitary* if $UU^\dagger = I$. Similarly, an operator is said to be *unitary* if $UU^\dagger = I$. A unitary operator U also fulfills that $U^\dagger U = I$.

Clearly, Hermitian and unitary operators are also normal. The importance of unitary matrices and operators in quantum computing lies in the following

Proposition 2. *Unitary operators preserve inner product between vectors. Thus, they also preserve the norm of a vector.*

Proof. Let $|u\rangle, |v\rangle \in \mathbb{C}^n$ and $U \in \mathcal{M}_n(\mathbb{C})$ be a unitary operator. Then:

$$(U|u\rangle, U|v\rangle) = \langle u|U^\dagger U|v\rangle = \langle u|I|v\rangle = \langle u|v\rangle = (|u\rangle, |v\rangle)$$

Which proves the proposition. □

An important type of Hermitian operators is the projectors.

Definition 10. Let W be a k-dimensional subspace of the d-dimensional vector space V. Let $|1\rangle, \ldots, |d\rangle$ be an orthonormal base of V where $|1\rangle, \ldots, |k\rangle$ is an orthonormal base of W. The *projector* onto the subspace W is defined by:

$$P = \sum_{i=1}^{k} |i\rangle\langle i|$$

It can easily be shown that this definition is independent from the chosen base $|1\rangle, \ldots, |k\rangle$. Since $|v\rangle\langle v|$ is Hermitian for any vector $|v\rangle$, P is also Hermitian: $P = P^\dagger$. We will often refer to the subspace onto which P projects simply as P for comodity. The *orthonormal completement* of P is $Q \equiv I - P$. It can be verified the vector subspace Q is spanned by the base $|k+1\rangle, \ldots, |d\rangle$.

As the reader may already imagine, n-qubits systems will be represented as vector states or *certain* state spaces. That is, as unitary vectors of certain Hilbert spaces. Thus, our definition of transformations on qubits must preserve their norm. These will be the qubits gates, which will be represented as unitary operators.

On the other hand, Hermitian operators will be key in order to study how a quantum system evolves with time using the Schrodinger equation in the third Postulate.

Finally, projectors will be used on as particular way of measurement and will be further discussed in section 1.4.3, once systems with multiple qubits have been introduced.

1.2.3 *Postulate 2 statement*

As previously discussed, the second postulate describes how a quantum system may be measured.

> **Postulate 2.** Quantum measurement are described by a collection $\{M_m\}$ of *measurement operators*. These act on the state space associated to the physical system being measured. The index m refers to the measurement outcomes that may occur. That is, if $|\varphi\rangle$ is the vector state before measure, then the probability of the result m occurring is:
>
> $$p(m) = \langle\varphi|M_m^\dagger M_m|\varphi\rangle$$
>
> and the state of the system after the measurement is:
>
> $$\frac{M_m|\varphi\rangle}{\sqrt{p(m)}}$$
>
> Finally, the measurement operator satisfy the *completeness equation*:
>
> $$\sum_m M_m^\dagger M_m = I$$

The completeness equation is equivalent to the fact that the probabilities of the different possible outcomes add up to one:

$$\sum_m p(m) = \sum_m \langle\varphi|M_m^\dagger M_m|\varphi\rangle = \langle\varphi|\sum_m (M_m^\dagger M_m)|\varphi\rangle = \langle\varphi|\varphi\rangle = 1$$

which holds for every state vector since they are unitary. Reciprocally, this equation occurring for every state vector $|\varphi\rangle$ implies the completeness equation.

The reader may have already noticed a huge difference between classic and quantum measurement: The measured state **changes** after the measurement. This means that we interfere with the system by merely looking into it! It will ultimately translate into huge differences between classical and quantum computing, such that we will generally not be able to clone a qubit (see theorem 3).

Let us look at an important example: *measurement of a qubit on the computational basis*. That is, measuring a qubit with two possible outcomes: $|0\rangle$ and $|1\rangle$. Although this is a particular case of projective measurement (further explained in section 1.4.3), it is worth introducing it now to deepen our understanding of qubits.

To obtain such results we use the operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. In order for $\{M_0, M_1\}$ to be a correct collection of measurement operators they must satisfy the completeness equation. Observe that each operator is Hemitian: $M_i^\dagger = (|i\rangle\langle i|)^\dagger = (\langle i|)^\dagger(|i\rangle)^\dagger = |i\rangle\langle i| = M_i$. Furthermore, $M_i^2 = |i\rangle\langle i|i\rangle\langle i| = |i\rangle\langle i| = M_i$.

Finally, the computational basis is an orthonormal basis and therefore the completeness relation tells us that:

$$\sum_i |i\rangle\langle i| = 1$$

We can see that the completeness equation holds:

$$\sum_i M_i^\dagger M_i = \sum_i M_i^2 = \sum_i M_i = \sum_i |i\rangle\langle i| = 1$$

Let's measure using this operators, also called *measure in the computational basis*, to better understand the measurement. Suppose the state being measured is $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$. Then, the probability of obtaining the outcome 0 is:

$$p(0) = \langle\varphi|M_0^\dagger M_0|\varphi\rangle = \langle\varphi|M_0|\varphi\rangle = |a|^2$$

Similarly, the probability of obtaining the outcome 1 is $p(1) = |b|^2$. Naturally, $|a|^2 + |b|^2 = 1$. What happens after measuring? The post-measurement state will be, respectively if we measured 0 or 1:

$$\frac{M_0|\varphi\rangle}{|a|} = \frac{a}{|a|}|0\rangle$$

$$\frac{M_1|\varphi\rangle}{|b|} = \frac{b}{|b|}|1\rangle$$

Factors like $a/|a|$ are known as *phases*. An importan result of quantum mechanics assures that multiplying by factors like these does not affect the state vector [1], so we virtually obtained $|0\rangle$ and $|1\rangle$. We may appreciate now how dividing by $\sqrt{p(m)}$ in the post-measurement state is only done so the resulting vector is a unit vector.

So if we measured a o, the post-measurement vector will be $|0\rangle$ and vice-versa. Any measurements performed after the first one will yield exactly the same result. This behavior is called *qubit collapsing*.

We defined measurement to be independent of any basis so we may measure in the most convenient basis at each point. For example, we may use the following other basis:

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

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

It can easily be proven that $\{|+\rangle, |-\rangle\}$ is a basis and therefore it satisfies the completeness equation. In fact, the operators $M_+ = |+\rangle\langle+|$ and $M_- = |-\rangle\langle-|$ satisfy the completeness equation. Using this operators, measuring the qubit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ will output $+$ with probability:

$$p(+) = \langle\varphi|M_+^\dagger M_+|\varphi\rangle = \langle\varphi|M_+|\varphi\rangle = \langle\varphi|+\rangle\langle+|\varphi\rangle = \langle\varphi|+\rangle^2 =$$

$$= \left(\frac{\alpha}{\sqrt{2}}\langle 0|0\rangle + \frac{\alpha}{\sqrt{2}}\langle 1|0\rangle + \frac{\beta}{\sqrt{2}}\langle 0|1\rangle + \frac{\beta}{\sqrt{2}}\langle 1|1\rangle\right)^2 = \frac{(\alpha+\beta)^2}{2}$$

$$p(-) = \frac{(\alpha-\beta)^2}{2}$$

And, naturally, $p(+) + p(-) = 1$. So $|\varphi\rangle$ will collapse to $|+\rangle$ with probability $(\alpha + \beta)^2/2$ when measured with this operators. Observe that the post-measurement state will never be $|0\rangle$ nor $|1\rangle$ in this case, we made the qubit collapse to the chosen state basis.

Finally, there is a technical fineness between the first and second postulates worth mentioning. The first postulate was stated for an *isolated* physical system, but by measuring the system we interfere with it. However, measuring devices are also quantum systems, so together the measured and the measuring system form a larger isolated system (it may be necessary to include more quantum systems, but this can be done).

1.2.4 *Real life qubit examples*

In classical computation, we may know the state of a bit by consulting it. That is, what can simply retrieve that information from the bit. The first difficulty we find in quantum computing is that once we *measure* a qubit it *collapses* to either $|0\rangle$ with probability $|\alpha|^2$, or to $|1\rangle$ with probability $|\beta|^2$. The obtained output reflects the qubit state *after* it has collapsed to either one of these states, so the outcome may only be either $|0\rangle$ or $|1\rangle$. This means that we may never retrieve directly the values α and β. Thus, being unable to clone a qubit. This is the main idea behind the no-cloning theorem 3.

We can, however, initialize qubits in a certain state and apply some operations to them in order to alter their coefficients, thus knowing their exact value. However, once a single measurement is done, the qubit collapses and the α and β values are 'lost'.

Superposition and collapsing might be counter-intuitive concepts, so let us look at them with an analogy. We can think of a perfect coin being tossed as the following qubit:

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

This does **not** represent a coin that has landed somehow on its side, but a spinning coin that has not landed yet. Upon measuring it, we 'make the coin land' and see the result: either heads or tails, and neither of the states in between. This example also describes the qubit collapse: once the coin has landed, we will see the same result every time we look at it -obviously-, just like every time a qubit is measured after the first measurement, the outcome will be the same since it has already collapsed. We will return to this state, also known as the Bell state, in section 1.4.2.

On the other hand, this was quite an inaccurate example since the system is not really isolated, although it was interesting intuition-wise. One of the first (accurate) qubit models ever proposed was the Schrodinger's Cat [5] [6]. In this hypothetical experiment, a cat would be locked in a room for an hour with a device that during that hour would *perhaps* trigger, killing the cat. On the other hand, with equal probability, it would not trigger at all. After the whole hour elapses, the cat would be alive and dead with equal probability, ending up in a halfway state. In this case, our computational bases would be the states alive and dead, and we achieve the state $|+\rangle$ after

that hour. Once we open the room and check on the cat, our qubit collapses to either state and stays on it until further disturbance.

Although physical implementations are discussed in section [TODOref chapter], we cannot proceed any further without providing a more accurate and reproducible description of a qubit than 'a coin being tossed' and such a hypothetical cat experiment. A possible realization of a qubit is an electron in a single atom's orbit, as seen in Figure 1. An electron in an orbit may be in the so-called *ground* and *excited* states, $|0\rangle$ and $|1\rangle$ respectively, depending on its energy. By shining light to the electron with a certain energy and for a certain amount of time, one can make the electron move from the ground state to the excited state and vice versa. But most interestingly, one can apply the light to the electron during a smaller amount of time, moving the electron somehow 'halfway' between both states.

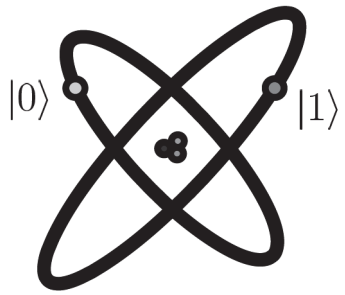


Figure 1: Qubit represented by two electron orbits in an atom, [1].

1.3 POSTULATE 3: EVOLUTION

The third postulate of quantum mechanics modulates the evolution of a quantum system. That is, the evolution of the state vector that describes the system. In order to properly formalize it, the concepts of eigenvectors, eigenvalues, and Hermitian operators are required.

1.3.1 *Eigenvalues and eigenvectors*

Definition 11. Let V be a vector space and A an operator on V. An *eigenvector* is a non zero vector $|v_\lambda\rangle$ such that $A|v_\lambda\rangle = \lambda|v_\lambda\rangle$ for a complex value λ called the associated *eigenvalue*.

Eigenvalues and their associated eigenvectors will usually be denoted with the same letter for simplicity: λ and $|\lambda\rangle$. We assume the reader is familiar with eigenvectors and values basic notions. For instance, that they may be calculated using the *characteristic equation*: $|I - \lambda A| = 0$.

Definition 12. A *diagonal representation* of an operator A is a representation $\sum_i \lambda_i |\lambda_i\rangle\langle\lambda_i|$ where the $|\lambda_i\rangle$ form an orthonormal set of A's eigenvectors and λ_i are the respective eigenvalues. An operator is said to be *diagonalizable* if it allows a diagonal representation. The *spectrum* of an operator is the set of its eigenvalues.

Example 1. As an example of this, let us consider the following matrix:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

This matrix is called the Z *Pauli* matrix. It is relevant for quantum computing and it will be introduced later on along with the rest of the Pauli matrices. For now, Let's compute its diagonalizable representation. Since it is already diagonal we can infer that its eigenvalues are $\{1, -1\}$. Computing the diagonal representation we realize that a pair orthonormal eigenvectors are $\{|0\rangle, |1\rangle\}$ respectively. Therefore:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Normal operators have significant relevancy thanks to the following result:

Theorem 1 (Spectral Decomposition Theorem)**.** *An operator A is normal if and only if it is diagonalizable.*

Proof. TODO: To be copied, Box 2.2, page 72, Nielsenchen. □

Since Hermitian and unitary operators are normal, it follows the next

Corollary 2. *Any Hermitian operator is diagonalizable. Any unitary operator is diagonalizable.*

1.3.2 *Postulate 3 statement*

> **Postulate 3.** The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state $|\varphi\rangle$ of the system at time t_1 is related to the state $|\varphi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,
>
> $$|\varphi'\rangle = U|\varphi\rangle$$

Just like the first postulate does not provide the state space or state vector of the system, the third postulate does not provide the unitary transformation that concretes this evolution. For our quantum computing case, we will be the ones to define the

unitary transformation to the system. That is, the quantum circuit that transforms our qubit.

Example 2. Let's consider the *X Pauli matrix*, also known as the *bit flip* matrix:

$$\sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

It is called the bit flip matrix because it takes $|0\rangle$ to $|1\rangle$ and vice-versa:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

This product that we just computed is precisely what postulate 3 states: the evolution of our state vector following the unitary operator X: $X|0\rangle = |1\rangle$. Even though for an arbitrary system we do not know the specific unitary transformation the system follows, we can create systems that follow certain desired transformations. These are the basics of quantum gates and quantum circuits.

The description of the system evolution provided by Postulate 3 only bears information for those fixed times t_1 and t_2. A continuous time-description of this evolution is provided by the Schrodinger equation, which provides a redefinition of the second postulate.

> **Postulate 3'.** The time evolution of the state of a *closed* quantum system is described by the Schrodinger equation:
>
> $$i\hbar \frac{d|\varphi\rangle}{dt} = H|\varphi\rangle$$
>
> where $\hbar$ is *Planck's constant*, i is the imaginary unit and H is a fixed Hermitian operator known as the *Hamiltonian*.

There are several notes to make about this postulate. First, the Hamiltonian is fixed for the given system and it is not be confused with the *Hadamard quantum gate*, also represented by an H. Second, $\hbar$ is a physical constant that can be absorbed into the Hamiltonian for our purposes, simplifying the equation. Finally, this is a differential equation, so by knowing the initial state space of the system and the exact Hamiltonian we may know the exact evolution of the system.

Let us study the Hamiltonian in general. Since it is a Hermitian operator, it allows a spectral decomposition by theorem 1:

$$H = \sum_E E|E\rangle\langle E|$$

where E are the eigenvalues and $|E\rangle$ the respective normalized eigenvectors. the states $|E\rangle$ are usually referred to as *energy eigenstates* or *stationary states*, and E is the *energy* of the state $|E\rangle$. Furthermore, the lowest energy is called the *ground energy state* while the corresponding eigenstate is called the *ground state*.

Example 3. Suppose a single qubit system has a the following Hamiltonian:

$$H = \hbar\omega X = \hbar\omega \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Where X is the first Pauli matrix, and ω is a positive parameter. H eigenenergy states are the same as X eigenstates: $(|0\rangle + |1\rangle)/2$ and $(|0\rangle - |1\rangle)/2$ with respective energies $\hbar\omega$ and $-\hbar\omega$. Thus, the ground state is $(|0\rangle - |1\rangle)/2$ with ground state energy $-\hbar\omega$.

Let us deduce the connection between the Hamiltonian perpestive of dinamics, Postulate 3', and the unitary operator perspective, postulate 3. We can solve the Schrodinguer equation, which can be proven to be:

$$|\varphi(t_2)\rangle = \exp\left(\frac{-iH(t_2 - t_1)}{\hbar}\right)|\varphi(t_1)\rangle$$

We define:

$$U(t_1, t_2) \equiv \exp\left(\frac{-iH(t_2 - t_1)}{\hbar}\right)$$

Which is a unitary operator. In fact, any unitary operator U may be expressed as $U = \exp(\mathrm{iK})$ for some Hermitian operator K. Thus, we obtained:

$$|\varphi(t_2)\rangle = U(t_1, t_2)|\varphi(t_1)\rangle$$

Following this procedure, we have proven that there is a one to one correspondence between the continuous time-varying postulate 3' using the Hamiltonian and the more stationary discrete-time version using the unitary operator. Although the discrete-time vision is usually used in quantum computing, Quantum Annealing -the specific application of quantum mechanics used in this thesis- rests mostly on the Hamiltonian point of view. We will deepen in this architecture on section [TODOref].

It is worth mentioning that both versions of this postulate assume our physical system to be *closed*. That is, there is no interaction with the system coming from the exterior. In reality, the only real closed system is the universe as a whole. However, we may recreate sufficiently closed systems so that they can be described with approximations as being closed.

Furthermore, this severely interferes with postulate 2, where an outsider to the system may interfere with it by measuring it. The quantum system will evolve following postulate 3 until measurement is applied. Then they will evolve following the behavior described on postulate 2.

In practice, obtaining the Hamiltonian for a given quantum system is a really laborious work and usually needs experimental data [TODO: add evidence? stated as such in Niel.]. However, for our computational purposes, we will be the ones designing the Hamiltonian such that our system evolves as desired. In particular, chapter [TODOref to QUBO chapter] describes in detail the construction of Hamiltonians for QUBO problems.

1.3.3 *Quantum Computing perspective: Quantum Gates*

Although quantum annealing does not make use of quantum gates, they are the basis of quantum computing and they fully rely on postulate 3. Because of their importance, a brief overview of quantum gates is provided o¡in this section. However, a profound understanding of them will not be necessary to understand the rest of the thesis.

Quantum computing is built upon the most simple operation we can compute on single qubits: quantum gates. Since state vectors are unit two-dimensional vectors, the operations we apply to them must preserve the norm. Thus, quantum gates will be represented by 2x2 unitary matrices. We have already introduce some of the most famous gates, the Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Some of the others most important gates are the Hadamard gate (denoted as H), the phase gate (denoted as S), and the $\pi/8$ gate (denoted as T):

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}; \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}; \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

Although we have not described the fundamentals of multiple qubits systems (postulate 4 is required for this), it is worth mentioning a simple two-qubits gate for the purpose of this section. Just like in classical computing, conditional operations are essential for the construction of complex quantum algorithms. The most basic conditional operation is the *controlled-NOT* gate, also referred to as the CNOT gate. This gate takes two bits $|c\rangle|t\rangle$, a control qubit, and a target qubit. It flips the target qubit if the first qubit is set to $|1\rangle$, producing the operation $|c\rangle|t\rangle \rightarrow |c\rangle|c \oplus t\rangle$, where $\oplus$ denotes the exclusive-or operation. Therefore, if the control qubit is $|c\rangle = \alpha|0\rangle + \beta|1\rangle$,

it will virtually flip the target qubit with probability $|\beta|^2$ and leave it how it is with probability $|\alpha|^2$.

One of the most important results of classical computation theory is that any boolean function may be constructed by only using AND, OR, and NOT gates, or by simply using NAND gates [7]. There is an equivalent result concerning quantum gates [3].

Theorem 2. *Any unitary matrix can be approximated by a combination of the Hadamard, CNOT, and $\pi/8$ gates.*

In this case, {H, CNOT, T} is called a *universal gate set*. In practice, is it efficient to build such any gate using only this gate set? The answer to the question is positive, as stated by the **Solovay-Kitaev theorem** [8]. However, this is out of the scope of this project. We provide a sketch proof of the previous theorem from [3].

Proof. TODO: copy proof from Bayens, Theorem 4.2 □

Further study of quantum circuits falls out of the scope of this thesis since we will use the D-Wave architecture, which relies on quantum annealing instead of quantum circuits.

1.4 POSTULATE 4: COMPOSITE SYSTEMS

In this section, we introduce the last postulate, which lets us understand the state space associated with a physical system composed of other minor systems. This will allow us to study multiple-qubits systems and state some of the most important results in quantum mechanics and quantum computing: The *Heisenberg uncertainty principle* and the no-cloning theorem.

1.4.1 *Tensor product*

Let V and W be complex vector spaces with dimensions m and n respectively. Then, $V \otimes W$, read 'V tensor W', is a mn complex vector space. Let $|v\rangle$ and $|w\rangle$ be vectors in V and W respectively. Then, $|v\rangle \otimes |w\rangle$ is in $V \otimes W$. Furthermore, any element of $V \otimes W$ may be expressed as a linear combinations of tensor products $|v\rangle \otimes |w\rangle$ of elements from V and W. We may describe elements of $V \otimes W$ using the following equivalent notations: $|v\rangle \otimes |w\rangle$, $|v\rangle|w\rangle$, $|v, w\rangle$ and even $|vw\rangle$.

Let $|i\rangle$ and $|j\rangle$ be basis for V and W respectively. Then, $|i\rangle \otimes |j\rangle = |ij\rangle$ is a basis for $V \otimes W$. For example, consider the complex vector space $\mathbb{C}^2$. Then

$$|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle = |00\rangle + |11\rangle$$

is an element in $\mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^4$.

Although the tensor product is generally described as an abstract construct, we may provide a more visual perspective using the *Kronecker product*. This is a matrix representation for the tensor product of finite vector spaces. Suppose $A = \{a\}_{ij}$ is a $n \times m$ matrix and $B = \{b\}_{ij}$ is an $p \times q$ matrix. Then $A \otimes B$ is a $np \times mq$, matrix with the following representation:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}B & a_{n2}B & \dots & a_{nm}B \end{pmatrix}$$

For example:

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \otimes \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} = \begin{pmatrix} 0 \times \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} & 1 \times \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} \\ 2 \times \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} & 3 \times \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} \end{pmatrix} =$$

$$\begin{pmatrix} 0 \times 10 & 0 \times 20 & 1 \times 10 & 1 \times 20 \\ 0 \times 30 & 0 \times 40 & 1 \times 30 & 1 \times 40 \\ 2 \times 10 & 2 \times 20 & 3 \times 10 & 3 \times 20 \\ 2 \times 30 & 2 \times 40 & 3 \times 30 & 3 \times 40 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 10 & 20 \\ 0 & 0 & 30 & 40 \\ 20 & 40 & 30 & 60 \\ 60 & 80 & 90 & 120 \end{pmatrix}$$

By definition, the tensor product satisfies:

- Let α be a complex number. For any $|v\rangle$ in V and $|w\rangle$ in W,
$$\alpha(|v\rangle \otimes |w\rangle) = (\alpha|v\rangle) \otimes |w\rangle = |v\rangle \otimes (\alpha|w\rangle)$$
- For any $|v_1\rangle$, $|v_2\rangle$ in V and $|w\rangle$ in W,
$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$$
- For any $|v\rangle$ in V and $|w_1\rangle$, $|w_2\rangle$ in W,
$$|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$$

The inner products of the spaces V and W may be use to extend and natural inner product the tensor space. Define:

$$\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle, \sum_j b_j |v'_j\rangle \otimes |w'_j\rangle \right) \equiv \sum_{ij} \overline{a_i} b_j \langle v_i | v'_j \rangle \langle w_i | w'_j \rangle$$

With this product and the previous properties in mind, it can easily be proven that the tensor product of Hilbert spaces is a Hilbert space.

Proposition 3. *Let H_1 be and H_2 be two Hilbert spaces with respective orthonormal bases $B_1 = \{|v_i\rangle\}_{i=1,\dots,n}$ and $B_2 = \{|w_j\rangle\}_{j=1,\dots,m}$. Then, $H_1 \otimes H_2$ is a Hilbert space with inner product given by*

$$(|v\rangle \otimes |w\rangle, |v'\rangle \otimes |w'\rangle) = \langle vw|v'w'\rangle = \langle v|v'\rangle\langle w|w'\rangle$$

For any $|v\rangle, |v'\rangle$ in V and $|w\rangle, |w'\rangle$ in W. Furthermore, $B_1 \otimes B_2 = \{|v_i w_j\rangle\}_{i,j}$ is an orthonormal basis of $H_1 \otimes H_2$.

