AALBORG UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Quantum Data Compression

Joana I.L. Miguéns

Supervisor: Yasser Omar
Co-Supervisor: Luca Aceto

November 9, 2003

ABSTRACT

Quantum Information Theory is about studying information encoded in quantum systems and exploring their unique properties. The topic of our report is Quantum Data Compression, a central problem to storage and transmission of data. Our report presents an overview of Classical Coding Theory and the main concepts of Quantum Coding Theory. Moreover, we introduce a particular quantum noiseless and lossless data compression scheme. Finally, we present three methods to improve this scheme. Our first method, *Brute Force (BF)*, presents only significant advantages for small amounts of data, but has otherwise a very high complexity time. This is further improved with our scheme *Improved BF*, where the complexity time is decreased by taking profit of the advantages of quantum algorithms. Another alternative is our *Adapted Algorithm*, based on a different way of ordering probabilities and which offers the best complexity time, but contrary to the previous two methods does not guaranty that an optimal solution will be found.

Preface

This report is submitted as the Master thesis in Software Systems Engineering in the Department of Computer Science at Aalborg University, Denmark, August 2003. Its aim is to study quantum data compression schemes.

Joana I.L. Miguéns

Contents

1 Introduction

Quantum Information Science is about studying information encoded in quantum systems and exploring their unique properties. The consequences that the processing of information is always performed by physical means is all but trivial. When quantum effects become important, for example at the level of atoms and photons, the existing classical theory of computation become fundamentally inadequate. Entirely new modes of computation become possible. Quantum Information is a new paradigm based on the laws of Quantum Mechanics. It is a new science expanding very fast and represents a revolution in our way of understanding and using information, resulting in important transformations in the Information Society we live in.

Quantum Information Theory, the quantum version of Classical Information Theory, has been developed due to the rising interest in both the theory of Quantum Computation and the realization of quantum computers. It brings together ideas from Classical Information Theory, Physics, Computer Science, Mathematics and Communication Theory.

The typical questions driving Quantum Information Theory Science are: what is it that separates the quantum and the classical world from the Information Theory point of view? What resources, unavailable in a classical world, are being utilized in a Quantum Computation? The existing answers to these questions are not always completely clear. It is our hope that the fog may lift in the years to come, and we will obtain a clear appreciation of the possibilities and limitations of Quantum Information Science.

Quantum Information Science started about 10 years ago and has a very promising future: it was already shown that in theory quantum computers are more powerful than classical ones in various specialized problems. A remarkable step of Quantum Information Theory is Shor's probabilistic polynomial-time algorithm for factoring [1]. Nowadays we believe there is no efficient classical algorithm for solving the factorization problem. This is of great importance in classical cryptography [2]: the reliability of the famous public-key cryptosystem RSA is based on that assumption. With Shor's algorithm one could easily break classical security systems. This could have huge implications on the economy, for the military and on any security system. On the other hand, using Quantum Cryptography protocols, in particularly quantum key distribution, is theoretically allowed to offer completely secure systems and, in fact, quantum cryptographic systems have been recently implemented [3].

Another case where a quantum algorithm is faster over classical ones is the search for a particular element in a long unsorted list. Suppose we want to find an item of such a list of N elements. A classical algorithm for this problem must examine all entries if the item we are searching for is not in the list, and has

therefore a worst-case complexity that is linear in N. However, thanks to the possibility of using superpositions of quantum states, quantum systems can be simultaneously in more than one state and these allow parallelism. Grover's algorithm was developed to perform a quantum search exploiting this property. Its worst-case complexity is a linear function of $\sqrt{N}$.

Perhaps the most intriguing product of Quantum Information Theory is the concept of quantum information itself, realizing that we can treat quantum states as if they were information. A particularly interesting and relevant topic of Quantum Information Theory is Quantum Data Compression, which is the topic of our report. The problem of compressing is central to storage and transmission of data. In addition to its evident utility for practical communication issues, the concept of information compression provides a bridge between the Theory of Information and Physics - it determines what are the minimal physical requirements to store and transmit information. Classical compression schemes have many applications in different fields of science. For example, the similarities between the theory of optimal investment in the stock market and Information Theory are striking. Cryptographic protocols is another important application, representing an important part of Communication Theory.

The essencial point of this report is to address the aim of Classical and Quantum Coding Theory. We introduce the main classical compression schemes [4] and the basic concepts of Quantum Information with a view to research Quantum Data Compression schemes. Schumacher [5] played a fundamental role in Quantum Data Compression when he introduced a theorem stating how much compression one could possibly achieve. Moreover a few quantum data compression agorithm have been proposed, a variable-length code [6], the quantum Lempel-Ziv code [7] and another universal quantum information compression [8]. We extend in three different ways one of this quantum compression schemes, the variable-length quantum code.

This report is organized as follows. In section 2.1, we present the essentials of classical compression information, introduce notation and the main theorems of Classical Noiseless Coding (for channel where no errors occur). Moreover, in section 2.2, four of the most commonly used classical compression schemes are described. *Huffman's Code*, described in section 2.2.1, achieves optimal compression but it has a high complexity time. In section 2.2.2, *Lempel-Ziv Code* is presented. This code requires no knowledge of the probabilities of the source symbols to compress, it is called a universal compression scheme. *Arithmetic Code* (in section 2.2.4) does not compress as much as the Huffman's code, but it is quite reasonable and the reusability of some parts of the code for different sources makes the complexity time become low. Finally, *Enumerative Code* in section 2.2.2, is used for sources in which the data sequence are equally likely.

In section 3 an introduction to Quantum Information is presented from the

computer science point of view. Firstly, in section 3.1, we introduce the basic concepts of Quantum Mechanics and Quantum Information. Then, section 3.2 has a description of one of the most important features for the speedup of Quantum Computation over Classical Computation, quantum parallelism, as well as one of its applications, Grover's algorithm. Another important and puzzling concept of Quantum Information is *entanglement*, introduced

in section 3.3. It has no analog concept in the classical world and plays an important role in Quantum Information Theory. One of its aplications, dense coding, is presented in section 3.3.1.

Finally, section 4 presents Quantum Data Compression. We introduce, in section 4.1, an overview of the main properties of quantum information itself and discuss the objectives and achievements of the quantum encoding schemes developed so far [8, 7, 9, 10, 11, 12, 13]. Moreover in section 4.2.1 we describe a variable-length quantum encoding scheme [6] which we improve in three different ways. In section 4.2.2 we present the *Brute Force* algorithm, it finds always the optimal solution, but it has exponential complexity time. Therefore, in section 4.2.3, *Improved BF* algorithm describes an improvement of Brute Force complexity time by using the Grover's algorithm and its extension. Finally, in section 4.2.4, a completely different approach is done to compress the quantum data. It reaches a quadratic complexity time, but it is not proven to find always the optimal solution.

2 Classical Coding Theory

Information Theory studies the way information is processed, meaning by this, roughly, the study of data compression and data transmission. In this section we introduce the basic terms of Coding Theory, its aim is to compress data as much as possible to use the less physical resources one can hope to use. The main concepts of classical coding are presented in section 2.1. Moreover, in section 2.2, we introduce three of the most commonly used classical compression schemes: Huffman's Code 2.2.1, Lempel-Ziv Code 2.2.2, Arithmetic code 2.2.3 and Enumerative code 2.2.4.

2.1 Classical Noiseless Coding

Firstly in this section we present some basic terminology of Information Theory, mainly the most important definitions and theorems. Then we present four of the most commonly used classical noiseless codes: Huffman code, Enumerative code, Arithmetic code and Lempel-Ziv code.