Proof. By definition, $H_1 \otimes H_2$ is a complex vector space with $dim(H_1 \otimes H_2) = dim(H_1) \cdot dim(H_2)$. Furthermore, $B_1 \otimes B_2 = \{|v_i\rangle \otimes |w_j\rangle\}_{i,j} = \{|v_i w_j\rangle\}_{i,j}$ is a base of it. Let $|v_i w_j\rangle, |v_k w_l\rangle \in B_1 \otimes B_2$:

$$\langle v_i w_j | v_k w_l\rangle = \langle v_i|v_k\rangle\langle w_j|w_l\rangle = \delta_{ik}\delta_{jl}$$

which equals one if and only if $i = k$ and $j = l$, and zero otherwise. Thus, proving that $B_1 \otimes B_2$ is orthonormal.

Finally, for $H_1 \otimes H_2$ to be a Hilbert space the product defined above needs to be, in fact, an inner product. It suffies to prove that it is definite positive. Let $|v\rangle \in H_1 \otimes H_2$. Since $B_1 \otimes B_2$ is an orthonormal basis, there exist $\alpha_{ij} \in \mathbb{C}$ such that:

$$|v\rangle = \sum_{i,j} \alpha_{ij} |v_i w_j\rangle$$

By linearity:

$$\langle v|v\rangle = \sum_{i,j,k,l} \overline{\alpha_{ij}}\alpha_{kl}\langle v_i w_j | v_k w_l\rangle = \sum_{i,j,k,l} \overline{\alpha_{ij}}\alpha_{kl}\ \delta_{ik}\delta_{jl} = \sum_{i,j} |\alpha_{ij}|^2 \geq 0$$

where the equality holds if and only if $\alpha_{ij} = 0\ \forall i, j$. □

From the inner product, the tensor space $H_1 \otimes H_2$ naturally inherits the notions of adjoint, unitary, normality and Hermicity. We will denote a vector $|\varphi\rangle$ tensored with itsel n times, $|\varphi\rangle \otimes \overset{n}{\dots} \otimes |\varphi\rangle$, as $|\varphi\rangle^{\otimes n}$, and equivalently with Hilbert spaces: $H \otimes \overset{n}{\dots} \otimes H$, as $H^{\otimes n}$. For our quantum computing purposes, the following case holds particular relevance.

Corollary 3. *The tensor product of $\mathbb{C}^2$ with itself n times, $\mathbb{C}^2 \otimes \overset{n}{\dots} \otimes \mathbb{C}^2$, is isomorphic to $\mathbb{C}^{2^n}$.*

Proof. Let $H = \mathbb{C}^2 \otimes \overset{n}{\dots} \otimes \mathbb{C}^2$. Since $dim(H) = dim(\mathbb{C}^2)^n = 2^n$, H is a Hilbert space with dimension 2^n. Thus, by Hausdorff's Theorem [4], it is isomorphic to the canonical complex vector space of dimension 2^n, $\mathbb{C}^{2^n}$. □

In the same way that the inner product was naturally extended to the tensor space, we may extend operators ensuring linearity. Let $A : V \longrightarrow V'$ and $B : W \longrightarrow W'$ two operators between Hilbert spaces. Then, we define $A \otimes B : V \otimes W \longrightarrow V' \otimes W'$ using the following equation:

$$(A \otimes B)(|v\rangle \otimes |w\rangle) \equiv A|v\rangle \otimes B|w\rangle$$

We extend this definition to every element of $V \otimes W$ ensuring linearity:

$$(A \otimes B)\left(\sum_i a_i|v_i\rangle \otimes |w_i\rangle\right) \equiv \sum_i a_i A|v_i\rangle \otimes B|w_i\rangle$$

It can be shown that $A \otimes B$ is a well-defined linear operator. In fact, any linear operator mapping $V \otimes W$ to $V' \otimes W'$ may be expressed as a linear combination of tensor product of linear operators mapping V to V' and W to W':

$$C = \sum_i a_i A_i \otimes B_i$$

where by definition:

$$\left(\sum_i a_i A_i \otimes B_i\right)|v\rangle \otimes |w\rangle = \sum_i a_i A_i|v\rangle \otimes B_i|w\rangle$$

1.4.2 *Postulate 4 statement*

Reaching the core of this section, suppose we consider two (or more) distinct physical systems. In order to describe the state of the composite system we need to make use of tensor product.

> **Postulate 4.** The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\varphi_i\rangle$, then the joint state of the total system is $|\varphi_1\rangle \otimes \ldots \otimes |\varphi_n\rangle$.

Let us provide an intuitive notion of why is the tensor product used in order to describe composite systems. After all, we would expect that there exists a *somehow canonical way* os describing the product composition of systems, just like the cartesian product is used with vector spaces. Let us refer again to the so-called *superposition principle of quantum mechanics*. Let $|\varphi\rangle$ and $|\psi\rangle$ be two vector states of a system. Then, any superposition of the states, $\alpha|\varphi\rangle + \beta|\psi\rangle$, should also be another vector state, where $|\alpha|^2 + |\beta|^2 = 1$. Suppose we now have two systems X and Y, with two vector states

$|\varphi_x\rangle$ and $|\varphi_y\rangle$. Then, we could describe the state of the composite system XY as some state $|\varphi_x\rangle|\varphi_y\rangle$. Studying the superposition principle on composite systems leads us naturally to the notion of tensor product. It is important to note that this development is not thorough, since we are not taking the superposition principle as a fundamental component of our description of quantum mechanics. It does help up to build an intuitive notion on why this tensor product is used.

We now introduce the *Bell State* or *EPR pair:*

$$|+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

Although it may seem harmless at first glance, this state has been responsible for many surprises during the development of quantum physics [9]. Let us have a first look into it, although we will come back to it later on.

The first thing to notice about this state is that it can be expressed as a single tensor product of state vector: $|+\rangle \neq |a\rangle|b\rangle$. This means that this state is not describing two independent physical systems put together with their respective vector states independently, it describes a **relation** between both systems too. It means that those systems are somehow interfering with each other. So we may affect one by disrupting the other.

This phenomenon is called *entaglement*. A state is called *entangled* when it cannot be expressed as the product of states of its component systems. This concept rests at the heart of the disparity between classic physics and quantum physics and it is key in quantum computing. It was deeply studied first by Einstein, Podolsky, and Rosen (EPR) [10] and second by John Bell [9].

1.4.3 *Projective measurement*

In this section, we explain an important particular case of measurement. In fact, this method of projective measurement is equivalent to Postulate 2, when they are combined with the capacity to perform unitary operations. It has huge relevancy in quantum computing and it is explained here because of its relation with composite systems. Let us state the alternative postulate:

> **Projective Measurement.** A projective measurement is described by an *observable*, M, a Hermitian operator on the state space of the system being described. The observable has a spectral decomposition:
>
> $$M = \sum_m m P_m$$

where P_m is the projector onto the eigenspace with eigenvalue m. The possible outcomes of the measurement correspond to the eigenvalues, m, of the observable. Upon measuring the state $|\varphi\rangle$, the probability of the result m ocurring is:

$$p(m) = \langle\varphi|P_m|\varphi\rangle$$

and the state of the system after the measurement, givent that outcome m ocurred, is:

$$\frac{P_m|\varphi\rangle}{\sqrt{p(m)}}$$

Projective measurements can be seen as a particular case of Postulate 2. Suppose the measurement operators from such postulate, in addition to satisying the completeness relation $\sum_m M_m M_m^\dagger = I$, also fulfill that M_m are *orthogonal projectors*. That is, M_m is Hermitian and $M_m M_{m'} = \delta_{mm'} M_m \ \forall m, m'$. With these aditional restricions it holds that $M_m M_m^\dagger = M_m M_m = M_m$. By choosing $P_m = M_m$ we see that projective measurements is a particular case of Postulate 2.

Thanks to the way projective measurements are expressed, they have nice and handy properties. Let us compute the average value of a projective measurement:

$$\begin{aligned} E(M) &= \sum_m m p(m) \\ &= \sum_m m \langle\varphi|P_m|\varphi\rangle \\ &= \langle\varphi|\left(\sum_m m P_m\right)|\varphi\rangle \\ &= \langle\varphi|M|\varphi\rangle \end{aligned} \tag{1}$$

The average value of an operator is usually written as $\langle M\rangle \equiv \langle\varphi|M|\varphi\rangle$. The standar deviation associated to observations of M is then written as:

$$\begin{aligned} [\Delta(M)]^2 &= \left\langle (M - \langle M\rangle)^2 \right\rangle \\ &= \langle M^2\rangle - \langle M\rangle^2 \end{aligned} \tag{2}$$

This means that if we prepare our state $|\varphi\rangle$ multiple times and measure it, the measures will follow a normal distribution with expected value $\langle M\rangle$ and standard deviation $\Delta(M) = \sqrt{\langle M^2\rangle - \langle M\rangle^2}$. This formulation of standard deviation using observables

provides us with an elegant proof of perhaps the most famous quantum mechanics result: The *Heisenberg uncertainty principle*. Such proof can be found in annex [TODOref anexo con la demostracion, copy from Box 2.4 in Niel.].

There are two additional notations worth mentioning regarding projective measurement. Instead of providing an observable M, we will sometimes provide a complete set of orthogonal projectors. That is, a set P_m such that $\sum_i P_m = I$ and $P_m P_{m'} = \delta_{mm'} P_m$. The corresponding implicit observable used in this case is $M = \sum_m m P_m$. Finally, another widely used notation is to 'measure in a basis $|m\rangle$', where $|m\rangle$ form an orthonormal basis. This simply means that the chosen projectors are $P_m = |m\rangle\langle m|$ and the observable is $M = \sum_m m|m\rangle\langle m|$. This is the usual notation in quantum computing.

1.4.4 *Quantum Computing perspective: Multiple qubits*

Suppose we have a pair of qubits. In the classical case, two bits can be in four possible states: 00, 01, 10, and 11. Similarly, the two qubits computational basis states are $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. We arrive at this natural notation by knowing that each qubit has $\mathbb{C}^2$ as their associated state space and using the tensor product. Just like in the single qubit case, our two qubits system may be in a superposition of these four states:

$$|\varphi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Correspondingly, the measurement of this system will result in either 00, 01, 10 or 11 since we are using the observable $\sum_{x\in\{0,1\}^2} x|x\rangle\langle x|$, where $\{0,1\}^2$ are the strings of length two where each character is either 0 or 1. In fact, it will yield state x with probability $|\alpha_x|^2$, being α_x the coefficient associated with the state $|x\rangle$. The condition of the probabilities adding up to one is also called the *normalization condition* and can be expressed as $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$ for the two qubits case.

The fundamental differences with the single qubit case start on measurement. Of course, we can measure both qubits at the same time, but we could also measure only one of them. In order to achieve this we can use the projectors $P_0 = |00\rangle\langle 00| + |01\rangle\langle 01|$ and $P_1 = |10\rangle\langle 10| + |11\rangle\langle 11|$. Upon measuring, the system will collapse to $P_0|\varphi\rangle / \sqrt{p_0}$ with probability:

$$p_0 = \langle\varphi|P_0|\varphi\rangle = |\alpha_{00}|^2 + |\alpha_{01}|^2$$

Since these are the coefficients associated with the first qubit being 0, this is the probability of first qubit being 0. Furthermore, our system will collapse to:

$$|\varphi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

Note the normalization term $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$, which appears so the post-measurement state still satisfies the normalization condition. Naturally, after obtaining 0 in the first qubit we can still obtain either 0 or 1 in the second qubit, with probabilities

$$\frac{|\alpha_{00}|^2}{|\alpha_{00}|^2 + |\alpha_{01}|^2} \quad \text{and} \quad \frac{|\alpha_{01}|^2}{|\alpha_{00}|^2 + |\alpha_{01}|^2}$$

respectively, adding up to 1. Correspondingly, the first qubit being measured will yield 1 with probability $p_1 = |\alpha_{10}|^2 + |\alpha_{11}|^2$.

Additionally, the first qubit independently should satisfy the normalization condition. That is, its probabilities of being 0 and 1 upon measurement must add up to 1. But those are p_0 and p_1, which add up to one because of the normalization condition for $|\varphi\rangle$, as expected.

Let us come back to the previously introduced *Bell State*:

$$|+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

When measuring using the previously defined projectors P_0 and P_1 we obtain that the first qubit will collapse to either 0 or 1 with equal probability $p_0 = p_1 = 1/2$. Most importantly, in the first case the qubit will collapse to the state:

$$\frac{|00\rangle / \sqrt{2}}{\sqrt{p_0}} = |00\rangle$$

Meaning that the second qubit also collapsed to 0. Note that we only affected the first qubit and the second one was disturbed. These are the consequences of entanglement, key in the appearance of impressive phenomenoms and discussions such as the EPR Paradox [10], [9].

Let us finally consider the more general case. In an n-qubits system our computational (orthonormal) basis would consist of the sates $|x_1 x_2 \dots x_n\rangle$, where $x_i \in \{0, 1\}$. As we already know, in a single qubit system we have two amplitudes α_0 and α_1. We have four amplitudes for a 2-qubits system, eight for a 3-qubits system... And 2^n for an n-qubits system. This means that the number of amplitudes grows exponentially as we add qubits to the system. An immense increment compared to the classical case where the quantity of information that our system holds grows linearly with the numbers of bits. Of course, it is not that simple, since there are huge limitations on how we may access this information in the quantum realm such as how a qubit collapses upon measurement and the no-cloning theorem. However, we can already glimpse the power of quantum computing versus the classical one.

1.5 THE NO-CLONING THEOREM

A first exposure to the basis of quantum computing is fundamentally incomplete without the no-cloning theorem. It lets us understand one of the fundamental inconveniences of quantum computing: the impossibilty of cloning arbitrary states.

We proceed by formalizing the copying mechanism. Let H be a state space and $|\varphi\rangle \in H$ the state to be copied. Given another state $|\varphi'\rangle \in H$, we would like to copy $|\varphi\rangle$ into $|\varphi'\rangle$. Since we can prepare $|\varphi'\rangle$ at will before the copying occurs, let $|\varphi'\rangle = |x_0\rangle$ be a fixed state. Then, our copying operation will take both states and produce the following result:

$$(|\varphi\rangle \otimes |x_0\rangle) \xrightarrow{U} |\varphi\rangle \otimes |\varphi\rangle$$

Operations between n-qubits (n-qubits gates) must preserve their norm, thus they must be unitary $n \times n$ matrices. In our case, U is a $2dim(H) \times 2dim(H)$ unitary matrix. Let us now state and prove the theorem.

Theorem 3 (No-cloning Theorem)**.** *There is no unitary operator $U : H^{\otimes 2} \longrightarrow H^{\otimes 2}$ and state $|x_0\rangle \in H$ such that for any arbitrary state $|\varphi\rangle \in H$ it holds*

$$U|\varphi\rangle|x_0\rangle = |\varphi\rangle|\varphi\rangle$$

Proof. Suppose there exists such unitary operator U and such state $|x_0\rangle$. Let $|\varphi\rangle, |\psi\rangle \in H$. We may apply the copying operators to both of them:

$$U|\varphi\rangle|x_0\rangle = |\varphi\rangle|\varphi\rangle$$
$$U|\psi\rangle|x_0\rangle = |\psi\rangle|\psi\rangle$$

Taking the inner product of both equations and using that U is unitary results in:

$$\Big(U|\varphi\rangle|x_0\rangle, U|\psi\rangle|x_0\rangle\Big) = \Big(|\varphi\rangle|\varphi\rangle, |\psi\rangle|\psi\rangle\Big) \iff$$
$$\Big(|\varphi\rangle|x_0\rangle, |\psi\rangle|x_0\rangle\Big) = \langle\varphi|\psi\rangle\langle\varphi|\psi\rangle \iff$$
$$\langle\varphi|\psi\rangle\langle x_0|x_0\rangle = \langle\varphi|\psi\rangle^2$$

Since $|x_0\rangle$ is a normalized vector, $\langle x_0|x_0\rangle = 1$, and we obtain:

$$\langle\varphi|\psi\rangle = \langle\varphi|\psi\rangle^2$$

Which only holds if $\langle\varphi|\psi\rangle$ equals to either 0 or 1, which means $|\varphi\rangle$ and $|\psi\rangle$ are either equal or orthonormal. But those vectors were arbitrary, thus such a general cloning operator is impossible. □

2

THE D-WAVE QUANTUM COMPUTER

In this second chapter, we aim to understand the D-Wave Quantum systems. In order to achieve that, we start by explaining the basis of quantum annealing that the D-Wave systems used underneath, and how to transform problems into the adequate format for these systems.

2.1 QUANTUM ANNEALING

Simulated annealing (SA) is a general optimization technique that was first introduced in 1983 by Kirkpatrick *et al.* [11]. The main idea is to mimic how thermal fluctuations works to let the system escape from local minimums in the cost function. The *temperature* of the system dictates the probability with which the is allowed to jump to worse solutions (higher values of the cost function). The *annealing schedule* is the decrease rate of the temperature, controlled by the programmer. Under the appropriate one, the system would be able to escape from local minimums to reach the global one. If the temperature decreases too quickly, the system converges prematurely to a local minimum. If it decreases too slowly, the technique transforms into a random walker, reaching the global minimum at some point but being worthless time-wise.

Inspiration in thermal fluctuations is used in SA so the system may escape from local minima. Similarly, *quantum tunneling* is used in Quantum Annealing (QA) to escape from local minimums. This new technique was introduced in its present form by two similar proposals [12] [13].

Quantum tunneling is a quantum mechanical phenomenon where a wavefunction (i.e. a quantum state) may propagate through a potential barrier [14]. The probability of this event occurring depends on the height and width of the barrier (see figure 2)

The main advantage of quantum tunneling compared to classical thermal fluctuations is that the probability of the system shifting depends not only on the height of the potential barrier but also on its width. Thus, the application of QA may potentially outperform SA in problems where the energy (cost) landscape consists of very high and thin barriers surrounding local minimums.

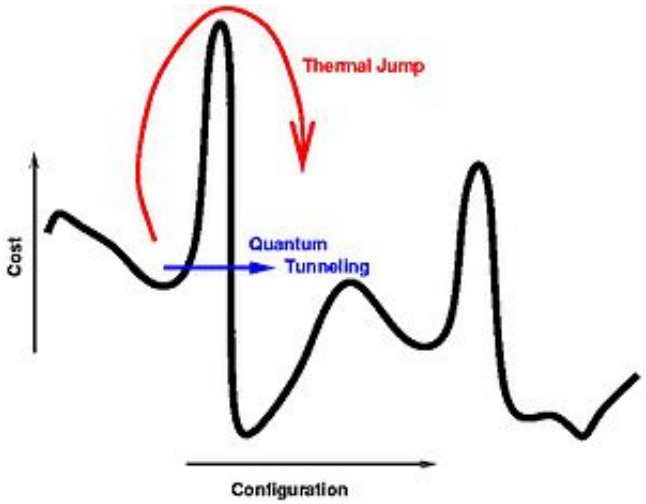


Figure 2: Thermal jumps vs quantum tunneling.

Particularly, as studied in [15], let Δ be the height of a barrier and ω its width. The probability of a thermal transition occurring is dictated by $exp\{-\frac{\Delta}{k_B T}\}$ where T and the temperature of the system and k_B is the Boltzmann constant. On the other hand, the probability of a quantum tunneling transition occurring through the same barrier (assuming isolation) is $exp\{-\frac{\sqrt{\Delta}\omega}{\Gamma}\}$, where Γ is an additional constant called the tunneling field [16]. Thus, the probability is much higher for the quantum tunneling effect on high and thin barriers.

We may also observe looking at the previous expressions how the tunneling field and the temperature play similar roles in the different methods. Thus, the annealing schedule of the QA system is controlled using the tunneling field just like the temperature is used in SA.

There are many examples in the literature of QA simulations using classical computers, both theoretical [17] and numerical -most of them using Quantum Monte Carlo (QMC) methods [18] [19]-. This evidence suggests the outperformance of SA by simulated QA under certain conditions. It is reasonable to wonder what happens if instead of simulating QA we could somehow encode our problems in a real physical system that tends to its lower energy state (ground state), thus using quantum tunneling naturally. This is precisely what the D-Wave system does.

Let us further deepen into the technicalities of quantum annealing in order to understand how it is implemented in the D-Wave system.

2.2 ADIABATIC EVOLUTION

The alternative form of the third postulate of quantum mechanics (section 1.3) stated that the evolution of a quantum system is described by the Schrodinger equation:

$$i\hbar \frac{d|\varphi\rangle}{dt} = H|\varphi\rangle$$

By better understanding the Hamiltonian H we may control this evolution. Suppose that the evolution of a given quantum system is described by a Hamiltonian $\hat{H}(t)$. Suppose that at some initial time t_0 our quantum system is in an eigenstate $|\varphi(t_0)\rangle$ of $\hat{H}(t_0)$. Since the evolution is continuous, at some other time $t_1 > t_0$ we could expect our system to be in the corresponding eigenstate $|\varphi(t_1)\rangle$ of $\hat{H}(t_1)$. This fact critically depends on the time $\tau = t_1 - t_0$ during which the modification takes place, as stated by the adiabatic theorem, first proposed by Max Born and Vladimir Fock (1928) [20].

Theorem 4 (Adiabatic Theorem)**.** *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.*

Thus, we may define diabatic and adiabatic processes depending on how they adapt to the system changes [21].

Definition 13. A *diabatic process* is a process where rapidly changing conditions prevent the system from adapting its configuration during the process. Typically there is no eigenstate of the final Hamiltonian with the same functional form as the initial state. The system ends in a linear combination of the states.

Definition 14. An *adiabatic process* is a process where gradually changing conditions allow the system to adapt its configuration. If the system starts in an eigenstate of the initial Hamiltonian, it will end in the corresponding eigenstate of the final Hamiltonian.

The 'gap condition' that appears in theorem 4 refers to an additional requirement: the spectrum of $\hat{H}(t)$ is *nondegenerate*, meaning there are no two equal eigenvalues at fixed time t. This condition is also called the *no-crossing* condition and allows us to sort the eigenstates using the eigenvalues without any ambiguity.

As a classical analogy for adiabatic evolution, consider a simple pendulum. If the pendulum's support point is moved, the oscillation of the pendulum may change. In fact, violent changes on the support point will dramatically affect the pendulum's movement, changing it completely. However, if the support point is moved slowly enough, the pendulum will remain unchanged. These are examples of diabatic and adiabatic processes respectively.

2.2.1 *Avoided crossing*

Let us consider an important physical example known as the *avoided crossing* [22]. Consider a two-state quantum system, i.e. a qubit. Suppose its Hamiltonian to be:

$$H = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}$$

Whose eigenvalues are E_1 and E_2, and eigenvectors,

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

which will be used as our base. Thus, a state vector describing the system may be written as a superposition of both states:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Supposing adiabatic evolution, if the system is prepared in either one of the eigenstates it will remain as such as long as the gap condition is sufficed: $E_1 \neq E_2$. However, if $E_1 = E_2$, any superposition of states will be an eigenstate, thus remaining unchanged. Hence, independently of E_1 and E_2, any system prepared in an eigenstate will remain as such, supposed adiabatic evolution.

Let us consider a perturbation P into our original system. For simplicity, we only consider perturbations with degenerated diagonal. Our new Hamiltonian will be the following:

$$H' = H + P = \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix} + \begin{pmatrix} 0 & \omega \\ \overline{\omega} & 0 \end{pmatrix} = \begin{pmatrix} E_1 & \omega \\ \overline{\omega} & E_2 \end{pmatrix}$$

where $\omega \in \mathbb{C}$. Note that by setting ω, the other value in the antidiagonal is fixed since H' must be Hermitian. By using the characteristic equation we may compute the eigenvalues of H':

$$E_+ = \frac{1}{2}(E_1 + E_2) + \frac{1}{2}\sqrt{(E_1 - E_2)^2 + 4|\omega|^2}$$
$$E_- = \frac{1}{2}(E_1 + E_2) - \frac{1}{2}\sqrt{(E_1 - E_2)^2 + 4|\omega|^2}$$

We may plot E_+ and E_- along the vertical axis while varying $(E_1 - E_2)$ in the horizontal axis, and find two branches of a hyperbola. The curves asymptotically approach the original unperturbed states E_1 and E_2. Analyzing the curves and the previous equations it becoms evident that the new energy staes are no longer equal, even if the original states were degenerate ($E_1 = E_2$). However, if the pertubation is removed, $\omega = 0$, if may find that where the states are degenerate and $(E_1 - E_2) = 0$, the new

eigen become equal, $E_+ = E_-$ and the levels cross. Thus, by adding a perturbation efffect, these level crossing are completely avoided.

In the table of figures 1, we find a series of plots describing this behavior for different values of ω. The eigenvalues have been set to $E_1 = -E_2$ (thus $E_1 + E_2 = 0$) for simplicity. We can see how as the perturbation ω becomes smaller, the hyperbola branches adjust better to the original states and the gap becomes thinner and thinner. Finally, when the perturbation is removed, the levels cross. The reader may find an interactive representation using GeoGebra in [23].