We may compress data in two different ways, lossless and lossy compression. These are terms that describe whether or not, in a compression, all original data can be recovered when the file is uncompressed. With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. All of the information is completely restored. This is generally the technique of choice for text or spreadsheet files, where losing words or financial data could pose a problem. The Graphics Interchange File (GIF) is an image format used on the Web that provides lossless compression. On the other hand, lossy compression reduces a file by permanently eliminating certain information, especially redundant information. When the file is uncompressed, only a part of the original information is still there (although the user may not notice it). Lossy compression is generally used for video and sound, where a certain amount of information loss will not be detected by most users. The JPEG image file, commonly used for photographs and other complex still images on the Web, is an image that has lossy compression. Using JPEG compression, the creator can decide how much loss to introduce and make a trade-off between file size and image quality. In this report we only study lossless compression.

We study classical coding algorithms for noiseless channels. In such a channel, where no errors occur, the only worry is about how efficiently the coding is done, in a sense that will be precise in a moment. Conversely, noisy channel are the ones where occurs error. Before any definition we try to introduce some intuition about all this. One of our goals is to measure information. But, what does measure information mean? Imagine we flip a coin. How much information do we have to guess the result? What is the amount of uncertainty in this experiment?

What about if we drop a dice instead? It is intuitive that the uncertainty of the second experiment increases. Now consider a closed box with 4 white balls and 1 black ball. If we take one ball from the box, which one gives us more information? To pick a white or the black ball? A little thought on this question leads us to the conclusion. If we pick the black ball then we know which is the color of any ball inside the box. This would not have happened if we had picked a white ball.

So, there must be a way to quantify information. Assume that we have an experiment with k equally probable results. The uncertainty of this experiment is determined by the value of *k* - (let us call this measure of uncertainty $H(k)$. As we will see in a moment to be the entropy): so, for $k = 1$ the result is completely determined, $H(1) = 0$. It is clear that the uncertainty growths as *k* growths. Suppose now we have two independent experiments with *l* and *k* outcomes, respectively, equally likely. If we compound the two experiments, the result would have *kl* likely probable results, intuitively, and the uncertainty would be the sum of the uncertainty in both experiments, $H(kl) = H(k) + H(l)$. All this conditions lead us to define $H(k) = \log_b(k)$, for some b constant. In a similar way we define *quantity of information* to be the sum of $\frac{1}{x}\log_b(x)$, where x stands for the outcomes. The intrusion of logarithms might have been a surprise when pointed out, today it actually seems quite natural. Nevertheless, the explicit use of logarithms helped Claude Shannon, an American mathematician and electrical engineer, to make a big connection in the 1940s when he showed how the theory of heat -thermodynamics- was applicable to information transmission. In a famous and much cited article entitled "A mathematical theory of communication," he presented a unifying theory of the transmission and processing of information [14].

The concept of *entropy* is the measure of uncertainty of a random variable. This concept agrees with the intuitive notion of what a measure of information should be.

Definition 1 *Let X be a discrete random variable with range $S = \{x_1, ..., x_n\}$. If $P(X = x_i) = p(x_i)$, then the **entropy** of X is defined by:*

$$H(X) = \sum_{i=1}^{n} p(x_i) \log \frac{1}{p(x_i)}. \tag{1}$$

More detail and properties of entropy can be found, for example, in [4].

Now you may be asking yourself, but how do we relate entropy and data compression? Later in this section we establish the limit of data compression as the entropy value of the source. *Data compression* is also called source coding in a sense that our goal is to encode the information contained in a source. It can be achieved by assigning short description to the most frequent outcomes of the data source and necessarily longer descriptions to the less frequent outcomes. Let us present a formal description of the term source.

Definition 2 *A **source** is an ordered pair $\mathcal{S}$=(S,P), where S=$\{x_1, ..., x_n\}$ is a finite set, known as a **source alphabet**, and P is a probability distribution (denoted by $p(x_i)$) on S.*

Furthermore, we assume that the different uses of the source are *independent* and *identically distributed* (*i.i.d.*, formal definition in appendix A). In the real world information source usually don't behave this way. For example consider an English text, the sequence of letter don't occur independently. Many times the letter "t" is followed by letter "h". Therefore, there exits a strong correlation between this two letters. However, most of the applications consider, even sources like an English text, i.i.d. sources, working pretty well in practice.