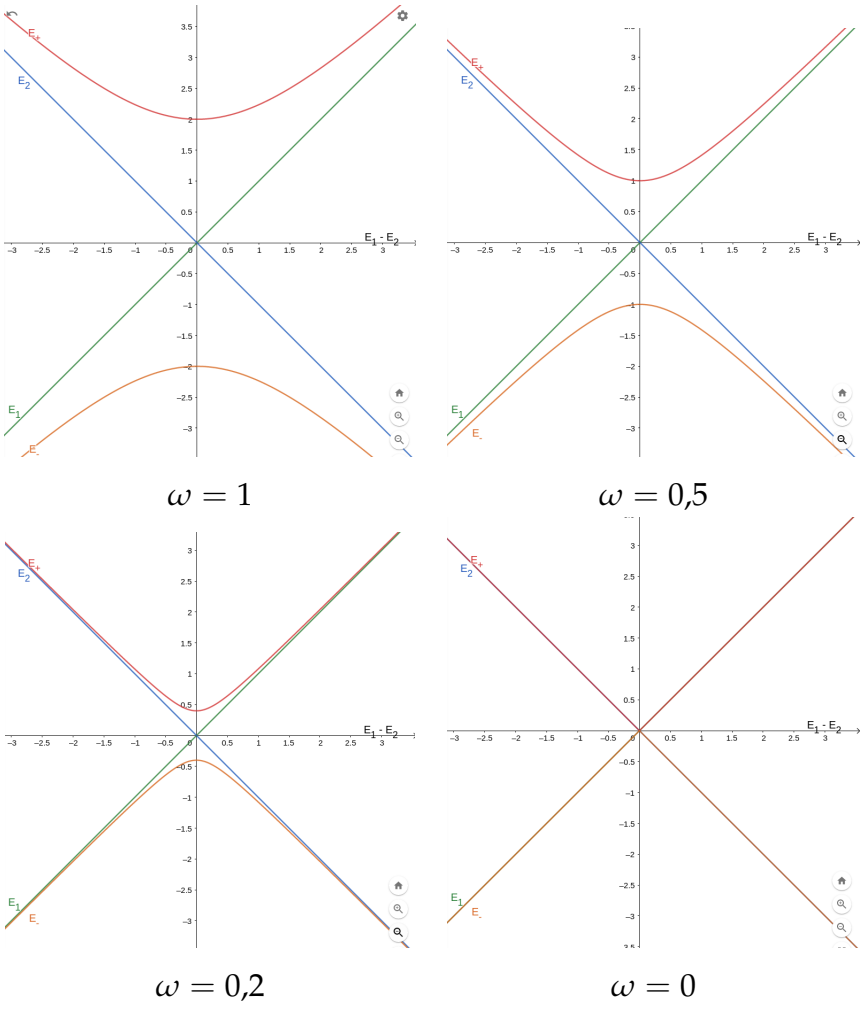


Table 1: Avoided crossing in two-state system [23]. By increasing the perturbation ω, the energy level crossing is avoided.

In the case of a single qubit, the two original states $|0\rangle$ and $|1\rangle$ are never degenerate by our definition, so the crossing does not occur with or without a perturbation.

The implications of energy level crossing in quantum annealing will be better explained in section 2.4.1. For now, it suffices to now that avoided crossing in energy levels

is a strongly desired hypothesis that in practice is difficult to fulfill. It will be partially responsible for suboptimality in the use of quantum annealing.

2.2.2 *Solving problems using adiabatic evolution*

In order to use adiabatic evolution to solve optimization problems we will configure our own Hamiltonian for the system. In particular, the Hamiltonian will be a time-dependent linear combination of two constant Hamiltonians:

$$H(t) = f(t) \cdot H_{initial} + g(t) \cdot H_{final} \quad \forall t \in [t_{initial}, t_{final}]$$

where $f, g : [t_{initial}, t_{final}] \longrightarrow \mathbb{R}_0^+$. The initial Hamiltonian, $H_{initial}$, will be an easy configuration so we may prepare our system to start in the ground state of $H_{initial}$. The final Hamiltonian, H_{final}, will encapsulate the cost function of our problem such that the minimum value is reached in the ground state.

Functions f and g fulfill that $g(t_{initial}) = f(t_{final}) = 0$. When times elapses and supposing adiabatic evolution, the system will stay on the ground state of H and by $t = t_{final}$, the system will be in the H_{final}'s ground state, codifying the minimum of our cost function.

However, we cannot always grant the necessary conditions for perfect adiabatic evolution. In some cases, the gap condition does not suffice as the system will jump with a certain probability to other eigenstates, providing suboptimal solutions which are still useful. This depends on the eigenvalues of the final Hamiltonian, which at the same time entirely depends on the problem being studied.

The only question remaining is how to encode our cost function in a Hamiltonian.

2.3 QUBO AND ISING MODELS

The QUBO and Ising models can be easily represented with a Hamiltonian. In practice, non-optimization problems are transformed into optimization problems and then codified as either QUBO or Ising to be solved using quantum annealing. Let us explore these types of problems and some examples of these transformations.

Let $B = \{0, 1\}$ and $f_Q : \mathbb{B}^n \longrightarrow \mathbb{R}$ be a quadratic polynomial over binary variables:

$$f_Q(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij} x_i x_j$$

where $x_i \in \mathbb{B}$ for $i \in \{1, \cdots, n\}$ and the coefficients $q_{ij} \in \mathbb{R}$ for $1 \leq j \leq i \leq n$. A **quadratic unconstrained binary optimization (QUBO) problem** consists of finding a binary vector x' such that x' is a minimum of f_Q:

$$x' = \arg\min_{x \in \mathbb{B}^n} f_Q(x)$$

QUBO problems can be formulated in a more compact matrix form:

$$f_Q(x) = x^T Q x$$

where Q is a $n \times n$ symmetric matrix containing q_{ii} and its diagonal and $q_{ij}/2$ in position (i, j) where $i \neq j$. Let us explore some simple properties of QUBO problems:

- Multiplying the coefficients q_{ij} by a constant $a \in \mathbb{R}^+$ scales the output by exactly a. Thus, x' remains the minimum:

$$f_{aQ}(x) = \sum_{i<j} (aq_{ij}) x_i x_j = a \sum_{i<j} q_{ij} x_i x_j = a f_Q(x)$$

- Flipping the sign of the coefficients flips the sign of $f_Q(x)$. Thus, x' is the maximum of $f_{-Q}(x)$

$$f_{-Q}(x) = \sum_{i<j} (-q_{ij}) x_i x_j = -\sum_{i<j} q_{ij} x_i x_j = -f_Q(x)$$

- If all coefficients are positive the minimum is $x = (0, \ldots, 0)$. Analogously, if all coefficients are negative, the minimum is $x = (1, \ldots, 1)$.

When the dependendy on Q is obvious, it may be left implicit by writting our cost function simply as f.

The **Ising models**, on the other hand, find their inspiration in physics. They are named after the physicist Ernst Ising who solved the one-dimensional model in his 1924 thesis [24]. They are stated using a Hamiltonian function $H : \{-1, 1\}^n \longrightarrow \mathbb{R}$:

$$H(\sigma) = -\sum_{\langle i\ j \rangle} J_{ij} \sigma_i \sigma_j - \mu \sum_j h_j \sigma_j$$

with parameters $h_j, J_{ij}, \mu \in \mathbb{R}$. The *spin variables* σ_i are in $\{-1, 1\}$ instead of $\mathbb{B}$. Moreover, the spin variables are arranged in a graph, typically a *lattice*, where a local structure repeats periodically. The only pair of variables $\langle i\ j \rangle$ which are neighbor nodes in the graph may have non-zero coefficients J_{ij}. Let us see the connection between the QUBO and Ising models by using the identity $\sigma \mapsto 2x - 1$:

$$
\begin{aligned}
f(x) &= \sum_{\langle i\ j\rangle} -J_{ij}(2x_i - 1)(2x_j - 1) - \sum_j \mu h_j(2x_j - 1) \\
&= \sum_{\langle i\ j\rangle} -4J_{ij}x_i x_j + 2J_{ij}x_i + 2J_{ij}x_j - J_{ij} + \sum_j 2\mu h_j x_j - \mu h_j \qquad \text{using } x_j = x_j x_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{i} q_{ij}x_i x_j + C
\end{aligned}
$$

where

$$
q_{ij} = \begin{cases} -4J_{ij} & \text{if } i \neq j \\ \sum_{\langle k\ i\rangle} 2J_{ki} + \sum_{\langle i\ l\rangle} 2J_{il} + 2\mu h_i & \text{if } i = j \end{cases}
$$

$$
C = -\sum_{\langle i\ j\rangle} J_{ij} - \mu \sum h_j
$$

Since adding a constant does not change the minimum x', it can be omitted during optimization and only be used for transforming one type of problem into another.

In practice, a given problem would be translated into either QUBO or Ising. It is useful to learn both types of problems since they both are popular in the literature. Let us deepen in the translation of different NP problems. The resolution of QUBO and Ising models will not be studied, but only the translations into these models. In particular, these examples of QUBO transformations are from [25] and [26] unless stated otherwise. For an extensive literature review on QUBO and Ising translations of NP problems, refer to [27].

2.3.1 *The Max Cut problem*

One of the first famous problems that was encoded as a QUBO was the max-cut problem due to its natural translation. It reads as follows: Given an undirected graph $G(V, E)$ with vertex set V and edge set E, seek a partition of V such that the number of edges between both sets is as large as possible.

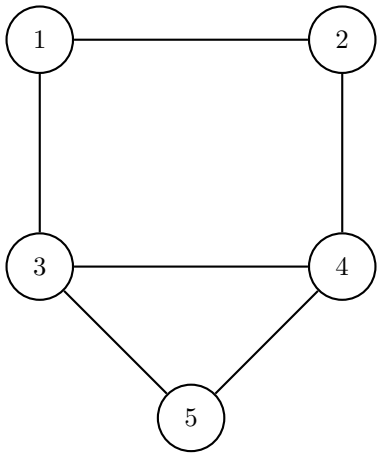
The max-cut problem can be modeled by setting variable $x_j = 1$ if vertex j is in one set and $x_j = 0$ if it is in the other one, for every vertex j in V. The quantity $(x_i + x_j - 2x_i x_j)$ marks whether the edge (i, j) is in the cut. That is, $(x_i + x_j - 2x_i x_j) = 1$ if the vertices x_i and x_j are in different sets, and 0 otherwise.

Thus, maximazing the number of edges in the cut is encoded as a QUBO as:

$$\text{Maximize } f(x) = \sum_{(i,j)\in E} \left(x_i + x_j - 2x_i x_j\right)$$

NUMERICAL EXAMPLE

Consider the following graph:



We would have five variables: $x_1, \cdots, x_5$. The cost associated to edge $(1,2)$ is $x_1 + x_2 - 2x_1x_2$. By repeating this process for every edge in the graph we obtain the cost function:

$$\begin{aligned}\text{Maximize } f(x) = (x_1 + x_2 - 2x_1x_2) + (x_1 + x_3 - 2x_1x_3) + (x_2 + x_4 - 2x_2x_4)\\ +(x_3 + x_4 - 2x_3x_4) + (x_3 + x_5 - 2x_3x_5) + (x_4 + x_5 - 2x_4x_5)\end{aligned}$$

or equivalently:

$$\begin{aligned}\text{Maximize } f(x) = 2x_1 + 2x_2 + 3x_3 + 3x_4 + 2x_5\\ -2x_1x_2 - 2x_1x_3 - 2x_2x_4 - 2x_3x_4 - 2x_3x_5 - 2x_4x_5\end{aligned}$$

We can transform the problem using the following symmetric matrix:

$$Q = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

Yielding the following compact form:

$$\text{Maximize } f_Q(x) = x^T Q x \tag{3}$$

Keep in mind that quadratic elements in the cost function are divided by two when added to the Q matrix, since after multiplying and adding in equation 3 they appear twice.

By solving this model using quantum annealing we obtait the minimum value of $f_Q(x)$: $x = (0, 1, 1, 0, 0)$. Thus, vertices 2 and 3 are in one set and vertices 1, 4 and 5 are in the other one.

2.3.2 *General translation method*

The previous problem was naturally transformed into QUBO due to its particular constraints. However, can any quadratic optimization problem be encoded as a QUBO?

The answer to this question is positive. Furthermore, any quadratic optimization problem with linear constraints may be transformed into a QUBO. Let us first study a simple case. Consider the following problem:

$$\begin{gathered}\text{Minimize } f(x_1, x_2) \\ \text{subject to the constraint} \\ x_1 + x_2 \leq 1\end{gathered}$$

where $x_1, x_2 \in \mathbb{B}$. We may transform our linear constraint into a penalty $P(x_1 x_2)$ for some positive scalar P. This term equals 0 if and only if our constraint is satisfied by the variables. Thus, by adding the penalty to our cost function, for P large enough, the problem

$$\text{Minimize } f(x_1, x_2) + P x_1 x_2$$

will minimize $f(x_1, x_2)$ and, at the same time, satisfy the constraint. Namely, the constraint translates into adding a non-negative value to the cost function. Some other useful translations can be found in table 2. In general, if the cost function is being minimized we may substitute constraints by non-negative terms that only equal zero if they are satisfied, multiplied by a large enough positive value P. Thus, we may remove the associated constraints by simply adding these terms to the cost function.

Specifically, let us study the more general case:

Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

Table 2: Constraints and penalties relations

$$\text{Minimize } f(x) = x^T C x$$
$$\text{subject to the constraints}$$
$$Ax = b$$

where $x \in \mathbb{B}^n$, k is the number of constraints, $b \in \mathbb{R}^k$, C is a $n \times n$ real matrix and A is a $k \times n$ real matrix. This accomodates to both linear and quadratic cost function since elements in the diagonal are $c_{ii}x_i^2 = c_{ii}x_i$. For a positive scalar P, the quadratic penalty $P(Ax - b)^T(Ax - b)$ is added to the cost function to obtain:

$$\begin{aligned} f'(x) &= x^T C x + P(Ax - b)^T(Ax - b) \\ &= x^T C x + P(x^T A^T A x - \cancel{x^T A^T b} - \cancel{b^T A x} + b^T b) \\ &= x^T C x + x^T D x + c \\ &= x^T Q x + c \end{aligned}$$

where $D = P(A^T A)$, $c = P(b^T b)$ and $Q = C + D$. Since the constant c does not affect where the minimum is reached, f and f' share this minimum. Therefore, we may simply solve:

$$\text{Minimize } f(x) = x^T Q x$$

which is a QUBO problem.

In this development, we only considered equality constraints. There is also a general method to transform inequality constraints into equality ones by introducing *slack variables* [28]. In practice, the general case will not be necessary since the constraints are usually quite simple.

Finally, we may also consider satisfiability problems. That is, given a set of constraints for the variables, find a configuration of them such that all the constraints are satisfied.

If the constraints are linear, these problems may be transformed into QUBO by setting the cost function as constant and following the same method. Suppose the constraints over x can be written as $Ax = b$. The equivalent optimization problem is written as:

$$\text{Minimize } f(x) = 0$$
$$\text{subject to the constraints}$$
$$Ax = b$$

Any solution satisfying the constraint will yield 0 on the penalties, thus minimizing the transformed cost function $f'(x) = x^T Q x$.

Let us explore some more complicated examples to expand on these ideas.

2.3.3 *Satisfiability problems: Max 2-SAT*

This example is based on [25] and [19].

In the previous section, we mentioned satisfiability problems, also known simply as SAT. In this section, we study these problems from its traditional logic perspective.

A *clause* is statement depending on a finite set of binary variables , $C = C(x_1, \ldots, x_k)$, that be either True or False. A *literal* is a variable or its negation: x and $\neg x$ are literals. Let $C_1, \ldots, C_n$ be a set of *Clauses* depending on variables $x_1, \ldots, x_m$. The satisbiality problem associated to this problem is to find a configuration for the variables $x_1, \ldots, x_m$ such that every clause is true:

$$C_1 \wedge \cdots \wedge C_n$$

SAT was the first problem proven to be NP-complete. This result is known as the Cook-Levin theorem, named after Stephen Cook and Leonid Levin [29]. Its relevance in computer science is astronomical. There have been many variations of SAT problems, let us explore one of them, the Max 2-SAT variation, and transform it into a QUBO model.

Max 2-SAT problem is an optimization problem. Its main restriction is that each clause may depend on at most two literals and it is true if and only if at least one literal is true. The problem consists of finding a configuration of variables such that the number of true clauses is maximized. Each clause $C(x_C, y_C)$ is associated with an energy function depending on the variables (x_C, y_C):

$$E_C(x_C, y_C) = \begin{cases} 0, & \text{if } (x_C, y_C) \text{ satisfies clause } C \\ 1, & \text{if } (x_C, y_C) \text{ violates clause } C \end{cases}$$

The energy of the whole system, which will work as our cost function, is defined as the sum of all the energies:

$$E = \sum_C E_C$$

Clearly $E \geq 0$ and $E = 0$ if and only if every clause is true. Furthermore, the minimum energy of the system E_0 will be reached where the maximum number of clauses are satisfied: exactly every clause but E_0 will be true. Note that this perspective does not depend on the Max 2-SAT problem. In fact, the same process is used to transform other satisfiability problems.

The remaining question is how to transform our clauses into energy functions. For the Max 2-SAT problem this is quite straightforward. Each clause may depend on at most two literals. Thus, there are either zero, one, or two negations in each clause. We may study this cases separately, as seen in table 3.

Clause	Traditional constraing	Energy function
$x \vee y$	$x + y \geq 1$	$(1 - x - y + xy)$
$\neg x \vee y$	$\neg x + y \geq 1$	$(x - xy)$
$\neg x \vee \neg y$	$\neg x + \neg y \geq 1$	(xy)

Table 3: Clause and energy functions relations

Since every energy function is quadratic, the total energy will also be quadratic. This is the cost function used in our QUBO model:

$$\text{Minimize } E(x) = \sum_C E_C$$

Note that, interestingly enough, our QUBO model does not depend on the number of clauses but only on the number of variables. This means that a problem with 30,000 clauses and 200 variables will only use 200 variables in the associated QUBO model.

NUMERICAL EXAMPLE

Given the following set of clauses found in table 4, find a solution to the Max 2-SAT problem associated with them.

A quick look at the first four clauses tells us that we cannot possibly find a configuration of variables that satisfy every clause. In fact, at most three of those four will be true. Adding the energies associated with every clause gives us the energy of the system, and therefore our QUBO model:

Clause #	Clause	Energy function
1	$x_1 \vee x_2$	$(1 - x_1 - x_2 + x_1x_2)$
2	$x_1 \vee \neg x_2$	$(x_2 - x_1x_2)$
3	$\neg x_1 \vee x_2$	$(x_1 - x_1x_2)$
4	$\neg x_1 \vee \neg x_2$	(x_1x_2)
5	$\neg x_1 \vee x_3$	$(x_1 - x_1x_3)$
6	$\neg x_1 \vee \neg x_3$	(x_1x_3)
7	$x_2 \vee \neg x_3$	$(x_3 - x_2x_3)$
8	$\neg x_2 \vee x_3$	$(x_2 - x_2x_3)$
9	$\neg x_2 \vee \neg x_3$	(x_2x_3)
10	$x_2 \vee x_4$	$(1 - x_2 - x_4 + x_2x_4)$
11	$x_3 \vee x_4$	$(1 - x_3 - x_4 + x_3x_4)$
12	$\neg x_3 \vee \neg x_4$	(x_3x_4)

Table 4: Clause and energy functions relations

$$\text{Minimize } E(x) = 3 + x_1 - 2x_4 - x_2x_3 + x_2x_4 + 2x_3x_4$$

or equivalently:

$$\text{Minimize } E(x) = 3 + x^T Q x$$

where

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1/2 & 1/2 \\ 0 & -1/2 & 0 & 1 \\ 0 & 1/2 & 1 & -2 \end{pmatrix}$$

Solving this QUBO gives $x_1 = x_2 = x_3 = 0$ and $x_4 = 1$ with energy $E(0,0,0,1) = 1$, meaning that all clauses but one are satisfied.

As a final remark, this method of solving Max 2-SAT problems has proven to be computationally useful up to hundreds of variables and thousands of clauses [30].

2.3.4 *Graph Coloring*

Consider the following problem: Given a graph G and K-colors, assign a single color to each vertex in V such that no two adjacent vertices have the same color. Let x_{ij} be

a binary variable that is set to 1 if and only if the i-th node is assigned the color j-th. We may develop the following two constraints. First, since all nodes must be colored exactly once:

$$\sum_{j=1}^{K} x_{ij} = 1 \quad \forall i \in \{1, \cdots, n\}$$

where n is the number of nodes. Moreover, the condition that two adjacent nodes cannot have the same color is translated into:

$$x_{ic} + x_{jc} \leq 1 \quad \forall c \in \{1, \cdots, K\}$$

for every pair of adjancent nodes (i, j) in G.

Thus, by finding a vector configuration for the variables x_{ij} that satisfy this constraint, we find a solution to the graph coloring problem. This is what we previously introduced as a satisfiability problem. Therefore, it may be written as:

$$\text{Minimize } f(x) = 0$$

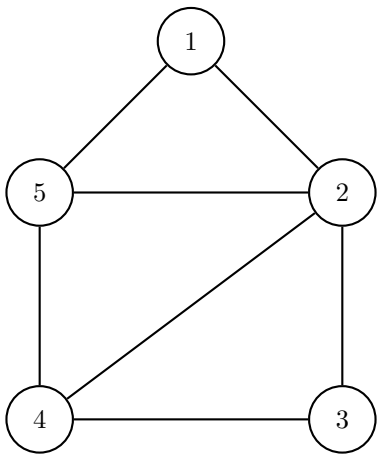
$$\text{subject to the constraints}$$

$$\sum_{j=1}^{K} x_{ij} = 1 \quad \forall i \in \{1, \cdots, n\} \quad \text{and}$$

$$x_{ic} + x_{jc} \leq 1 \quad \forall c \in \{1, \cdots, K\} \quad \text{for all adjacent nodes i and j}$$

NUMERICAL EXAMPLE

Given the following graph, find a coloring using three colors.



For this particular graph there are 15 variables: x_{ij} for all $i \in \{1, \cdots, 5\}$ and for all $j \in \{1, 2, 3\}$. The associated constraints are:

$$x_{i1} + x_{i2} + x_{i3} = 1 \quad \forall i \in \{1, \cdots, 5\}$$
$$x_{ic} + x_{jc} \leq 1 \quad \forall c \in \{1, 2, 3\} \quad \text{for all adjacent nodes i and j}$$

a total of 26: 5 for assigning one and only one color to each node, and 3 additional ones per edge in the graph (7) for no adjacent equally colored nodes. First, let us rename our variables from 1 to 15 as follows:

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, \ldots, x_{52}, x_{53}) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots, x_{15}, x_{16})$$

The first set of constraints may be substitued by $P(x_{i1} + x_{i2} + x_{i3} - 1)^2$, which only equals 0 if the constraint is satisfied. For instance, for the first node $P(x_1 + x_2 + x_3 - 1)^2$ is tranformed to $P(-x_1 - x_2 - x_3 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 + 1)$. By chosing a value for $P = 4$ and dropping the constant we can insert these penalties into an under developed Q matrix:

$$Q = \begin{pmatrix}
-4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -4 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & -4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -4 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & -4
\end{pmatrix}$$

Secondly, in order to transform $x + y \leq 1$ we can refer to table 2 and use $P(xy)$. We add these penalties to the previous matrix in the proper places:

$$Q = \begin{pmatrix}
-4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
4 & -4 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\
0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\
0 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 \\
2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 \\
0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & -4 & 4 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 4 & -4
\end{pmatrix}$$

Since the matrix above incorporates all the constraints, finding a graph coloring is equivalent to finding a minimum of:

$$\text{Minimize } f(x) = x^T Q x$$

Solving this model using quantum annealing yields a possible solution: $x_2 = x_4 = x_9 = x_{11} = x_{15} = 1$, and 0 for the rest of the variables. Meaning, nodes 1 and 4 are assigned color number 2, node 2 is assigned color number 1, and nodes 3 and 5 are assigned color number 3.

As a final remark, this method of graph coloring has proven to be computationally useful for graphs with up to 450 nodes and 9757 edges [30].

2.3.5 *The Travelling Salesman problem*

This example is mainly based on [25].

In order to formulate our next problem, a couple of extra definitions are needed. Given a graph, a *Hamiltonian path* is a path -a sequence of connected nodes- such that it only passes through each node exactly once. A *cycle*, also called *tour*, is a path that starts and ends in the same node. A *Hamiltonian cycle* is a cycle that passes through each node a single time, except for the first/last node which appears twice

in the cycle. Additionally, a weighted graph $G = (V, E, W)$ is a graph with a series of weights $w_e \in \mathbb{R}$, each associated to an edge $e \in E$.

The Travelling Salesman problem is the following: Given a directed weighted graph, find a Hamiltonian cycle with minimum total weight. The problem is sometimes stated with a Hamiltonian path instead of a Hamiltonian cycle, but as it will be seen in section [TODOref], for our purposes the cycle-statement will be of better use.

We may suppose without losing generality that the graph is complete. That is, there is always an edge from every two nodes in every direction. If the given graph is not complete, we may add the remaining edges with a high enough weight so that they are never used.

Since solutions to this problem are a series of $n \equiv |V|$ nodes -the number of nodes in the graph-, we will encode n-nodes cycles as $(v_{k_0}, \ldots, v_{k_{n-1}})$, where nodes v_{k_0} and $v_{k_{n-1}}$ are also conected. Thus, our problem may originally may stated as:

$$\text{Minimize} \sum_{i=0}^{n-1} w_{k_i,k_{i+1}}$$

$$\text{subject to the constraint that } (v_{k_0}, \cdots, v_{k_{n-1}}) \text{ e}$$

where $w_{k_i,k_{i+1}}$ is the weight associated to the edge connecting nodes v_{k_i} and $v_{k_{i+1}}$; and where the addition is performed under modulus n: $k_n \equiv k_0$. The binary variables $x_{i,p}$ will be used, where $i, p \in \{0, \ldots, n-1\}$. $x_{i,p}$ will equal 1 if and only if node i is in position p in our cycle. Hence, we will have n^2 variables under the following constraints:

1. Every node must be assigned. Thus, node assigment will be rewarded with a non-positive weight (favorable bias) called **self-bias** with an associated non-positive penalty:

   $$ax_{i,p} \quad \forall i, p \in \{0, \ldots, n-1\}$$

   where $a \leq 0$. In practice, this will not be necessary for every problem. We may remove the self-bias by setting $a = 0$. Adding these penalties for every node and position yields:

   $$a \sum_{i=0}^{n-1} \sum_{p=0}^{n-1} x_{i,p} \tag{4}$$

2. Every node must be assigned to only one position. This constraint will be called **repetition**:

$$\sum_{p=0}^{n-1} x_{i,p} = 1 \quad \forall p \in \{0, \dots, n-1\}$$

Resulting in the following penalty:

$$b\Big(\sum_{p=0}^{n-1} x_{i,p} - 1\Big)^2 \quad \forall p \in \{0, \dots, n-1\}$$

where b is a positive parameter. Adding these penalties for eveyr node we obtain:

$$b \sum_{i=0}^{n-1} \Big(\sum_{p=0}^{n-1} x_{i,p} - 1\Big)^2 \tag{5}$$

3. Every position must be assigned only one node. This constraint will be called **colocation**:

$$\sum_{i=0}^{n-1} x_{i,p} = 1 \quad \forall i \in \{0, \dots, n-1\}$$

Resulting in the following total penalty:

$$c \sum_{p=0}^{n-1} \Big(\sum_{i=0}^{n-1} x_{i,p} - 1\Big)^2 \tag{6}$$

where c is a positive parameter.

Finally, we may re-write the cost function in terms of the binary variables. Weight w_{ij} must be added in the cost function if and only if nodes i and j follow each other in the path. That is, if and only if $x_{i,p}x_{j,p+1} = 1$ for any position p. In fact, it may be needed to add it multiple times if the previous equality holds for multiple positions. Thus, the penalty associated to weight $w_{i,j}$ is:

$$w_{i,j} \sum_{p=0}^{n-1} x_{i,p}x_{j,p+1}$$

Adding up the penalties associated with every weight yields the cost function:

$$\text{Minimize} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} w_{i,j} \sum_{p=0}^{n-1} x_{i,p}x_{j,p+1}$$

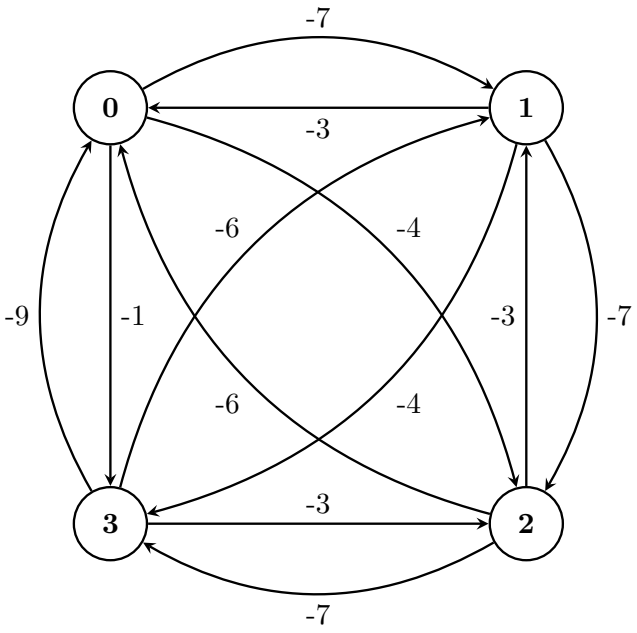
The final QUBO model is obtained by adding the penalties 4, 5 and 6 to the cost function:

$$\begin{aligned}\text{Minimize } & \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_{i,j}\sum_{p=0}^{n-1} x_{i,p}x_{j,p+1} \\ & + a\sum_{i=0}^{n-1}\sum_{p=0}^{n-1} x_{i,p} && \text{(self-bias)} \\ & + b\sum_{i=0}^{n-1}\Big(\sum_{p=0}^{n-1} x_{i,p}-1\Big)^2 && \text{(repetition)} \\ & + c\sum_{p=0}^{n-1}\Big(\sum_{i=0}^{n-1} x_{i,p}-1\Big)^2 && \text{(colocation)}\end{aligned} \tag{7}$$

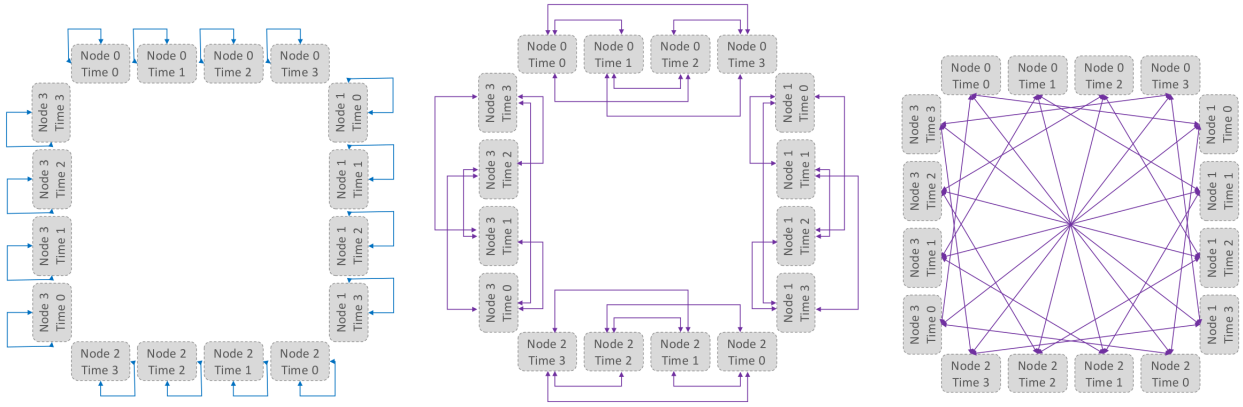
Since this is a quadratic unconstraint function, this is already a QUBO. The conversion to a matrix is straightforward. Let us see this in an example.

NUMERICAL EXAMPLE

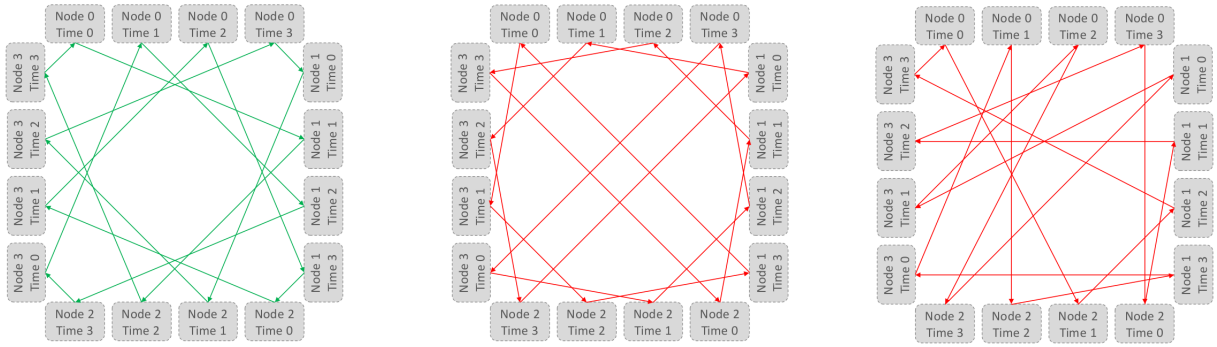
Given the following graph:



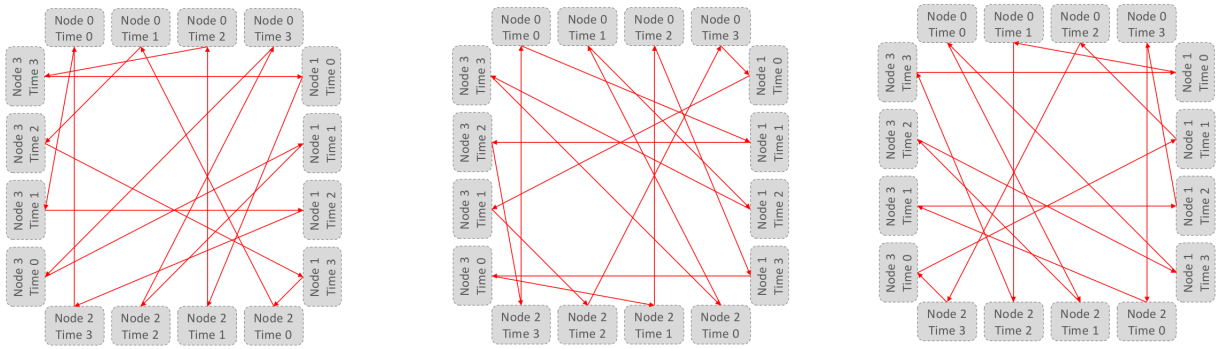
Find a Hamiltonian cycle with minimum c



1: Self-bias 2: Repetition 3: Colocation



4: Type A tours 5: Type B tours 6: Type C tours



7: Type D tours 8: Type E tours 9: Type F tours

Table 5: Graphical representation of penalties between interactions [31]

ost.

First of all, let us take a deeper look into our possible solutions. There are $(n-1)!$ cycles in a graph with n nodes. For our 4-nodes graph there are $3! = 6$ type of cycles:

- Type A: $0 \to 1 \to 2 \to 3 \to 0$
- Type B: $0 \to 3 \to 2 \to 1 \to 0$
- Type C: $0 \to 2 \to 1 \to 3 \to 0$
- Type D: $0 \to 3 \to 1 \to 2 \to 0$

- Type E: $0 \to 1 \to 3 \to 2 \to 0$
- Type F: $0 \to 2 \to 3 \to 1 \to 0$

We may see this graphically in table 6.

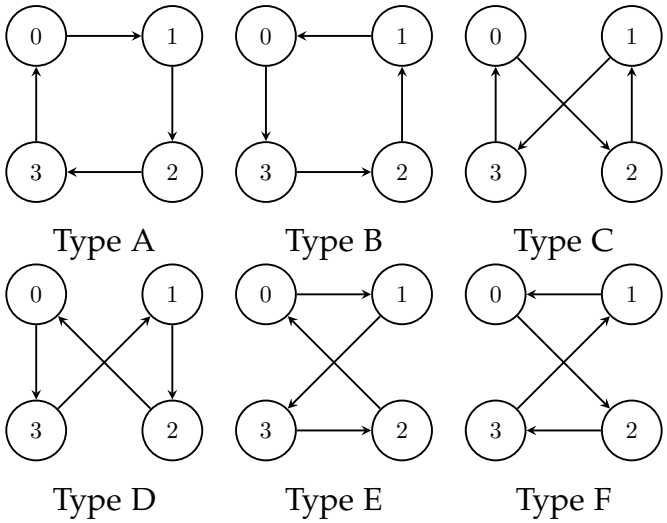
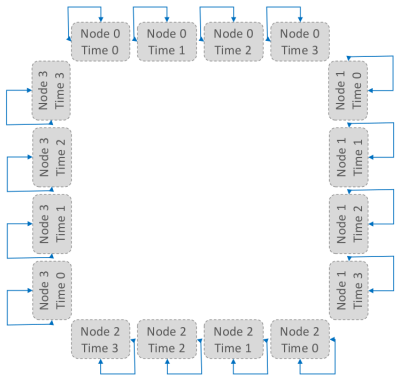


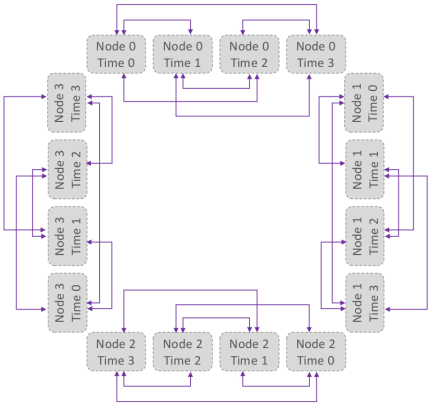
Table 6: Type of cycles in a 4-nodes graph

These six are the only possible cycles through our graph and one of them will have minimum cost. Since our binary encoding additionally depends on the node position, there are 4 solutions of our encoding that represent the same cycle. For example, $(0 \to 1 \to 2 \to 3 \to 0)$ and $(1 \to 2 \to 3 \to 0 \to 1)$ both represent cycle type A. In this case, type A will be the tour with minimum cost.

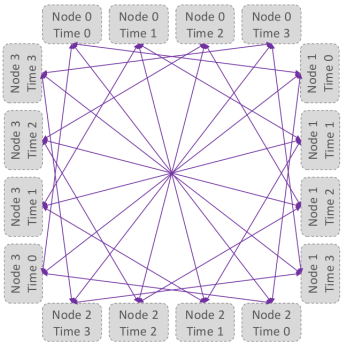
A graphical representation of the different penalties interactions is shown in figure 7, based on the constraints defined above.



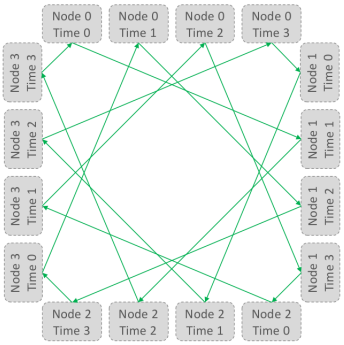
1: Self-bias



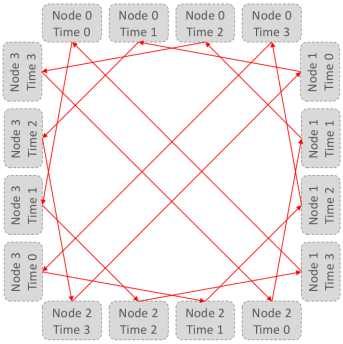
2: Repetition



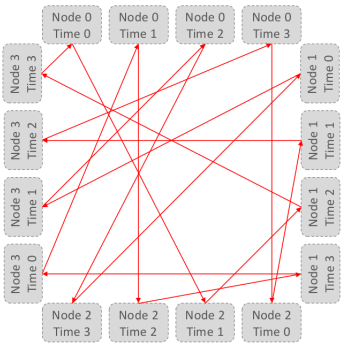
3: Colocation



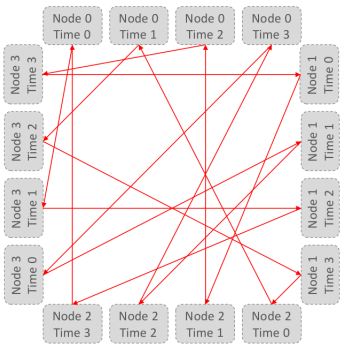
4: Type A tours



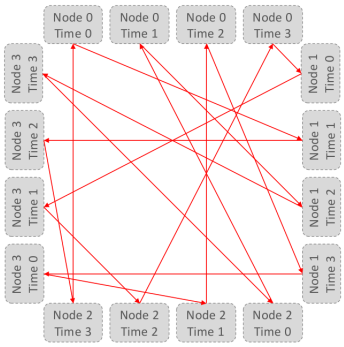
5: Type B tours



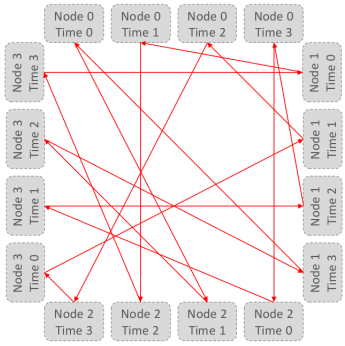
6: Type C tours



7: Type D tours



8: Type E tours



9: Type F tours

Table 7: Graphical representation of penalties between interactions [31]

We start the problem refactoring by renaming the variables:

$$(x_{0,0}, x_{0,1}, x_{0,2}, x_{0,3}, x_{1,0}, x_{1,1}, x_{1,2}, \ldots, x_{3,2}, x_{3,3}) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots, x_{15}, x_{16})$$

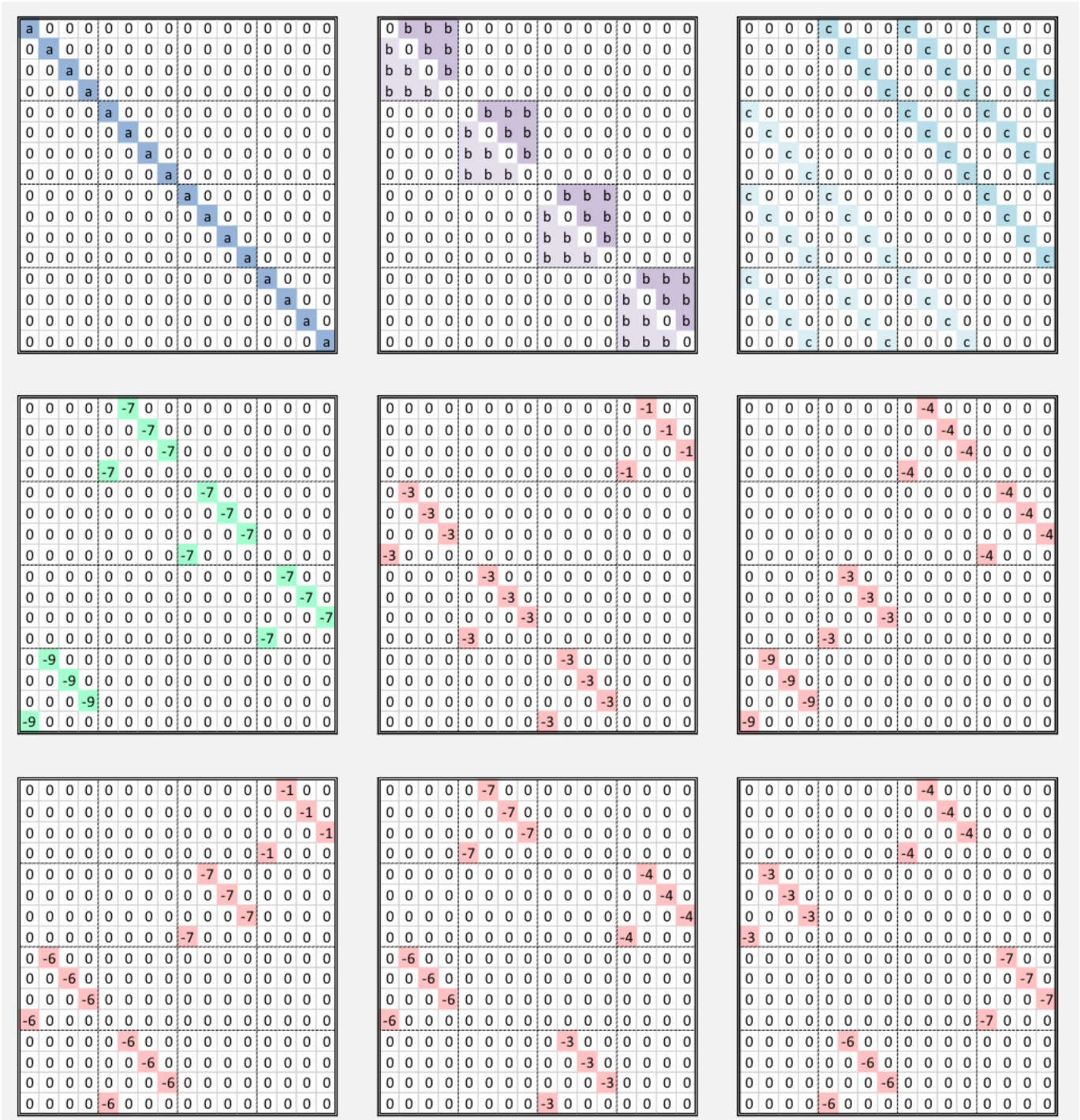


Figure 3: Term by term penalty matrices [31]

Each of the terms in the cost function 7 may be transformed into a matrix. The term associated to the weights is additionally split using the different types of cycles in which they appear. We thus obtain nine adjacency matrices, found on figure 3.

Note that the matrices associated with the constraints are all symmetric. This happens because they only depend on the graph topology and not on the weights. On the other hand, since the cost of going back and forth between a pair of nodes is not the same, the matrices associated with the cycles are never symmetric.

The final Q matrix for our problem is obtained by adding these matrices. Experimental evidence suggests that setting $a = 0$ and $b = c = 13$, and applying quantum annealing yields optimal solutions for our cost function [31]. Using these values for the penalties, the final Q matrix is seen in figure 4.

0	13	13	13	13	-14	0	0	13	-8	0	0	13	-2	0	0
13	0	13	13	0	13	-14	0	0	13	-8	0	0	13	-2	0
13	13	0	13	0	0	13	-14	0	0	13	-8	0	0	13	-2
13	13	13	0	-14	0	0	13	-8	0	0	13	-2	0	0	13
13	-6	0	0	0	13	13	13	13	-14	0	0	13	-8	0	0
0	13	-6	0	13	0	13	13	0	13	-14	0	0	13	-8	0
0	0	13	-6	13	13	0	13	0	0	13	-14	0	0	13	-8
-6	0	0	13	13	13	13	0	-14	0	0	13	-8	0	0	13
13	0	0	0	13	-6	0	0	0	13	13	13	13	-14	0	0
-12	13	0	0	0	13	-6	0	13	0	13	13	0	13	-14	0
0	-12	13	0	0	0	13	-6	13	13	0	13	0	0	13	-14
-12	0	-12	13	-6	0	0	13	13	13	13	0	-14	0	0	13
13	-18	0	0	13	0	0	0	13	-6	0	0	0	13	13	13
0	13	-18	0	-12	13	0	0	0	13	-6	0	13	0	13	13
0	0	13	-18	0	-12	13	0	0	0	13	-6	13	13	0	13
-18	0	0	13	-12	0	-12	13	-6	0	0	13	13	13	13	0

Figure 4: Final Q-matrix [31]

Finally transforming our problem into a QUBO in its matrix form, using the previous matrix:

$$\text{Minimize } f(x) = x^T Q x$$

Finding one of the following minimas:

- $x^T = [1000010000100001]$, equivalent to $x_{0,0} = x_{1,1} = x_{2,2} = x_{3,3} = 1$ and 0 otherwise. Or in path notation: $(0 \to 1 \to 2 \to 3 \to 0)$
- $x^T = [0100001000011000]$, equivalent to $x_{0,1} = x_{1,2} = x_{2,3} = x_{3,0} = 1$ and 0 otherwise. Or in path notation: $(1 \to 2 \to 3 \to 0 \to 1)$

- $x^T = [0010000110000100]$, equivalent to $x_{0,2} = x_{1,3} = x_{2,0} = x_{3,1} = 1$ and 0 otherwise. Or in path notation: $(2 \to 3 \to 0 \to 1 \to 2)$
- $x^T = [0001100001000010]$, equivalent to $x_{0,3} = x_{1,0} = x_{2,1} = x_{3,2} = 1$ and 0 otherwise. Or in path notation: $(3 \to 0 \to 1 \to 2 \to 3)$

Which corresponds to the four ways of encoding type A cycles with our notation.