Definition 3 *If successive symbols are i.i.d., the information source is a **zero-memory source** (or **memoryless source**).*

Let $\mathcal{A} = \{a_1, ..., a_n\}$ be a finite set, called **alphabet**. A **word** w over $\mathcal{A}$ is any finite sequence of elements of $\mathcal{A}$. We define the **length** of a word, denoted as *len*, as the number of alphabet symbols in the word. We write $\mathcal{A}^*$ for the set of all words over $\mathcal{A}$.

Definition 4 *Let $\mathcal{A} = \{a_1, ..., a_r\}$ be a finite set, called the **code alphabet**. An **r-ary** code is a nonempty subset C of the set $\mathcal{A}^*$.*

Usually the elements of the code are denominated **code words**. The most used code alphabet is $C = \{0, 1\}$, a binary code. From a source $\mathcal{S}$ we intent to construct codewords to assign to the source symbols, the procedure is called *encoding scheme*, defined below:

Definition 5 *Consider the source $\mathcal{S}$=(S,P). An **encoding scheme** for the source $\mathcal{S}$ is an ordered pair (C, f), where C is a code and $f : S \to C$ is an injective function.*

This way we assign a code word to each source symbol.

It is clear that the average codeword length of an encoding scheme is not affected by the nature of the source symbols themselves. Therefore, we may think directly on the probabilities assigned to the codewords, thus obtaining an encoding scheme *C*=(c_1,...,c_n) for the probability distribution *P*=(p_1, ...,p_n). We define the **codewords length average** as:

$$L(C) = \sum_{i=1}^{n} p_i len(c_i). \tag{2}$$

where *len(x)* denotes the length of *x*.

Below we have an example with these concepts.

Example 1 *Consider a source:*

$$S = \{N, S, E, W\} \textit{ with probabilities } P = (0,5; 0,05; 0,05; 0,4).$$

The code alphabet is $\{0, 1\}$*. Consider an encoding scheme* $C = \{0, 10, 110, 1110\}$ *and* f *defined as* $f(N) = 0$*,* $f(S) = 10$*,* $f(E) = 110$ *and* $f(W) = 1110$*. Therefore the length average is:*

$$L(C) = 0,5 \times 1 + 0,05 \times 3 + 0,05 \times 4 + 0,4 \times 2 = 1,65 \tag{3}$$

The problem we deal with in data compression is to reduce as much as we can the length average of the codewords. We prove later in this section, that for a sufficiently long sequence of data, there is an approach between the compression limit and the entropy as close as we want. The example above has $H(S) = 1,461$, therefore we could probably compress the data more, depending on the amount of data we want to compress. Our goal now is to determine the *minimum* average codeword length among all the "good" encoding schemes (in a sense we will make precise soon), as well as the method for constructing such encoding scheme. In general we may define two classes of encoding schemes. If all the codewords in a code C have the same length, we say that C is a **fixed length code**. If C contains codewords of different lengths, we say that C is a **variable length code**. Mostly the variable length encoding schemes are more efficient. However, those schemes have a potential problem. Imagine we have a source $S = \{a, b, c\}$ and a variable encoding scheme with $C = \{0, 01, 001\}$ and $f(a) = 0$, $f(b) = 01$ and $f(c) = 001$. This encoding scheme is not *uniquely decipherable*, is a sense that the code word 001 could be decoded as *ab* or as *c*. The difficulty here can be traced to the fact that a word of code alphabet symbols may represent more than one word of codewords. This leads to the following definition.