2.4 D-WAVE SYSTEMS

D-Wave Systems Inc. is a Canadian company dedicated to quantum computing. In 2011, they announced the first commercial quantum computer system, D-Wave One. During the following two decades, the D-Wave team has developed a series of quantum computers dedicated to quantum annealing. The last one being the Advantage System (figure 5).

Figure 5: Advantage^{TM} system

The quantum processing unit (QPU) of this system consists of 5640 qubits and 40,484 *couplers* (links between qubits that allow entanglement between a pair of qubits). It can be seen in figure 6. The number of couplers is especially relevant. A coupler is a physical mechanism that allows two qubits to be entangled. We will explain couplers in-depth in the next section. The different D-Wave architectures are explained in section 2.4.2. In table 8, we see a comparison between the number of qubits and couplers from the different versions of the D-Wave computer system.

	D-Wave One	D-Wave Two	D-Wave 2X	D-Wave 2000Q	Advantage
Release date	May 2011	May 2013	August 2015	January 2017	2020
Qubits	128	512	1152	2048	5640
Couplers	352	1,472	3,360	6,016	40,484

Table 8: D-Wave historical comparison [32]

Since the QPU must be isolated to operate, it is encapsulated in a system at temperatures below 15 mK. In addition, radio frequency (RF)-shielded enclosure and magnetic shieldings are used to protect it from electromagnetic interference [33].

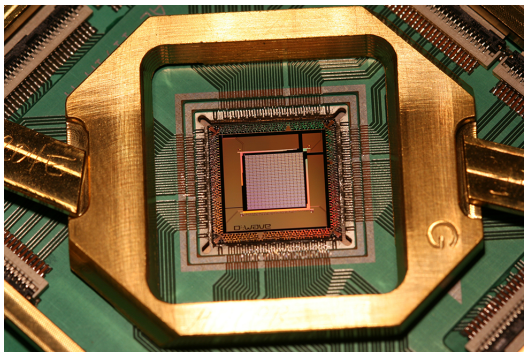


Figure 6: D-Wave QPU

D-Wave provides an easy-to-use software environment to solve problems using quantum annealing. We will explain it in-depth in section [TODOref].

2.4.1 *Quantum Annealing in D-Wave*

For this section, we refer to the D-Wave documentation, which explains how quantum annealing is implemented and may be exploited in the D-Wave systems [34].

Let us start by describing how qubits are physically implemented in D-Wave's QPU. Superconducting loops are used for this purpose, with a circulating current and a magnetic field. Depending on the direction of the current, the qubit will be in state 0 or 1; see figure 7.

In figure 8 we see a diagram with the evolution of a single qubit's energy during the annealing. By applying a magnetic field to the qubit we may affect the current and put the qubit in a superposition of both states (a). At this point, the energy is represented by a single valley with a single minimum: the superposition state where the qubit starts. Remember that at the end of the annealing we will measure every qubit and make them collapse to either one of those states, so we will never see a superposition in our measurements.

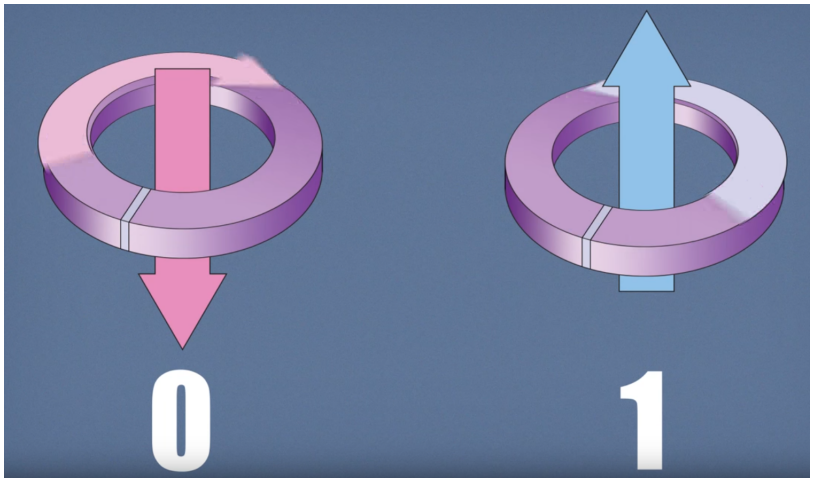


Figure 7: D-Wave's qubit, using a circulating current [34]

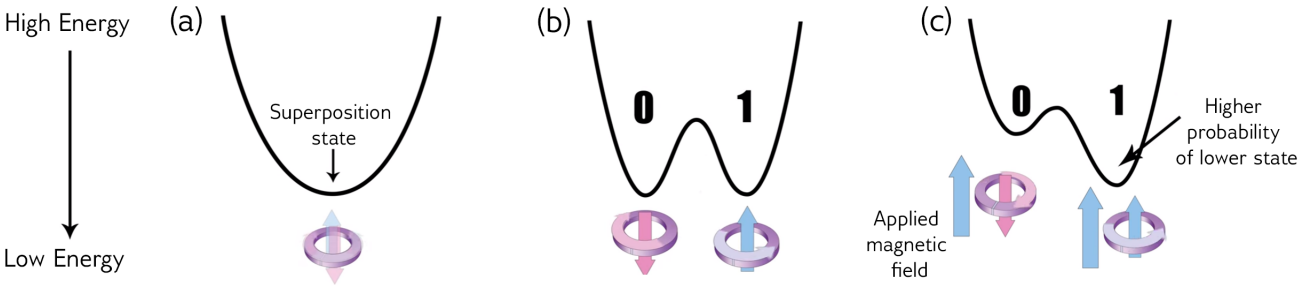


Figure 8: Energy diagram changes upon running the annealing and applying bias [34]

Then, the quantum annealing is run, turning the energy diagram into what is called a *double well potential* (b). The lower point of the left valley corresponds to the 0 state and the right one to the 1 state. If we measured the qubit now, it would end up with equal probability in either one of both states. However, we may influence the qubit's energy by applying a magnetic field and tilting the double-well to one of the base states (c), increasing the probability of ending in that state. This new magnetic field is called *bias* and the qubit minimizes the energy subject to the applied bias.

Biases applied on single qubits alone do not take full advantage of quantum annealing. We will also need *couplers*, devices that entangle qubits to each other. A coupler can make two qubits tend to end in the same state or in opposite states with certain strength. As with bias, the coupler strength, usually called *weight* may be controlled by the programmer. Together the biases and the weights are the parameters that define a problem in the D-Wave system. As the reader may already imagine, these are the parameters of a QUBO / Ising model that we may define.

Consider a 2-qubits system for the following simple example. There will be three parameters: the 2 qubits biases and the coupler weight between them. As we know, two entangled qubits may be in four possible states: $(0,0)$, $(0,1)$, $(1,0)$ or $(0,0)$. The

previous parameters define what is known as the *energy landspace*. Figure 9 illustrates this idea. The energy of each state depends on the biases and the single weight.

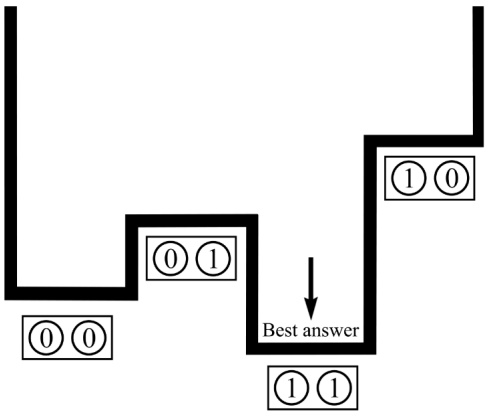


Figure 9: Example of 2-qubit energy landscape [34]

Let us take a look at the underlying quantum physics of this process. We make use of the Hamiltonian of a quantum system, which describes the evolution of the quantum system as per Postulate 3' [TODOref]. It maps the *eigenstates* of the system to their energies. Only when the system is on an eigenstate, the energy of the system is well defined and called *eigenenergy*, which are just the eigenvalues. The set of eigenstates and eigenenegies is called the *eigenspectrum*. Recall that the lowest energy state is called the *ground state*.

For a D-Wave system, the Hamiltonian may be described as:

$$H(s) = \underbrace{-\frac{A(s)}{2}\left(\sum_i \sigma_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j}\sigma_z^{(i)}\sigma_z^{(j)}\right)}_{\text{Final Hamiltonian}}$$

where $\sigma_{x,z}^{(i)}$ are the Pauli matrices X and Z respectively operating on qubit (i), h_i are the qubit biases, and $J_{i,j}$ are the couplers weights.

As explained in the adiabatic evolution section 2.2, the Hamiltonian is made of two terms:

- The Initial Hamiltonian, also called *tunneling Hamiltonian*, where the ground state is when every qubit is in superposition but no entanglement between qubit takes place.
- The Final Hamiltonian, also called *problem Hamiltonian*, where the biases and weights are such that the ground state is the solution of the problem we are trying to solve.

In fact, the final Hamiltonian is precisely an Ising model. By expressing our problem as an Ising -or, equivalently, a QUBO- we may encode it inside this Hamiltonian using the parameters h_i and $J_{i,j}$.

We may parametrize $s \in [0,1]$ as an abstract parameter that controls the annealing timing. Assuming the initial time is $t_0 = 0$, s is defined as $s \equiv t/t_{final}$. Thus, the annealing occurs when s goes from 0 to 1. We may define the annealing functions $A(s), B(s)$ such that $A(0) \gg B(0)$ and $A(1) \ll B(1)$. For instance, as seen in figure 10.

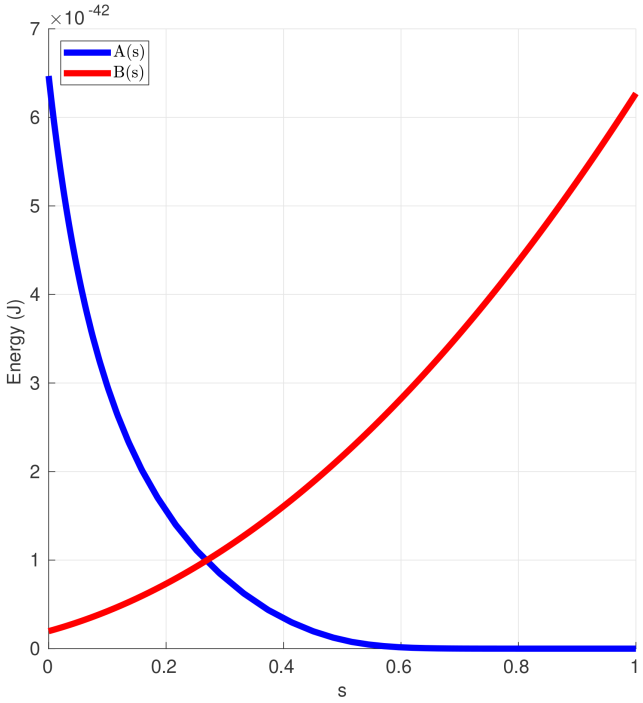


Figure 10: Annealing functions for the D-Wave 2X systems [34]

With these restrictions on the annealing functions, the system's Hamiltonian satisfies $H(0) = H_{initial}$ and $H(1) = H_{final}$. Therefore, the ground state at time $s = 0$ matches the ground state of the initial Hamiltonian, and at time $t = 1$, it matches the final Hamiltonian one. Under adiabatic evolution hypothesis, if the system starts in the initial Hamitolnian's ground state, it will end up in the final Hamiltonian one -which encodes the solution to our problem- at the end of the annealing.

The question remaining is, are the adiabatic evolution hypothesis met? We may control the time that the annealing takes by adjusting our annealing functions, a few microseconds are enough in most cases, but the exact time that would make the annealing adiabatic is never known. The additional hypothesis of the adiabatic theorem (4) mentions the gap between the ground energy and the rest of eigenenergies. Let us look at a visual representation of the evolution of the eigenenergies of a given system against time in figure 11.

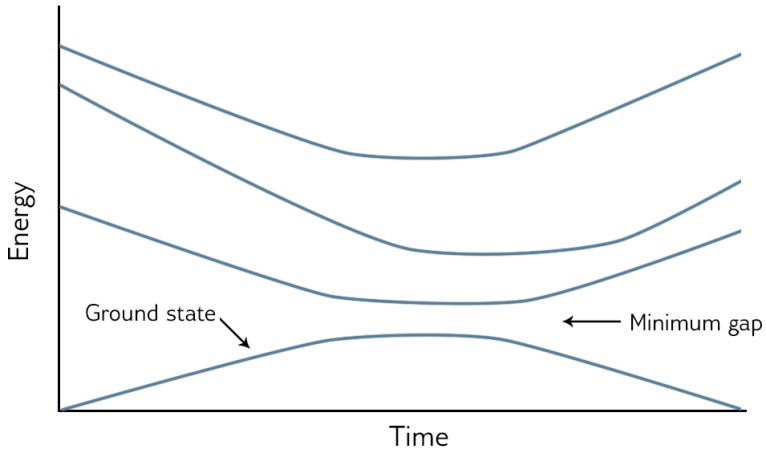


Figure 11: Example of eigenspectrum evolution [34]

The ground energy is plotted at the bottom. While the eigenenergies are far (the minimum gap holds), the probability of the system state jumping to another eigenstate is really low. When time elapses, the ground energy gets increasingly closer to the rest of the eigenenergies. The closer those energies are, the higher the probability of the system state jumping gets too. For every problem, there is a different Hamiltonian and, consequently, a different corresponding eigenspectrum. Generally, the most difficult problems for quantum annealing are those with really small minimum gaps.

In practice, these jumps may occur due to certain factors like thermal fluctuations or running the annealing process too quickly. Either way, a complete adiabatic process in the real world would mean perfect isolation, which is impossible. For some problems, the probability of staying in the ground state is really low, although low-energy states are also useful since they are associated to solutions with a low value in our cost function.

2.4.2 *D-Wave's QPU Topologies*

In table 8, we saw the number of qubits and couplers of each one of the D-Wave QPUs. These QPUs implement a graph topology where the nodes are qubits and the couplers are the edges connecting two nodes together. However, a complete n-nodes graph has $n(n-1)$ nodes, and none of the D-Wave QPUs have as many couplers. For instance, the D-Wave 2000Q has 2048 qubits and 6016 couplers, far from the $2048 * 2047 = 4,192,256$ edges that it could have. Let us explore the graph topology underneath these systems in order to understand their problems and how to work around them.

There are two different topologies being used in the D-Wave systems. The *Chimera* topology was used up until the D-Wave 2000Q system, include; while the *Pegasus* topology is only in the recent Advantage system.

In the **Chimera topology**, qubits are 'oriented' either horizontally or vertically, as seen in figure 12.

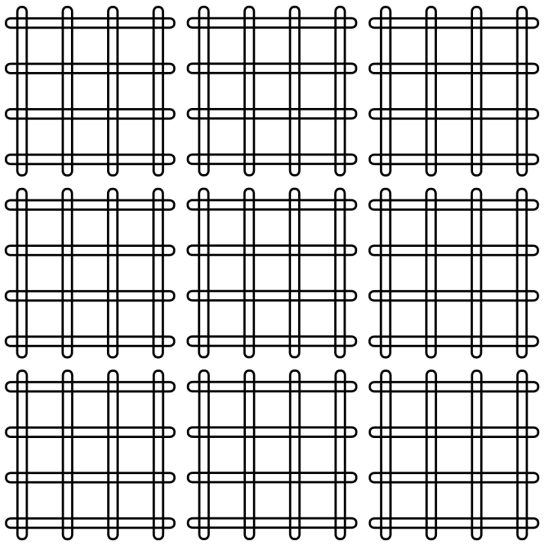


Figure 12: Qubits represented as horizontal and vertical loops. This graphic shows three rows of 12 vertical qubits and three columns of 12 horizontal qubits for a total of 72 qubits, 36 vertical and 36 horizontal. [35]

For this topology it is conceptually useful to split couplers into two categories: Internal and external couplers. The **internal couplers** connect pair of orthogonal qubits. For example, in figure 13 we see a green qubit connected to four black qubits through internal couplers.

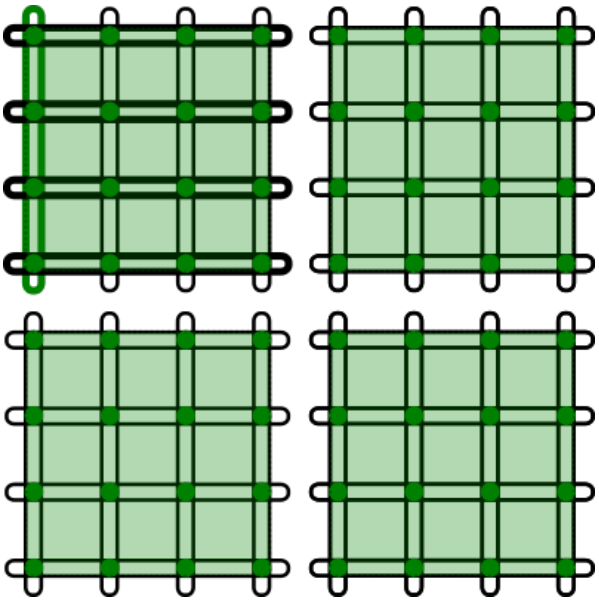


Figure 13: Internal couplers, represented as green dots at the intersections between qubits [35]

The Chimera topology has a repetitive structure where sets of 4 by 4 qubits are grouped together. These are the translucent green squares in figure 13 and are called *unit cells*.

A unit cell can also be represented either as a cross or a column, as shown in figure 14. They are $K_{4,4}$ bipartite graphs. Meaning, there are two sets of 4 qubits, qubits on a given set are connected to every qubit in the opposite set and have no more connections. same set but are connected to every qubit of the other one. For example, the green qubit labeled as 0 is connected to all the nodes from the opposing set: $(4, 5, 6, 7)$.

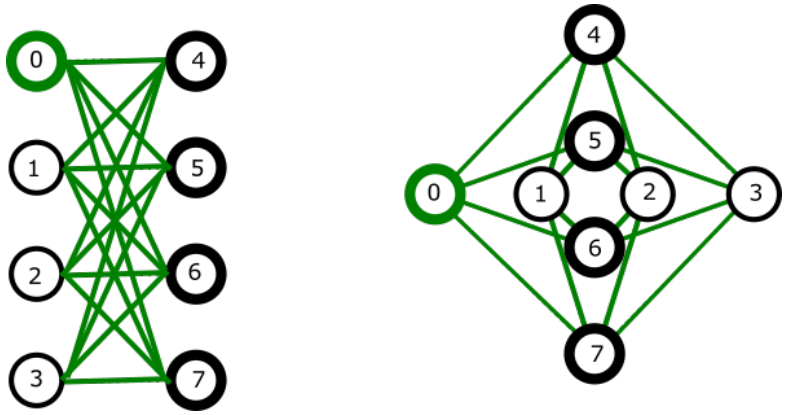


Figure 14: Chimera unit cells [35]

The **external** couplers connect pairs of qubits in the same row or column, but from different unit cells. For example, in figure 15 the green qubit in the center unit cell is connected to the two blue qubits in other unit cells and two four black qubits in the same unit cell.

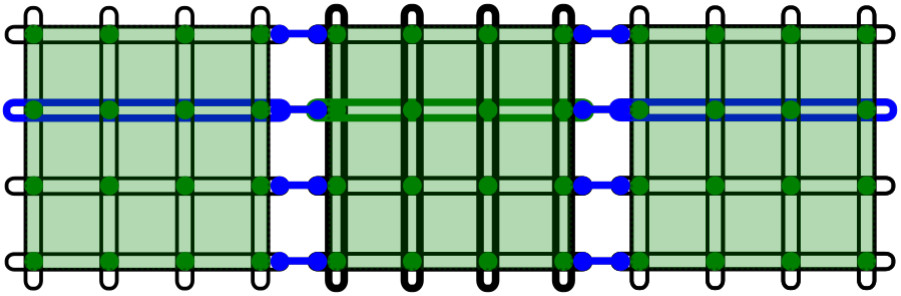


Figure 15: External couplers, represented as blue connections between qubits [35]

Chimera qubits have:

- A *nomial length* of 4. That is, each qubit is connected to 4 orthogonal qubits via internal couplers.
- A *degree* of 6. That is, is qubit is connected to a total of 6 other qubits.

The $K_{4,4}$ unit cells are connected by external couplers forming what is called a lattice. For instance, figure 16 shows four connected unit cells that are part of a bigger topology. The notation C_n refers to a chimera grid with an $N \times N$ grid of unit cells. Figure 13 shows a C_2. The D-Wave 2000Q QPU supports a C_{16} chimera graph.

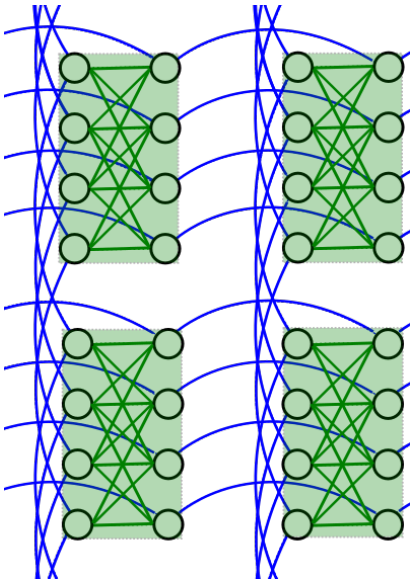


Figure 16: Cropped view of four unit cells, part of a bigger Chimera graph [35]

The **Pegasus topology** is also organized in horizontal and vertical qubits, but they are shifted, as shown in figure 17. Conceptually, the chimera and the pegasus topology are not that different: the graph is divided into unit cells, connected between them using external couplers. However, a new kind of coupler is introduced in the Pegasus topology: the odd coupler.

We will not deepen anymore into the Pegasus graph structure. It suffices to know that pegasus qubits have a nominal length of 12 (each qubit is connected to 12 orthonormal qubits using internal couplers) and a total degree 15. A Pegasus cell contains 24 qubits. Additionally, P_n represents a Pegasus topology with an $N \times N$ grid of unit cells. The Advantage system supports a P_{16} Pegasus graph.

2.4.3 *Embeddings*

Let us explore the main disadvantages of these topologiesand how to overcome them. Consider a simple example with three binary variables (a, b, c): find a configuration for the variables such that $a + b + c = 1$.

This simple satisfability problem can be transform using the energy function $E(x) = (1 - a - b - c)^2$, or equivalenty $E(x) = 2ab + 2ac + 2bc - a - b - c + 1$. This matches the following QUBO model:

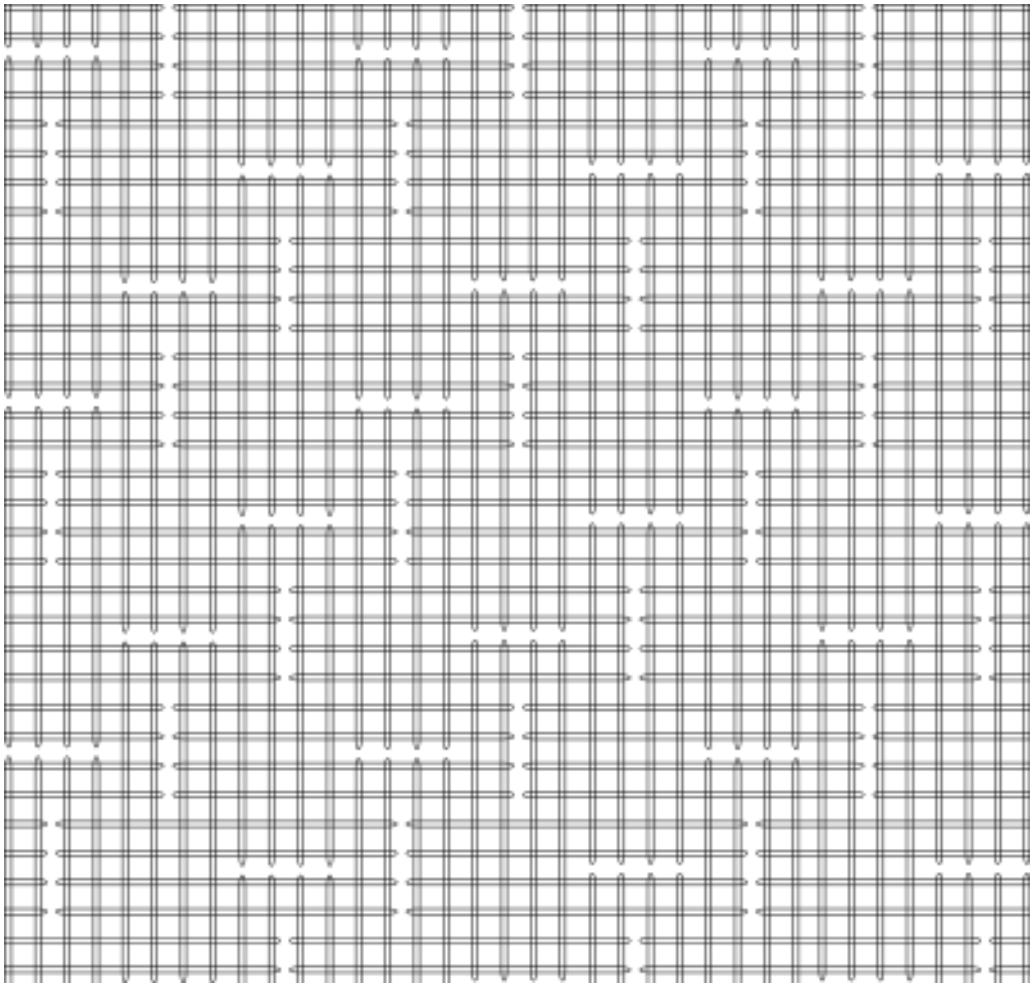


Figure 17: Pegasus graph [35]

$$\text{Minimize } E(x) = x^T Q x + 1$$

where

$$Q = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}, \quad x = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

This can be seen as the graph in figure 18. Suppose we want to use a Chimera topology to solve this problem. The remaining question is how to embed this graph into a chimera unit cell such as 19.

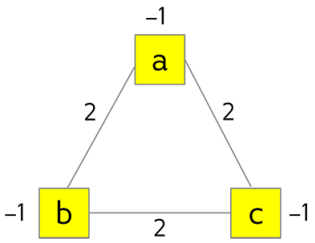


Figure 18: Graph associated to QUBO model [36]

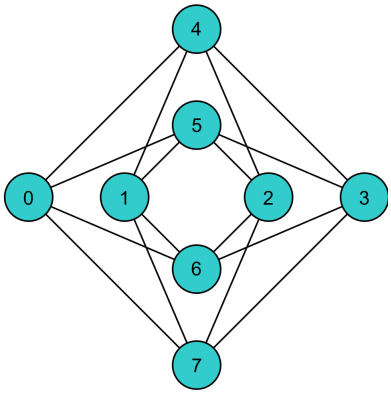


Figure 19: Chimera unit cell [36]

However, there are no three nodes in a bipartite graph that form a triangle, so a direct embedding is not possible. Instead, a *chain* is used: adjacent qubits are group together and conceptually assigned to a single node of our QUBO model graph. This process is shown in figure 20. In practice, the coupler between qubits in a chain is set to a great negative value so the probability of both qubits ending up in the same state is really high. In most complex cases we will not manually embed a graph since D-Wave provides an automatic way of computing an (at least) suboptimal embedding.

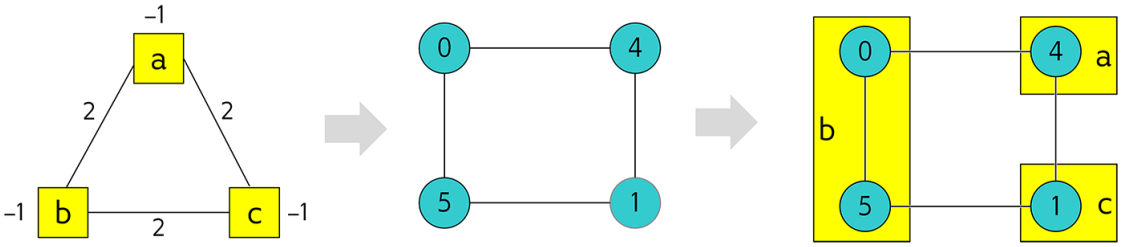


Figure 20: Embedding process [36]

3

SOLVING THE DE NOVO GENOME ASSEMBLY

In this last chapter, we approach the genome assembly problem with the use of quantum annealing. Let us first understand what this problem is, and then translate it to a QUBO model and solve it using quantum annealing.

3.1 THE GENOME ASSEMBLY PROBLEM

The genome of an organism is all its genetic material [37]. The deoxyribonucleic acid (DNA) is the carrier of that genetic information. It consists of two long chains twisted to form a double helix [38]. Each of these chains is composed of a series of nucleotides or bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Since these bases are matched in pairs in the DNA double helix, they are called base pairs (bp).

A genome sequence is the complete list of nucleotides of every chromosome of an organism. With today's technology, automated sequence machines can read up to 1000 bp at a time [39] while the human genome contains 3 Mbp, so we cannot simply read the whole genome. This is where genome assembly comes in.

Genome assembly refers to the process of, given a large number of short DNA reads, stitch them together to form a large representation of the original chromosome where the reads came from. The two main techniques used to reconstruct these sequences are the ab initio reference-free alignment and the de novo reference-based assembly.

3.1.1 *Ab initio reference-based alignment*

In this method, the DNA reads are matched against a known trusted reference of the same organism. This is essentially a pattern matching problem, where we find the index of a given sub-string in a larger string. However, after the reconstruction is complete the result is compared to the reference in order to identify implications; therefore introducing bias based on the reference [31].

In the naive approach, the short sub-string is compared to the reference starting at the first index. If the end of the sub-string is reached with a positive, a match is obtained. Otherwise, the sub-string is shifted a single position and we compare again. Heuristic methods that improve on this idea are based on shifting a greater number of spaces after a mismatch.

Different number of strategies have been developed in this direction. For instance, the classic Boyer-Moore and Knuth-Pratt-Morris algorithms [40]. However, these in particular are not adequate for genome assembly since these are exact string matching algorithms and DNA reads usually need approximate matches due to reads errors. Other algorithms worth mentioning are the Needleman-Wunsch algorithm [41] and the Smith-Waterman algorithm [42], for global and local alignment respectively. These are dynamic programming algorithms designed specifically with DNA reads in mind.

State of the art algorithms trades off accuracy for speed and memory. Given enough computational power, the de novo reference-free method yields better results without introducing any reference bias.

3.1.2 *De novo reference-free assembly*

On the other hand, the de novo reference-free method is, as its name suggests, reference free. Meaning, it is based only on the DNA reads. Thus, it has no reference bias but it is more computationally complex. It is usually used the first time a species DNA is read.

In this technique, multiple copies of the same DNA are made before slicing it. After chopping each copy at random places the data is redundant and the different reads overlap, making the assembly easier. There are multiple methods for de novo assembly based on different tools: Overlap-Layout Consensus (OLC) methods, de Bruijn graph (DBG) methods, string graphs, greedy and hybrid methods are some of the most famous examples. Depending on the reading method and the number and length of the DNA reads, different methods excel from the rest. For instance, short-read technologies with a large number of reads favor DBG methods while high-quality long reads favor OCL methods. For our purposes, we will focus on the OCL method.

In the OLC graph used for the de novo whole-genome assembly, each node represents a different DNA read. Directed edges are associated a weight depending on how well these two reads are stitched together in a certain order. For example, the directed edge going from reading r_1 to read r_2 will be assigned a weight depending on how well $r_1 r_2$ can be stiched.

The weights computation depends on the implementation. For the purpose of this thesis, we will use exact matches, no taking reading errors into consideration. Then, the weight assigned to an edge is the length of the overlap between both reads without

any errors, with a change of sign. For example, given the reads $r_1 = AATT$ and $r_2 = TTCC$, the perfect stitching will produce $AATTCC$, so the overlap between both reads is 2, giving a weight of -2. We may call this the *distance* between reads r_1 and r_2 (in that order). It does not fulfill the mathematical definition of distance since it is not even symmetric, but it will be useful for us anyway.

A Hamiltonian path in our overlap graph will represent a series of reads in a certain order. By minimizing the total cost of our Hamiltonian path, we maximize the overlap between reads, resulting in the shortest possible final chain. This is exactly the same as solving the Travelling Salesman problem associated with our overlap graph.

Figure 21 shows the whole problem resolution [43]. Given the DNA reads (a) we compute the overlap graph (b) using a distance between the reads. We continue by viewing this problem as a traveling salesman and transforming it into a QUBO model (c). Then, either a simulated annealer (e) or the D-Wave quantum annealer (d) are used to obtain the Hamiltonian path/cycle of minimum cost (f). Finally, we traverse the cycle and build the resulting genome sequence.

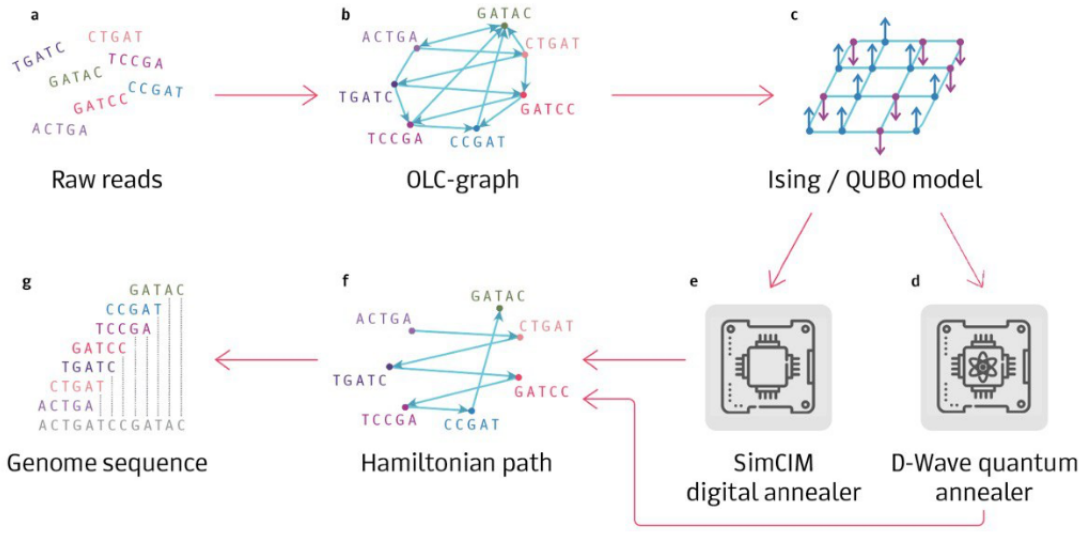


Figure 21: Resolution diagram for the genome assembly using quantum annealing [43]

NUMERICAL EXAMPLE

Let us study a final numerical example based on [31] that shows the whole process. Suppose we are given the following reads:

- $r_0 = ATGGCGTGCA$
- $r_1 = GCGTGCAATG$
- $r_2 = TGCAATGGCG$

- $r_3 = AATGGCGTGC$

We may compute the overlap between the different reads. This results in the overlap graph from figure 22.

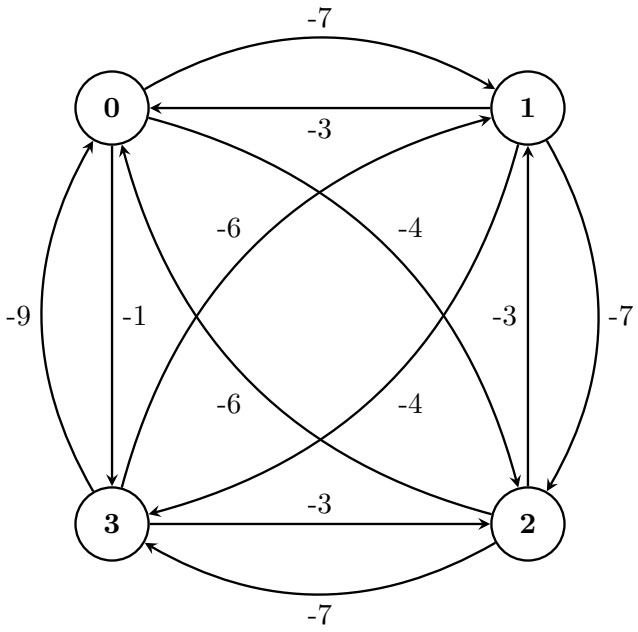


Figure 22: OLC graph

Which is the same graph studied in the traveling salesman section, 2.3.5. From the study done in that section, we know that there are six types of cycles in the graph. They are displayedin table 6. In figure 23, we see how these types of cycles represent different ordinations of our DNA reads, as well as their overlaps and the total length of the resulting chain.

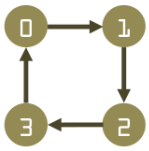
As we know, the type of Hamiltonian cycle that minimizes the cost is type A. This type translates into the shortest chain, with a total length of 19. We can easily compute the resulting assembly by traversing the graph and weaving the reads together.

3.2 EXPERIMENTATION

In this last section, we will explain the experiments reproduced using the D-Wave systems.

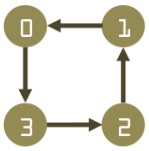
3.2.1 *How to reproduce the experiments*

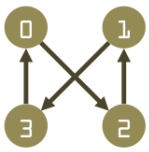
In order to run configure the samplers and submit jobs to D-Wave quantum annealers we used the *D-Wave Ocean Software*, a suite of tools provided by D-Wave to use their




```
Type A : 0 1 2 3 0
ATGGCGTGCA
___GCGTGCAATG
_______TGCAATGGCG
__________AATGGCGTGC
ATGGCGTGCAATGGCGTGC
Cost = -(7+7+7+9) = -30
Length = 19
```

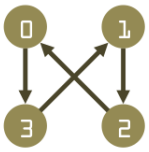
```
Type B: 0 3 2 1 0
ATGGCGTGCA
__________AATGGCGTGC
__________________TGCAATGGCG
________________________GCGTGCAATG
ATGGCGTGCAATGGCGTGCAATGGCGTGCAATG
Cost = -(1+3+3+3) = -10
Length = 33
```

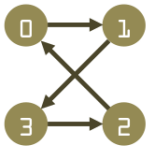





```
Type C: 0 2 1 3 0
ATGGCGTGCA
_______TGCAATGGCG
_____________GCGTGCAATG
____________________AATGGCGTGC
ATGGCGTGCAATGGCGTGCAATGGCGTGC
Cost = -(4+3+4+9) = -20
Length = 29
```

```
Type D: 0 3 1 2 0
ATGGCGTGCA
__________AATGGCGTGC
_____________GCGTGCAATG
__________________TGCAATGGCG
ATGGCGTGCAATGGCGTGCAATGGCG
Cost = -(1+6+7+6) = -20
Length = 26
```






```
Type E: 0 1 3 2 0
ATGGCGTGCA
___GCGTGCAATG
__________AATGGCGTGC
__________________TGCAATGGCG
ATGGCGTGCAATGGCGTGCAATGGCG
Cost = -(7+4+3+6) = -20
Length = 26
```

```
Type F: 0 2 3 1 0
ATGGCGTGCA
_______TGCAATGGCG
__________AATGGCGTGC
_____________GCGTGCAATG
ATGGCGTGCAATGGCGTGCAATG
Cost = -(4+7+6+3) = -20
Length = 23
```

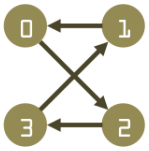


Figure 23: Type of cycles and their corresponding overlap analysis [31]

quantum systems [44]. Using python and the provided packages we may connect to *Leap*, a 'quantum' cloud service that provides access the quantum computers [45].

Leap provides a minute of free QPU time with the (free) developer plan. We may extend this time by obtaining either commercial or research access. This can be done by contacting the Leap team.

For the pourpose of this thesis I used the developer plan. In total, I submitted 104 jobs to Leap, out of the allowed 106. I consumed 59,064 QPU seconds, as can bee seen in Leap's breakdown in figure 24. The number of jobs and QPU time used in each system can be seen in table 9.

System	Number of jobs	QPU time (s)
D-Wave 2000Q	26	11.015
Advantage	77	46.681
Total	104	59.064

Table 9: Leap usage summary breakdown between systems.

Apart from Leap, in order to reproduce this experiments you will need to install tthe following Python packages, available thorugh *pip*: ***dimod***, ***neal***, ***minorminer***, ***dwave***

and ***dwave_networkx***. The code I used for this experiments can be found in [46], which are inspired on [31].

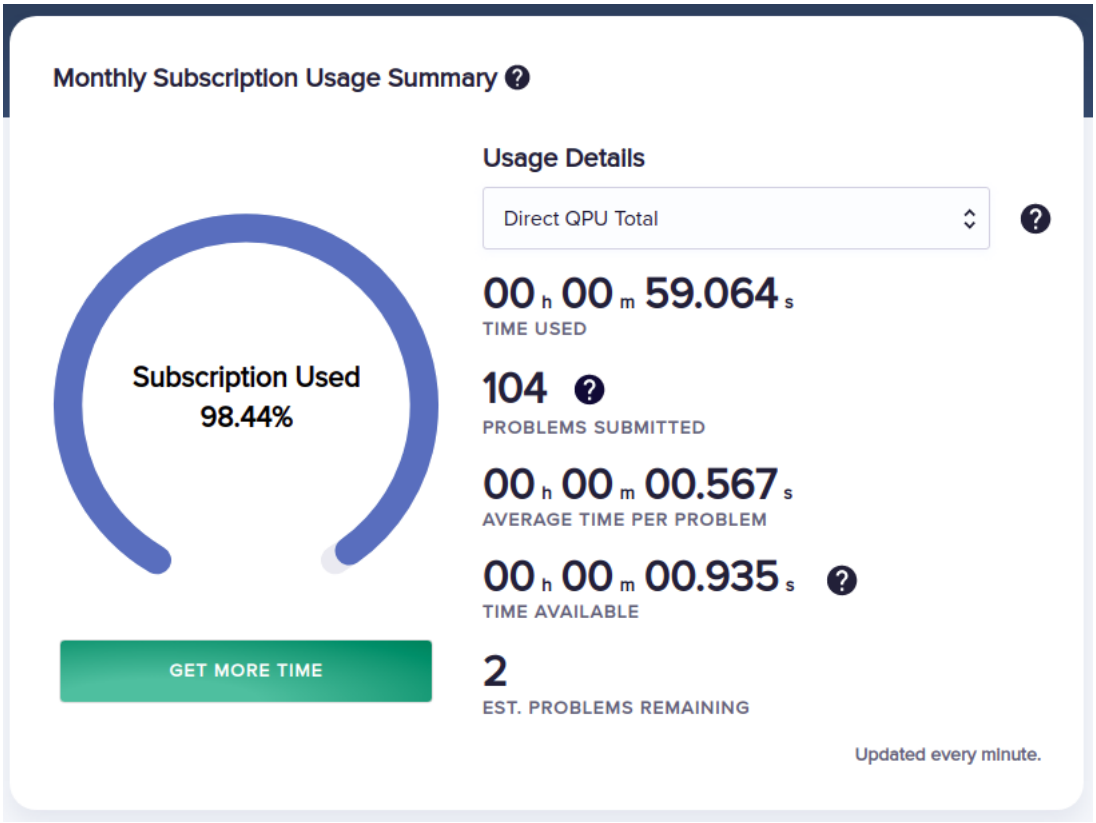


Figure 24: Leap monthly usage summary.

3.2.2 *Experiment 1: Data preparation and exact solving*

The first experiment aims to solve the previously discussed example using an exact solver provided by the Ocean's *dimod* library [47]. The steps taken to transform the given reads to the QUBO matrix that Ocean's classes and functions use will be shared between the simulated annealer and the real quantum solvers. Therefore, the data preparation applied will be shared between experiments. The pseudocode of the experiment can be seen in the following snippet.

As we already know, using quantum annealing does not guarantee that we will obtain the optimal solution for a given cost function. In order to overcome this problem we use multi-sampling: running the experiment multiple times and and look at the best obtained solutions. Ocean already implements different types of *samplers* to facilitate this task [48]. In this first experiment we use an *ExactSolver*, which simply check every possible solution. Although time-costly, this method will let us know if our data manipulation before solving the experiment is correct, and how the solutions landscape looks like.

Experiment 1

Data preparation:

- Compute the distance between every two reads, creating an adjacency matrix.
- Transform the TSP adjancency matrix into a QUBO Q matrix, as explained in section 2.3.5.
- Transform the QUBO matrix into an adjacency dictionary.

Solving:

- Initiaize a sampler: ***dimod.ExactSolver***.
- Solve the prepared QUBO model using the selected solver.

Present the results.

The third step in data preparation is a formatting step. Ocean requires the QUBO and Ising models to be in an adjacency dictionary instead of a matrix. Let us look at an example of this transformation in order to better understand it. Consider a 2-reads, the associated TSP will have $2^2 = 4$ nodes: $n0t0$, $n0t1$, $n1t0$ and $n1t1$. Suppose the following (inconsistency) matrix is the associated Q matrix:

$$Q = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

Then, it will be transformed into the dictionary, with 11 missing entries:

```
quboDict: {
        ('n0t0', 'n0t0'): 1,
        ('n0t0', 'n0t1'): 2,
        ('n0t0', 'n1t0'): 3,
        ('n0t0', 'n1t1'): 4,
        ...
        ('n1t1', 'n1t1'): 16,
}
```

In particular, this experiment is applied to the already studied example with the following four reads:

- $r_0 = ATGGCGTGCA$
- $r_1 = GCGTGCAATG$
- $r_2 = TGCAATGGCG$
- $r_3 = AATGGCGTGC$

The pair-wise distances are computed, providing a direct measure of much two reads overlap. Then, these values are normalized for easier use. The resulting normalized

TSP matrix is transformed into a QUBO matrix using 1,6 as multi-location and repetition penalties, and $-1{,}6$ for self-bias, as done in [31]. Finally, we initialize an *ExactSolver* and use it to sample every possible solution.

After the execution is completed, we find the lowest energy in the obtained solutions and print every solution with that energy:

```
{'n0t0': 0, 'n0t1': 0, 'n0t2': 1, 'n0t3': 0,
 'n1t0': 0, 'n1t1': 0, 'n1t2': 0, 'n1t3': 1,
 'n2t0': 1, 'n2t1': 0, 'n2t2': 0, 'n2t3': 0,
 'n3t0': 0, 'n3t1': 1, 'n3t2': 0, 'n3t3': 0} --> -7.9811

{'n0t0': 0, 'n0t1': 1, 'n0t2': 0, 'n0t3': 0,
 'n1t0': 0, 'n1t1': 0, 'n1t2': 1, 'n1t3': 0,
 'n2t0': 0, 'n2t1': 0, 'n2t2': 0, 'n2t3': 1,
 'n3t0': 1, 'n3t1': 0, 'n3t2': 0, 'n3t3': 0} --> -7.9811

{'n0t0': 1, 'n0t1': 0, 'n0t2': 0, 'n0t3': 0,
 'n1t0': 0, 'n1t1': 1, 'n1t2': 0, 'n1t3': 0,
 'n2t0': 0, 'n2t1': 0, 'n2t2': 1, 'n2t3': 0,
 'n3t0': 0, 'n3t1': 0, 'n3t2': 0, 'n3t3': 1} --> -7.9811

{'n0t0': 0, 'n0t1': 0, 'n0t2': 0, 'n0t3': 1,
 'n1t0': 1, 'n1t1': 0, 'n1t2': 0, 'n1t3': 0,
 'n2t0': 0, 'n2t1': 1, 'n2t2': 0, 'n2t3': 0,
 'n3t0': 0, 'n3t1': 0, 'n3t2': 1, 'n3t3': 0} --> -7.9811
```

As expected, these are the four minimums of the cost functions, representing the four ways of describing a type A loop in the used codification.

Finally, let us plot the landscape of solutions for the given reads. We sort the solutions by increasing energy and simply plot their energy, as seen in figure 25.

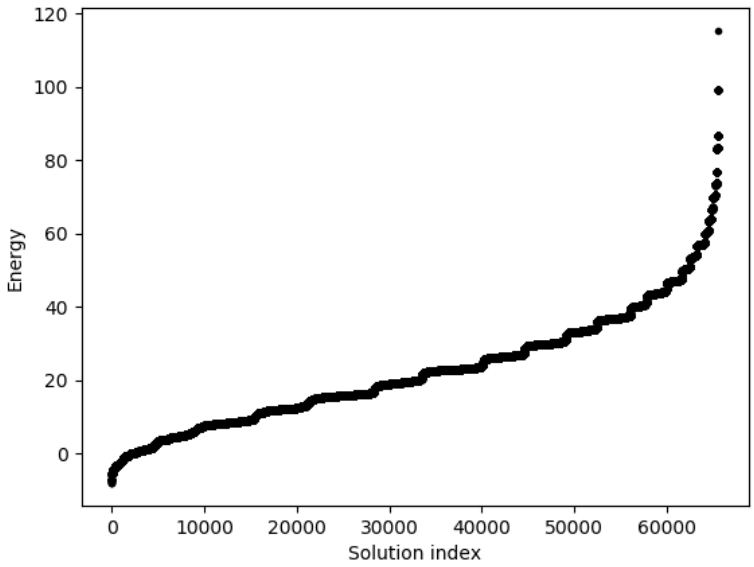


Figure 25: Landscape of solutions

3.2.3 *Experiment 2: Simulated Quantum Annealing*

For our second experiment, we aim to solve our example using Simulated Annealing. Although *dimod* provides a simulated annealing sampler, we use for our final experiments the *SimulatedAnnealingSampler* from *Ocean*'s *Neal* package as it yields more distributed results and has far better computing time. The same experiment was reproduced 10 times using both samplers in my machine. Dimod's sampler run in a mean time of 11,76 seconds while *Neal*'s run in a mean time of 0,0610 seconds.

The pseudo-code does not change much from the first experiment to the second one, we simply change the annealer:

Experiment 2

Data preparation:

- Compute the distance between every two reads, creating an adjacency matrix.
- Transform the TSP adjancency matrix into a QUBO Q matrix, as explained in section 2.3.5.
- Transform the QUBO matrix into an adjacency dictionary.

Solving:

- Initiaize a sampler: ***neal.SimulatedAnnealingSampler***.
- **Sample from** the prepared QUBO model using the selected **sampler**.

Present the obtained samples.

It is worth mentioning that we do not *solve* the QUBO model in this experiment, we *sample* different results using simulated annealing.

Since *Neal*'s sampler is quite efficient we were able to execute experiments with up to 10,000 repetitions of the experiment, also called *sample reads* or simply *reads*. We developed an automated and scalable way to check whether a given cycle returned by the annealer was valid, and to recover its associated type. Using these tools we can easily check the number of occurrences each type of cycle appeared in the obtained sample reads. These results are presented, along with the associated energy to each result, in table 10 and figure 26. See figure 23 to recall the cycle types and cost computing.

Cycle type	Occurences	Energy
Type A	3722	-7.9811
Type C	1474	-7.4541
Type D	1469	-7.4541
Type F	1458	-7.4541
Type E	1431	-7.4541
Type B	446	-6.927

Table 10: Results of experiment 2

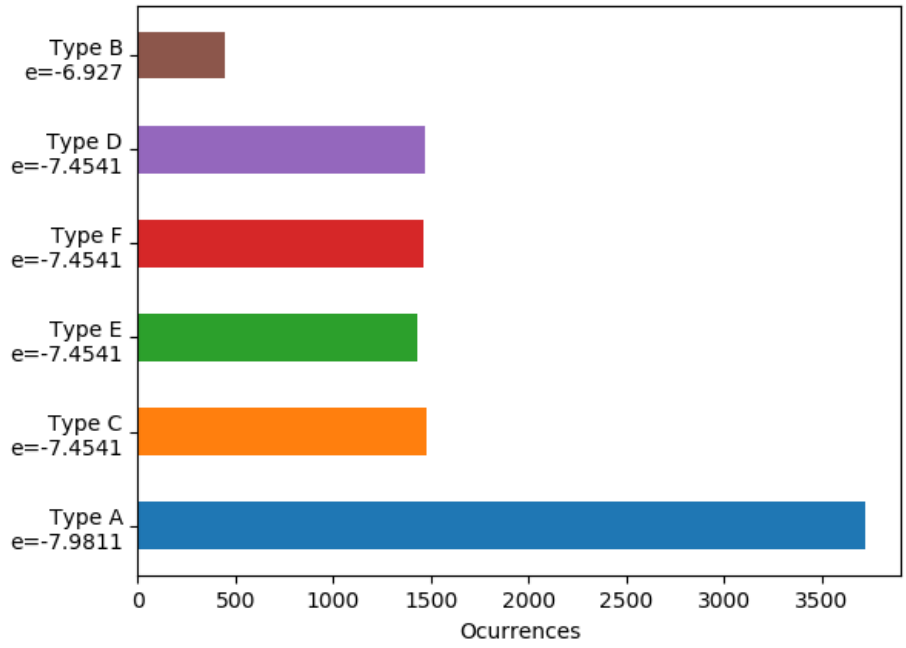


Figure 26: Ocurrences of each type in a 10,000 reads experiment using Simulated Annealing

We can appreciate in figure 26 that the best type of cycle agglomerates most of the samples, 37,22 %. In the second place, the four types of cycles that have the exact same energy also have almost exactly the same number of samples. This matches our quantum annealing theory: the physical system has an equal probability of ending in eigenstates with equal eigenenergies.

It is worth mentioning that there was not a single sample that encoded an invalid cycle. This means that the penalties values used for the experiment (1,6 for multi-location and repetition, and $-1{,}6$ for self-bias) are working to prevent invalid cycles.

Now that we are familiar with the type of results and codifications we may find, let us jump to the quantum realm by using the D-Wave's Quantum Computer Systems.

3.2.4 *Experiment 3: Quantum Annealing using D-Wave*

For our third experiment, we aim to solve our example using D-Wave's quantum annealers. For this purpose, we need to add some extra steps to our data processing. Section 2.4.3 explained how not every graph can be directly mapped to the existing Chimera / Pegasus topologies that D-Wave systems use, and how we can overcome this problem by embedding our problems graphs into these topologies. The extra steps in our data processing are related to these embeddings: we will need to embed our graph into the fixed topology that will be used. Then, after the anneal takes place, we will translate the solutions from the embedded graph. Ocean provides a set of utilities to deal with these embeddings [49].

Additionally, we will need to connect to Leap to send our jobs to the quantum annealers using a *client*. First, a configuration file is created, which includes your API key and some extra configuration details:

```
[ocete]
solver = {"qpu": true}
token = DEV-<api key goes here>
endpoint = https://cloud.dwavesys.com/sapi
```

In our code we initialize a *client* and a *solver* using the configuration file. These are objects that encapsulate the connection functionality and solving problems in the associated machine respectively. We also initialize a sampler as we did before, but this time a production sampler is used: a *DWaveSampler* that also needs our configuration file. The pseudo-code for this experiment can be found below. Each new step will be explained in detail below.

The first thing to be noticed is that the initialization now needs to be done before the data preparation. This is because, in order to find the embedding, we need to know which exact topology the quantum system will have. This information is provided through the *solver*, previously initialized. The initialization step is as simple as follows:

```
import dwave

# Create the solver (connecting to D-Wave) and the Sampler
```

Experiment 3

Initialization:

- Initiaize a sampler: ***system.samplers.DWaveSampler***.
- **Initialize a client and obtain an associated solver**.

Data preparation:

- Compute the distance between every two reads, creating an adjacency matrix.
- Transform the TSP adjacency matrix into a QUBO Q matrix, as explained in section 2.3.5.
- Transform the QUBO matrix into an adjacency dictionary.
- **Find an embedding from our graph to the selected machine's topology (either Chimera or Pegasus)**.
- **Use the previous embedding to create a new QUBO model equivalent for the embedded graph**.

Solving:

- Use the solver to send a job to the client. This job samples from the prepared QUBO model multiple times.

Format the results:

- **Use the computed embedding to translate our answers**.

```
config_file='../dwave.conf'
client = cloud.Client.from_config(config_file, profile='ocete')
solver = client.get_solver()
dwsampler = system.samplers.DWaveSampler(config_file=config_file)
```

Suppose Q already holds are computed QUBO model. We can compute an embedding and obtained the new associated model as follows:

```
adjacency_dict = embedding.utils.edgelist_to_adjacency(solver.edges)
embedding = minorminer.find_embedding(Q, solver.edges)
Q_embedded = embed_qubo(Q, embedding, adjacency_dict)
```

By using the same configuration file, the sampler already knows what machine it is associated to. We will sample from it, with the same syntax as in the second experiment:

```
response = dwsampler.sample_qubo(Q_embedded, num_reads=num_reads)
```

Where *num_reads* is a parameter that sets the number of samples to be read. However, these solutions are associated to the *Q_embedded*, not to our original Q model. We need to translate the responses to understand them:

```
bqm = dimod.BinaryQuadraticModel.from_qubo(Q)
unembedded_response = embedding.unembed_sampleset(response, embedding, bqm)
```

For this experiment, the D-Wave 2000Q was used (see section 2.4 to see its characteristics), since it was the default solver. After running the experiment with the same parameters as the simulated annealing experiment (10,000 reads and $(-1{,}6; 1{,}6; 1{,}6)$ for the QUBO parameters), we obtained that more than 90 % of the samples were invalid, a huge difference with the astonishing 0 % obtained in the second experiment.

My initial hypothesis was that the QUBO parameters were well adjusted for SA but not for QA, and thus allowed further tunning. After many attempts with different parameters values, this hypothesis was discarded. The only other set of parameters worth presenting is $(-1{,}5; 1{,}5; 1{,}5)$. These results are presented in table 11.

Cycle type	**Occurences (param=1.5)**	**Occurences (param=1.6)**	**Energy**
Type A	131	50	-7.9811
Type C	124	46	-7.4541
Type D	40	91	-7.4541
Type F	55	117	-7.4541
Type E	112	118	-7.4541
Type B	99	64	-7.4541
Invalid	9221	9194	$>$ -5.6433

Table 11: Results of experiment 3, 10,000 reads using the quantum annealer.

In figures 27 and 28 we see a comparison of the distribution between the different type of cycles in both experiments. We cannot see a distribution as we did in 26, potentially due to the low numbers of reads that actually represent a valid cycle. In fact, such distribution between the cycle types is not needed: a single sample with the lowest energy solution will completely solve our genome assembly problem.

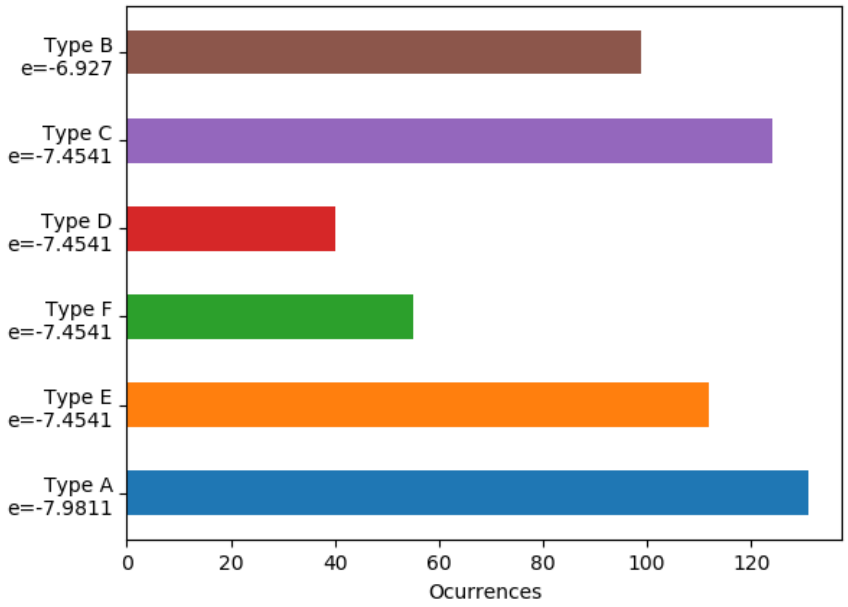


Figure 27: Ocurrences of each cycle type in a 10,000 reads experiment using Quantum Annealing, filtering out invalid cycles, with QUBO parameters $(-1{,}5; 1{,}5; 1{,}5)$

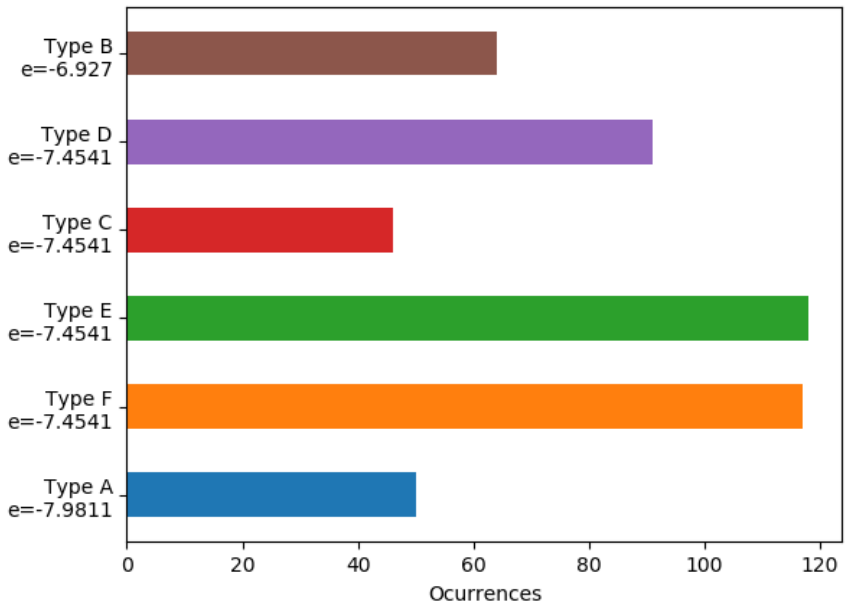


Figure 28: Ocurrences of each cycle type in a 10,000 reads experiment using Quantum Annealing, filtering out invalid cycles, with QUBO parameters $(-1{,}6; 1{,}6; 1{,}6)$

With these results in mind, is it possible that we are sampling completely random solutions? In a 4-reads problem, we obtain a 16 binary variables QUBO model. That means there are up to 2^{16} possible solutions using our encoding, even more after we embed it in the Chimera topology. If we were to randomly sample from that solution

space, we will never get almost 10 % of samples from a subset of 24 solutions that represent our 6 valid cycles. So we do know that the quantum system is working, just know as good as expected.

We have not found a clear improvement precision-wise in our small example, although QA does solve the problem. What about time-wise? In figure 29 we see the time breakdown provided by Leap after our last experiment. We can see a total QPU sampling time of 2,389 seconds, while if we repeat the experiment with the same parameters using SA we obtain a mean of 2,525 sampling seconds, over 10 experiment repetitions. Keeping in mind that our QA time data comes from a single execution and the small difference between both time measures, we cannot conclude that either approach has any time advantage in such small problems.

However, does QA actually scales better than SA with the problem size? In the next experiment, we will study the Pegasus topology to understand what is the maximum problem size that can be tackled using the Advantage system. Finally, the last couple of experiments will try to answer the scalability comparison inquiry.

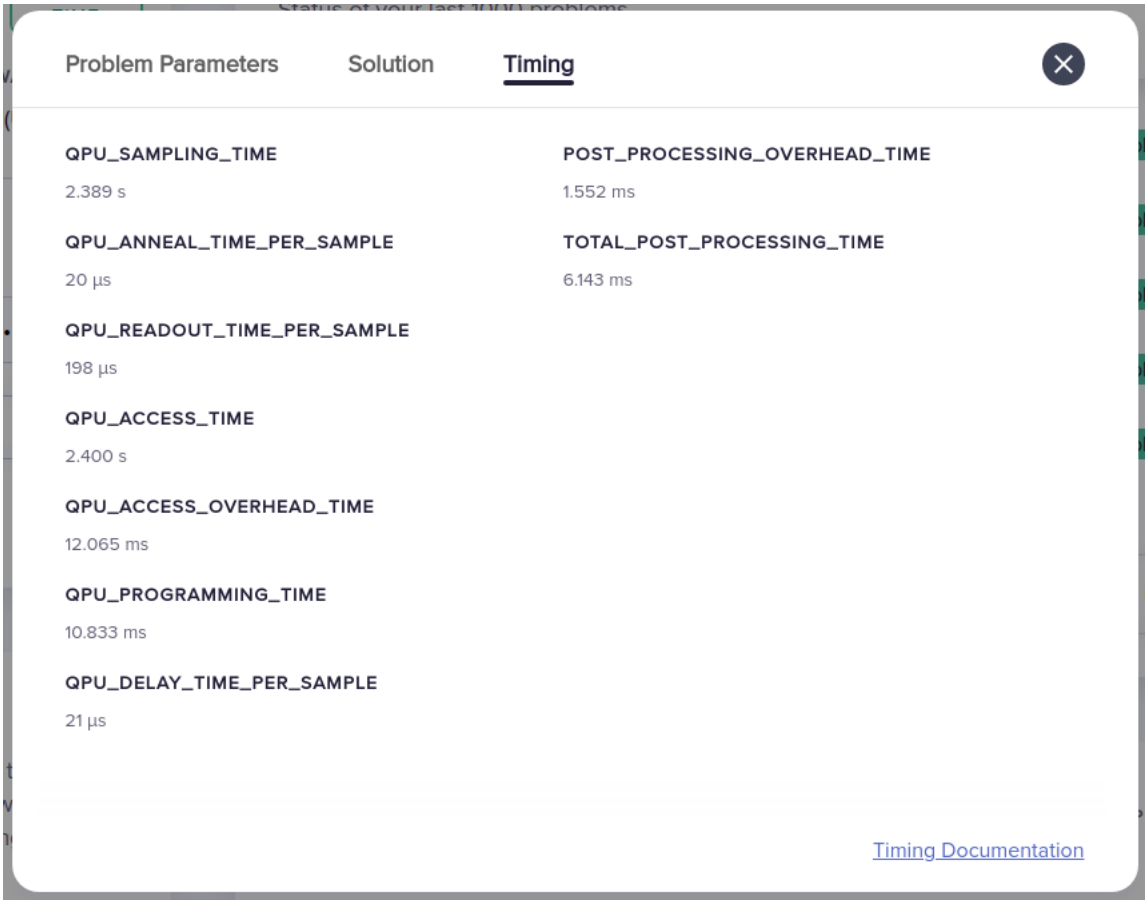


Figure 29: Time breakdown provided by Leap after the last experiment.

3.2.4.1 *Plotting chimera embeddings*

Let us plot some logical embeddings using the D-Wave 2000Q system, with relies on a Chimera graph. In figure 30 we see the embedding of a 4-nodes complete graph into the Chimera topology. The grey nodes represent unused nodes while nodes with the same color represent a single logical qubit mapped into multiple physical qubits.

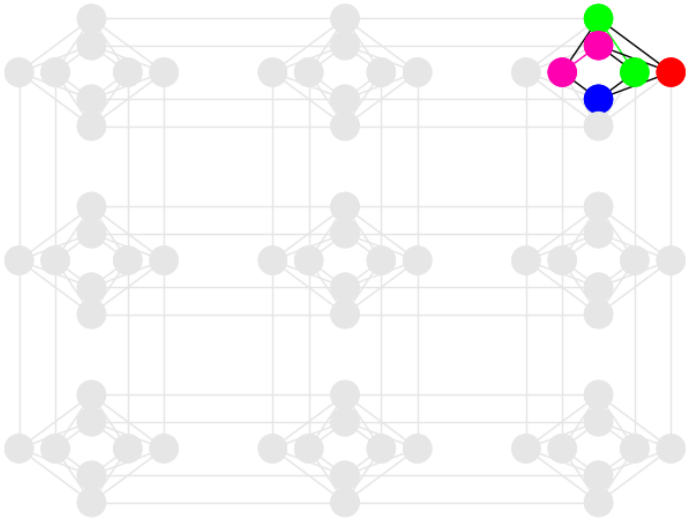


Figure 30: Embedding of a 4-nodes complete graph into a Chimera topology.

The last embedding had a *maximum chain length* of 2. That is, the longest chain in a logical qubit is 2 physical qubits. This is a critical parameter to be minimized in embeddings: the longer the qubit chain is, the more probable it is for the qubits in a logical qubit to be desynchronized and end up in different states. Thus, the more difficult it is for the system to stay in the ground state.

In figure 31 we see an embedding of a 13-nodes complete graph into a Chimera topology. In this case, the maximum chain length is 5, obtained by both the dark blue, purple, red, dark green, lime, and cyan logical qubits. In the rest of the experiments, I looked at the maximum chain length obtained in the pegasus embeddings, obtained lengths up to 15 qubits. Since I did not have the time to study this characteristic in-depth, it will be a future line of work.

Recall that in every 16-nodes chimera cell, every node in the vertical axis is connected to every node in the horizontal axis, and to no other node from the vertical axis. For example, take the blue logical qubit, represented by 3 nodes at the very bottom in figure 31. In the first cell, it connects with lime, pink, dark green, and cyan. In the second cell, it connects with light green, purple, red, and dark blue. In the third cell, it connects to brown, light blue, pink, and yellow. That makes a total of 12 connections, as expected since the embedded started with a complete 13-nodes graph.

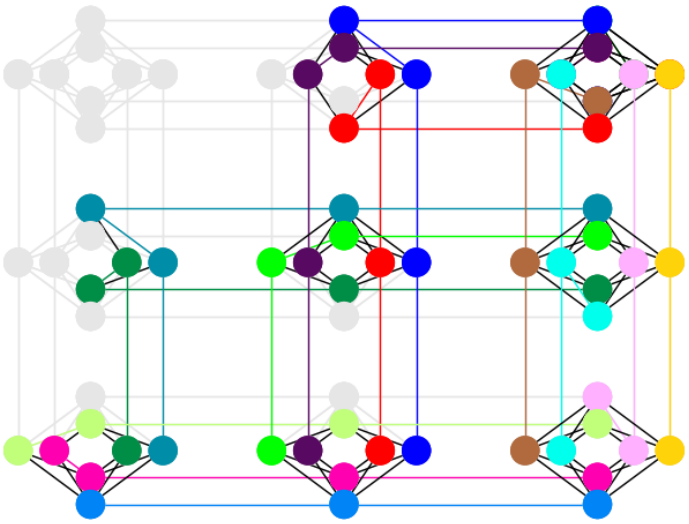


Figure 31: Embedding of a complete 13-nodes complete graph into a Chimera topology.

3.2.5 *Experiment 4: D-Wave's Advantage limits*

For our fourth experiment, we aim to test the limits of the Advantage architecture for our problem. That is, given a number of DNA reads, we will compute the associated QUBO problem and try to embed the obtained graph into the Advantage architecture. We will test these embeddings against a Pegasus 16 (P_{16}) topology since that is the one supported by the Advantage system.

In fact, given a set of n reads, this is simply finding if a n^2-nodes complete graph (K_{n^2}) can be embedded into a P_{16}. The reads themselves are not relevant for the embedding. However, I have developed an automated way of creating fake tests, originally for a general-purpose, but it can also be used here.

This automated test production works as follows: Given a number of reads, the size of each chain (default value 150), and the required overlap between adjacent chains (default value 50), it randomly creates the original DNA chain, and then chop it using the given parameters. This makes sure that adjacent reads have enough overlap so it is basically impossible for two random reads to overlap more than 50 bps.

Using these random tests we created graphs of an increasing number of reads until we were unable to find an embedding from K_{n^2} to P_{16}. The results of the experiment can be seen in table 12. We can see that a valid embedding was found to problem size up to 14 reads (K_{196} associated graph). The embedding time, as well as the test generation time, are also displayed. I decided to display both since, as time grew larger and larger, I was not sure whether the test generation was adding too much of an overload.

Number of reads	Nodes in graph	Test generation time	Embedding time	Total time
2	4	0.000157	0.081310	0.081708
3	9	0.000216	0.108525	0.109205
4	16	0.000269	0.194807	0.196045
5	25	0.000335	0.377673	0.379814
6	36	0.000392	1.147603	1.154559
7	49	0.000433	3.514801	3.523473
8	64	0.000497	11.876785	11.888235
9	81	0.000558	19.226392	19.244367
10	100	0.000610	43.457666	43.477268
11	121	0.000676	62.499896	62.528545
12	144	0.000724	89.277396	89.313831
13	169	0.000812	63.782066	63.827706
14	196	0.000840	152.262340	152.316223

Table 12: Results of experiment 4

It is worth noting that the algorithm used to find the embeddings does not guarantee to find a valid one if it exists. I tried to run the algorithm 50 times with 15 reads and could not find a valid embedding.

The reader might have already noticed that the embeddings are quite time-consuming. In fact, this is an NP problem itself and time will grow exponentially on input size. However, keep in mind that this embedding does not depend on the reads. Once a single embedding from a K_{196} to a P_{16} has been pre-computed it can be used -or further optimized- for different problems.

Now that we know the maximum number of reads that the Advantage system may tackle, let us put it to the test with 'big' input problems in the next experiment.

3.2.6 *Experiment 5: Scalability comparison between Simulated Annealing and Quantum Annealing*

In our fifth experiment, we put the D-Wave's quantum annealers to the test with bigger input problems and study how they behave compared with the simulated annealing experiments. We will start with a 3 reads and then progressively increase this value, creating tests of this size and trying to solve them using both the simulated annealer and the Advantage system.

For each fixed number of reads, we will sample 10,000 times using the annealers, with the same set of parameters used for the previous experiments. The number of

valid solutions (i.e. valid cycles) will be displayed, along with the number of cycles that reached the real solution, which we know in advance since we created the test.

We will also display the energy of the best sample obtained, along with the difference between the real solution energy and the best sample obtained, which I called *energy delta*. Finally, we will also display the sampling time, not including the time it took to create the test nor to prepare the data.

The results of this experiment using the simulated annealer can be seen in table 13.

N. reads	Valid cycles	Times sol. reached	Energy delta	Solution energy	Sampling time
3	10000	9564	0	-6.214214	1.881374
4	10000	6852	0	-8.131935	2.178147
5	10000	3427	0	-9.999001	3.487972
6	10000	1308	0	-11.834192	5.121054
7	10000	351	0	-13.656181	7.057235
8	10000	108	0	-15.443562	9.680610
9	10000	21	0	-17.232667	12.797996
10	10000	3	0	-18.995278	16.080461
11	10000	0	0.625356	-20.764680	20.118828
12	10000	0	0.601820	-22.510011	25.351562
13	10000	0	0.570361	-24.256735	30.612547
14	10000	0	1.378872	-26.005003	34.897780

Table 13: Results of experiment 5, 10,000 reads using the simulated annealer.

We stopped at 15 reads since that is the maximum value that will be able to compare with using the quantum annealer. In the experiment results, we see how the number of valid cycles out of 10,000, which turns out to be every single cycle up to 14 reads. However, the number of samples that reached the best solution go down from 95 % to ground cero from 11 reads onward. Although we do not obtain the real solution, how close are these samples to it?

The columns related to the energy of the solutions try to answer this question. We may see how the solution's energy goes down to around -26, which is totally normal: the more reads in our graph, the higher the number of reads overlap, and the lower is the resulting solution energy. As soon as we cannot find the solution, the energy delta starts to increase, going up to around 1,37. This means the energy of the best sample is quite close to the actual value. Is this enough?

When we tackle a classic TSP, a near-optimal solution is a really valid and useful solution. We do not necessarily need the actual minimum. However, although a valid cycle with low energy represents a valid way of sorting and sewing the genome

reads, it does not provide the real genome the reads came originally from. It has some value since it could be further tuned to obtain the solution, but it does not solve our problem.

Lastly, the sampling time seems to grow linearly with the number of reads. However, using this simple simulated annealer for a real case with up to 3 Mbps would be simply impossible.

For the second part of this experiment, I run the same experiment but using the quantum annealer. Some extra steps come with it, like computing the embedding and connecting to Leap, but they have all been explained in previous experiments. We again sample 10,000 times a maintain the same values for the annealing schedule self-bias, multi-location, and repetition parameters. The results of this experiment can be seen in table 14.

N. reads	Valid cycles	Times sol. reached	Energy delta	Solution energy	Sampling QPU time
3	4574	3295	0	-6.214214	1.035
4	683	156	0	-8.131935	0.962
5	88	6	0	-9.999900	1.272
6	2	0	-1.322536	-11.834014	1.272
7	1	0	-1.647457	-13.671186	1.206
8	0	0	-3.867139	-15.444996	1.339
9	0	0	-36.592409	-17.227226	1.570
10	0	0	-50.03568	-18.468571	1.276

Table 14: Results of experiment 5, 10,000 reads using the quantum annealer.

The first thing we may notice is that the experiment did not reach 14 reads. That is because I manually stopped it before it finished. The available QPU time was quite tight at the moment, and seeing the results of the experiment with a number of reads up to 10 showed that we were not going to learn anything new by letting the experiment finished and wasting our remaining QPU time.

Back to the results, we can appreciate the number of valid cycles rapidly goes down to 0, and even quicker the number of reads that reached the solution. At 4 reads we see similar results to what we saw in experiment 3: around 700 valid cycles and 150 times the real solution (type A cycle in the third experiment) was reached. Although we have developed this experiment in different machines, D-Wave 2000Q and Advantage, keep in mind that the improvements from one generation to another are size-related: we may tackle bigger (and thus harder) problems by using the Advantage system. It does not mean that its computation is somehow better.

Not only the responses are not valid, but they are also quite far from the solution. We can see that in how the delta grows hugely as we increase the number of reads. Leap

provides a graph for each submitted experiment displaying the samples' distribution. In figures 32 and 33 we see this distribution for 4 and 10 reads respectively.

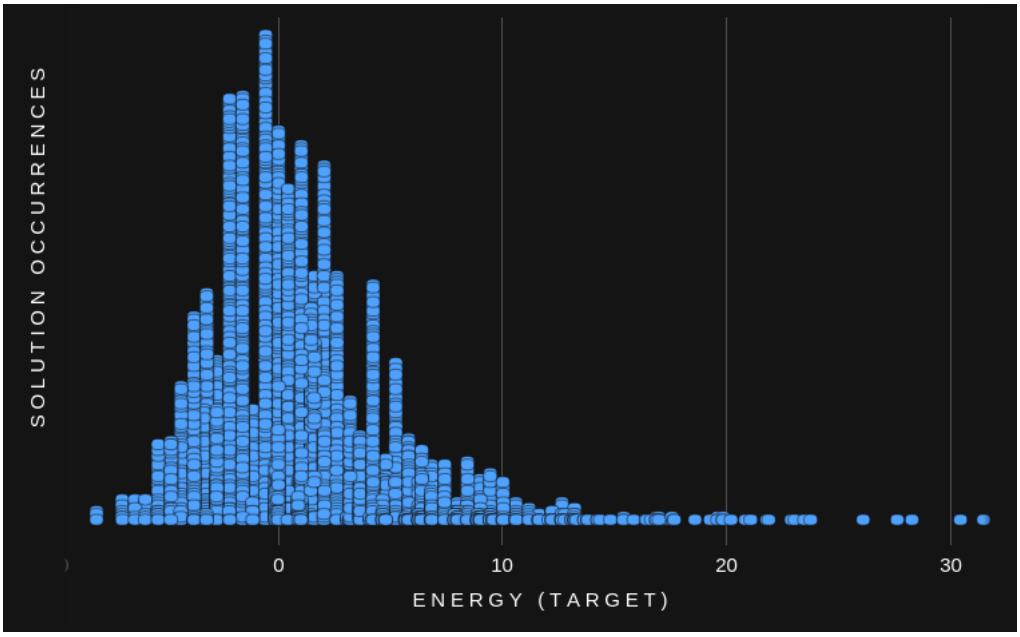


Figure 32: 10,000 samples' distribuition for 4 reads using the quantum annealer.

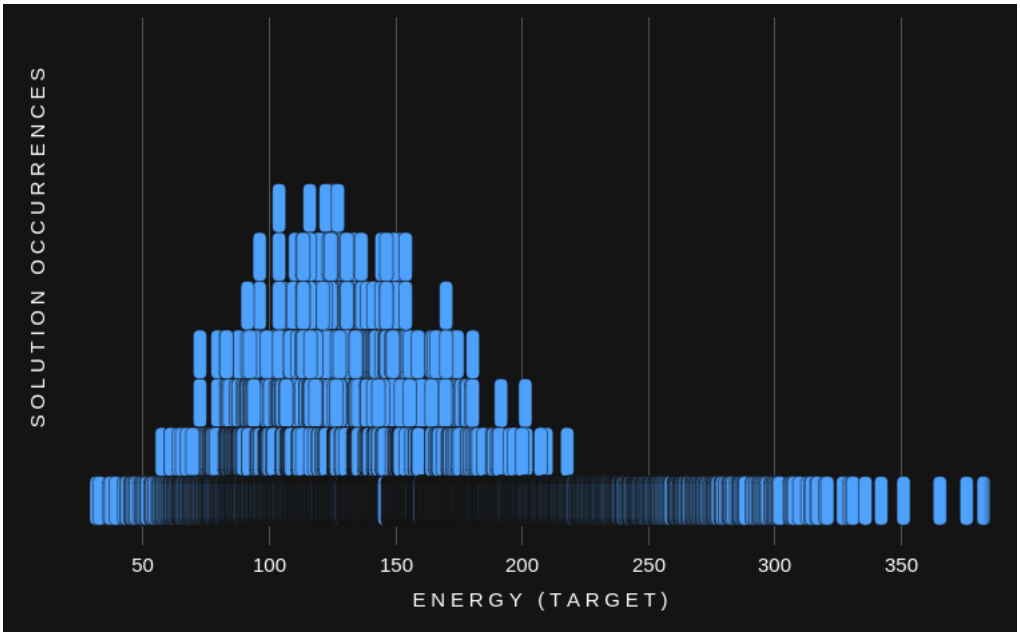


Figure 33: 10,000 samples' distribuition for 10 reads using the quantum annealer.

We can see how in the 4 reads experiment, most of the samples group around 0 energy, and only a very few are close to the solution, which is actually reached in this execution. However, in the 10 reads experiment, most of the samples are gathered

around 100/150 energy, these solutions are not only invalid but really far from being so.

Lastly, let us comment on the QPU sampling time. This value is provided directly by Leap. From our code, we cannot separate the connection overload and other computations from the actual QPU sampling time. We may appreciate how time does not grow with the input. This is because the time it takes for the annealer to converge is fixed by the annealing schedule.

In conclusion, these results are astonishingly awful. The simulated annealing sampler not only provides closer samples to the solution, but in every seen case it reaches valid samples. My current hypothesis is that either the annealing schedule is not optimal, or that the values of the parameters can be improved for our specific problem. In our last experiment, I will spend all our remaining QPU time checking these two hypotheses.

3.2.7 *Experiment 6: Advantage tunning*

For our last experiment, we aim to test our previous hypothesis and try to solve a 10 reads problem using the Advantage system. In the first set of iterations, I tried to increase the multi-location and repetition penalties in order to increase the number of valid cycles sampled. For this experiment, a single test with 10 reads was created. Its solution energy was $-18{,}994150$. The results of the experiment can be found in table 15, where the mentioned parameters are *(self-bias, multi-location, repetition)*.

Parameters	Valid cycles	Times sol. reached	Energy delta	Sampling QPU time
(-1.6, 1.6, 1.6)	0	0	-76.344431	1.500
(-3, 3, 3)	0	0	-87.263801	1.624
(-5, 5, 5)	0	0	-83.657639	1.272
(-10, 10, 10)	0	0	-258.290739	1.248
(-20, 20, 20)	0	0	-357.917063	1.311
(-50, 50, 50)	0	0	-1017.605383	1.441
(-100, 100, 100)	0	0	-1418.541757	1.261

Table 15: Results of experiment 6, 10,000 reads using the quantum annealer for different parameters configurations.

As we can see in the table, we did not find a single valid cycle in these iterations. The energy delta naturally increases as we increase the parameter values: every single penalty is greater penalized. However, the energy delta barely increases until our parameters get a value of 10. This could show how the responses for $parameters = (-5, 5, 5)$ incurred in a lower number of penalties. In figure 34 we find the samples

distribution for *parameters* $= (-5, 5, 5)$. The overall energy is much higher than what we saw for *parameters* $= (-1{,}6, 1{,}6, 1{,}6)$ in figure 33, which occurs due to the higher penalty. This experiment concludes that either we have not been able to find the correct values for the parameters, or the values of the parameters are not the problem.

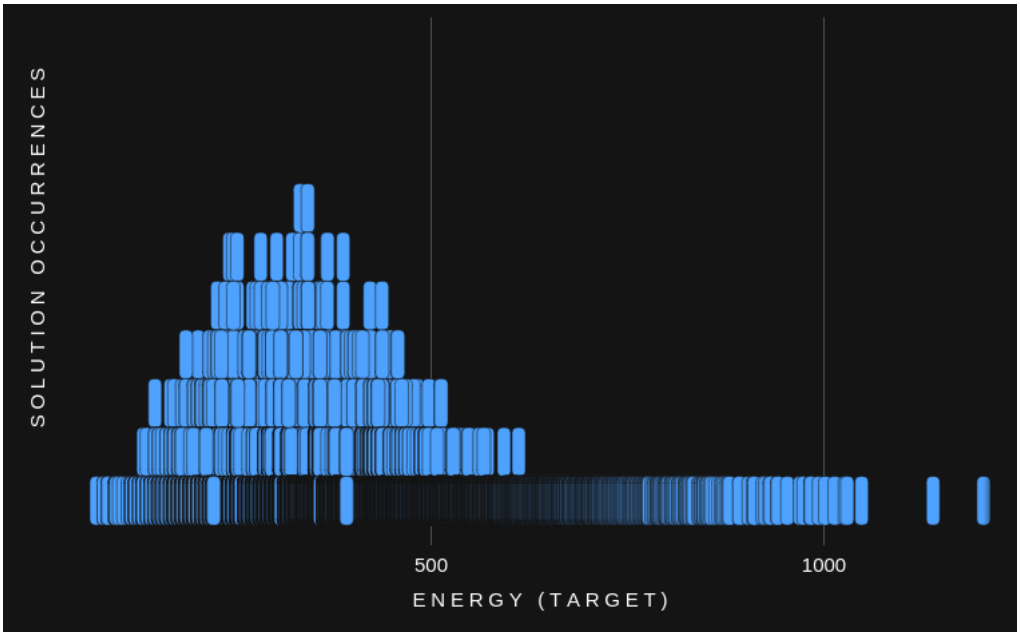


Figure 34: 10,000 samples distribuition for *parameters* $= (-5, 5, 5)$.

Finally, we try changing the annealing schedule. There are two possible parameters that we may modify in this sense: the *annealing time* -time let for the annealer to converge-, or the *annealing schedule* -which modifies the annealing functions as defined in figure 10. Since my initial hypothesis was that the annealer did not have enough time to find good solutions, let us focussed on the former for our last experiment.

The anneal time is meassured in microseconds, with a default value of 20. Although the initial idea was to start with that value and make it progressively higher, at this point I was able to submit only one more worthy experiment, with a anneal time of 40 microseconds. This was due to a bug in the D-Wave *get_solver*() function, which did not read the annealing time correctly and led to some invalid experiments at the last minute. We can see the obtained results in table 16.

Annealing time (μs)	**Valid cycles**	**Times sol. reached**	**Energy delta**	**Sampling QPU time**
20	0	0	-76.344431	1.500
40	0	0	-57.944162	1.573

Table 16: Results of experiment 6, 10,000 reads using different annealing times.

Although the energy delta is reduced, the improvement is marginal: we still find no valid cycles and the responses have a high penalty. Probably with even more annealing time we will obtain further refined results.

BIBLIOGRAPHY

[1] Michael A. Nielsen, Isaac Chuang, and Lov K. Grover. *Quantum Computation and Quantum Information* , volume 70. 2002. ISBN 9781107002173. doi: 10.1119/1.1463744.

[2] Daniel Manzano. A short introduction to the Lindblad master equation. *AIP Adv.*, 10(2), feb 2020. ISSN 21583226. doi: 10.1063/1.5115323.

[3] Pablo Bayens. Bachelor Thesis: Modelos de computación cuánticos, 2019. URL https://github.com/mx-psi/tfg/blob/master/tfg.pdf.

[4] Rafael Payá. Apuntes de Análisis Funcional. In Rafael Payá, editor, *Apunt. Análisis Func.*, chapter 5, pages 53–62. Granada, 2020. URL https://www.ugr.es/~rpaya/documentos/Funcional/2020-21/Apuntes_05.pdf.

[5] E. Schrödinger. Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften*, 23(48):807–812, nov 1935. ISSN 00281042. doi: 10.1007/BF01491891.

[6] John Trimmer. The present situation in quantum mechanics: A translation of Schrödinger'sÇat Paradox"paper. *Proc. Am. Philos. Soc.*, 124(5):323–338, 1980. ISSN 0003-049X. URL https://archive.is/20121204184041/http://www.tuhh.de/rzt/rzt/it/QM/cat.html#sect5.

[7] Dietlinde Lau. *Function Algebras on Finite Sets*. 2006. doi: 10.1007/3-540-36023-9.

[8] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Inf. Comput.*, 6(1):081–095, jan 2006. ISSN 15337146. doi: 10.26421/qic6.1-6. URL https://arxiv.org/abs/quant-ph/0505030v2.

[9] J. S. BELL. on the Einstein Podolsky Rosen Paradox. Technical Report 3, 1995.

[10] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47(10):777–780, may 1935. ISSN 0031899X. doi: 10.1103/PhysRev.47.777. URL https://journals.aps.org/pr/abstract/10.1103/PhysRev.47.777.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science (80-.).*, 220(4598):671–680, may 1983. ISSN 00368075. doi: 10.1126/science.220.4598.671. URL https://science.sciencemag.org/content/220/4598/671https://science.sciencemag.org/content/220/4598/671.abstract.

[12] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chem. Phys. Lett.*, 219(5-6):343–348, mar 1994. ISSN 00092614. doi: 10.1016/0009-2614(94)00117-0.

[13] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, 58(5):5355–5363, 1998. ISSN 1063651X. doi: 10.1103/PhysRevE.58.5355.

[14] G Nimtz and A Haibel. Zero time space: how quantum tunneling broke the light speed barrier. *Choice Rev. Online*, 46(02):46–0955–46–0955, 2008. ISSN 0009-4978. doi: 10.5860/choice.46-0955. URL https://inis.iaea.org/search/search.aspx?orig_q=RN:39067461.

[15] P. Ray, B. K. Chakrabarti, and Arunava Chakrabarti. Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations. *Phys. Rev. B*, 39(16):11828–11832, 1989. ISSN 01631829. doi: 10.1103/PhysRevB.39.11828.

[16] Arnab Das, Bikas K. Chakrabarti, and Robin B. Stinchcombe. Quantum annealing in a kinetically constrained system. *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, 72(2), aug 2005. ISSN 15393755. doi: 10.1103/PhysRevE.72.026701.

[17] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *J. Math. Phys.*, 49(12):125210, 2008. ISSN 00222488. doi: 10.1063/1.2995837. URL https://api.semanticscholar.org/CorpusID:13992889#id-name=S2CID.

[18] Sergei V. Isakov, Guglielmo Mazzola, Vadim N. Smelyanskiy, Zhang Jiang, Sergio Boixo, Hartmut Neven, and Matthias Troyer. Understanding quantum tunneling through quantum Monte Carlo simulations. *Phys. Rev. Lett.*, 117(18), 2016. ISSN 10797114. doi: 10.1103/PhysRevLett.117.180402.

[19] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution. 2000. URL http://arxiv.org/abs/quant-ph/0001106.

[20] M. Born and V. Fock. Beweis des Adiabatensatzes. *Zeitschrift für Phys.*, 51(3-4):165–180, mar 1928. ISSN 14346001. doi: 10.1007/BF01343193. URL https://ui.adsabs.harvard.edu/abs/1928ZPhy...51..165B/abstract.

[21] Tosio Kato. On the Adiabatic Theorem of Quantum Mechanics. *J. Phys. Soc. Japan*, 5(6):435–439, dec 1950. ISSN 13474073. doi: 10.1143/JPSJ.5.435. URL https://journals.jps.jp/doi/abs/10.1143/JPSJ.5.435.

[22] Cohen-Tannoudji. Quantum Mechanics Vol 1, 2006. URL https://www.zuj.edu.jo/download/quantum-mechanics-vol-1-cohen-tannoudji-pdf/.

[23] Jose Antonio Alvarez Ocete. Interactive Avoided crossing using GeoGebra. URL https://www.geogebra.org/graphing/svkrnu3s.

[24] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Phys.*, 31 (1):253–258, 1925. ISSN 00443328. doi: 10.1007/BF02980577. URL https://link.springer.com/article/10.1007%2FBF02980577.

[25] Fred Glover, Gary Kochenberger, and Yu Du. Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4or*, 17(4):335–371, 2019. ISSN 16142411. doi: 10.1007/s10288-019-00424-y. URL https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects.

[26] Andrew Lucas. Ising formulations of many NP problems. *Front. Phys.*, 2:1–14, feb 2014. ISSN 2296424X. doi: 10.3389/fphy.2014.00005. URL www.frontiersin.org.

[27] Gary Kochenberger, Jin Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: A survey. *J. Comb. Optim.*, 28(1):58–81, 2014. ISSN 15732886. doi: 10.1007/s10878-014-9734-0.

[28] David G. Hull. Constrained Minimization: Inequality Constraints. pages 59–75, 2003. doi: 10.1007/978-1-4757-4180-3_4. URL https://link.springer.com/chapter/10.1007/978-1-4757-4180-3_4.

[29] Ronald V. Book. Book Review: Computers and intractability: A guide to the theory of NP-completeness. *Bull. Am. Math. Soc.*, 3(2):898–905, 1980. ISSN 0273-0979. doi: 10.1090/s0273-0979-1980-14848-x. URL https://archive.org/details/computersintract0000gare.

[30] Gary A. Kochenberger, Fred Glover, Bahram Alidaee, and Cesar Rego. An unconstrained quadratic binary programming approach to the vertex coloring problem. *Ann. Oper. Res.*, 139(1):229–241, 2005. ISSN 02545330. doi: 10.1007/s10479-005-3449-7.

[31] Aritra Sarkar, Zaid Al-Ars, and Koen Bertels. QuASeR: Quantum Accelerated de novo DNA sequence reconstruction. *PLoS One*, 16(4 April):1–24, 2021. ISSN 19326203. doi: 10.1371/journal.pone.0249850.

[32] D-Wave Computer Systems Comparison - Wikipedia. URL https://en.wikipedia.org/wiki/D-Wave_Systems#Comparison_of_D-Wave_systems.

[33] Welcome to D-Wave — D-Wave System Documentation documentation, . URL https://docs.dwavesys.com/docs/latest/c_gs_1.html.

[34] What is Quantum Annealing? — D-Wave System Documentation documentation, 2021. URL https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[35] D-Wave QPU Architecture: Topologies — D-Wave System Documentation documentation, . URL https://docs.dwavesys.com/docs/latest/c_gs_4.html#.

[36] Constraints Example: Minor-Embedding — D-Wave System Documentation documentation, . URL https://docs.dwavesys.com/docs/latest/c_gs_7.html.

[37] Stephanie Clare Roth. What is genomic medicine? *J. Med. Libr. Assoc.*, 107(3): 442–448, jul 2019. ISSN 15589439. doi: 10.5195/jmla.2019.604.

[38] B Alberts, A Johnson, J Lewis, M Raff, K Roberts, and P And Walter. Molecular Biology of the Cell - NCBI Bookshelf, 2007. URL http://www.ncbi.nlm.nih.gov/books/NBK21054.

[39] Barton E. Slatko, Jan Kieleczawa, Jingyue Ju, Andrew F. Gardner, Cynthia L. Hendrickson, and Frederick M. Ausubel. "First generation.ªutomated DNA sequencing technology. *Curr. Protoc. Mol. Biol.*, (SUPPL.96), oct 2011. ISSN 19343639. doi: 10.1002/0471142727.mb0702s96.

[40] Susan P. Holmes and Dan Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Technical Report 447, 1999.

[41] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, mar 1970. ISSN 00222836. doi: 10.1016/0022-2836(70)90057-4.

[42] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. Technical Report 1, 1981.

[43] A. S. Boev, A. S. Rakitko, S. R. Usmanov, A. N. Kobzeva, I. V. Popov, V. V. Ilinsky, E. O. Kiktenko, and A. K. Fedorov. Genome assembly using quantum and quantum-inspired annealing. *arXiv*, pages 1–8, 2020. ISSN 20452322. doi: 10.1038/s41598-021-88321-5. URL http://arxiv.org/abs/2004.06719%0Ahttp://dx.doi.org/10.1038/s41598-021-88321-5.

[44] D-Wave Ocean Software Documentation, . URL https://docs.ocean.dwavesys.com/en/stable/index.html.

[45] D-Wave Leap, . URL https://cloud.dwavesys.com/leap.

[46] Github repository containing my code for this project. URL https://github.com/Ocete/TFG.

[47] dwavesystems/dimod: A shared API for QUBO/Ising samplers. URL https://github.com/dwavesystems/dimod.

[48] Samplers — Ocean Documentation 3.4.0 documentation, . URL https://docs.ocean.dwavesys.com/en/latest/docs_dimod/reference/sampler_composites/samplers.html#module-dimod.reference.samplers.simulated_annealing.

[49] Embedding — dwave-system 1.6.0 documentation, . URL https://docs.ocean.dwavesys.com/projects/system/en/latest/reference/embedding.html.