Definition 6 *A code C is* ***uniquely decipherable*** *if whenever* $c_1, ..., c_k$ *and* $d_1, ..., d_j$ *are codewords in C and*

$$c_1...c_k = d_1...d_j, \tag{4}$$

then $k = j$ *and* $c_i = d_i$ *for all* $i = 1, ..., k$*.*

Clearly this property is extremely desirable. There are methods to show that a particular code is uniquely decipherable. One of the difficulties with uniquely decipherable codes is that, even though a code may have this property, it may be necessary to wait until the entire message has been received before we can begin to decode. This leads us to the following definition:

Definition 7 *A code is said to be* ***instantaneous*** *if each codeword in any word of codewords can be decoded (reading from left to right) as soon as it is received.*

Obviously an instantaneous word is also uniquely decipherable, but a uniquely decipherable code does not need to be instantaneous, as the following example shows.

Example 2 *Considering the same source sample of the previous example we state* $C = \{0, 01, 011, 0111\}$*, such that* $f(N) = 0$*,* $f(S) = 01$*,* $f(E) = 011$ *and* $f(O) = 0111$*. It is not difficult to check that this is an uniquely decipherable code. Suppose we send the codeword* 0111*. When we receive the* 0 *we don't know whether it is suppose to decode like N or if it is the first part of other codeword. The same happens while we receive the first and the following* 1*'s. So, the message only can be decoded when completely received.*

The following method is sufficient to check if a code is instantaneous. It also gives a good idea to construct instantaneous words.

Definition 8 *A code has the* ***prefix property*** *if no codeword is a prefix of any other codeword, that is, if whenever* $c = x_1x_2...x_n$ *is a codeword, then* $x_1x_2...x_k$ *is not a codeword for* $1 \leq k < n$*.*

The theorem below states the importance of the prefix property, [4].

Theorem 1 *A code C is instantaneous if and only if it has the prefix property.*

We desire to construct instantaneous codes of minimum expected length to describe a given source. It is clear that we can not assign short codewords to all source symbols and still be prefix free. In 1949 L. G. Kraft published a remarkable theorem stating a criterion to determine if there is an instantaneous code with given codeword lengths.

Theorem 2 ***(Kraft's Theorem)*** *There exists an* r*-ary instantaneous code C with codewords of length* $l_1, l_2, ..., l_m$ *if and only if the following inequality (Kraft's inequality) holds*

$$\sum_{i=1}^{m} \frac{1}{r^{l_i}} \leq 1 \tag{5}$$

The proof can be found for example in [4]. Kraft's theorem states that if the lengths $l_1, l_2, ..., l_m$ satisfy the Kraft's inequality, then there must be an instantaneous code with these code lengths. It does not states that any code whose lengths satisfy Kraft's inequality must be instantaneous. So far we know that any instantaneous code is uniquely decipherable, but the opposite is not true. Therefore Kraft's inequality is also sufficient for the existence of a uniquely decipherable code, but in 1954 McMillan stated that Kraft's inequality also is a necessary condition for the existence of uniquely decipherable codes.

Theorem 3 ***(McMillan's Theorem)*** *The codeword lengths of any uniquely decodable code, $l_1, l_2, ..., l_m$, must satisfy the Kraft inequality*

$$\sum_{i=1}^{m} \frac{1}{r^{l_i}} \leq 1. \tag{6}$$

Conversely, given a set of codeword length that satisfy this inequality, it is possible to construct a uniquely decodable code with these codeword lengths.

The proof of McMillan's theorem may be found in [4]. Kraft's and McMillan's theorems are two basic and relevant theorems with important practical consequences. The lemmas below (stated in [4]) provide two of the these consequences.

Lemma 1 *If exists an uniquely decodable code with codeword lengths $l_1, l_2, ..., l_m$, then there must exist an instantaneous code with the same code word lengths.*

Lemma 2 *The minimum average codewords length among all uniquely decipherable codes for a given source, S, is equal to the minimum average codewords length among all instantaneous encoding scheme for S.*

Hence, in seeking to minimize the average codeword length over all uniquely decodable encoding schemes, we may restrict ourselves to instantaneous codes. Since our goal is to reduce as much as we can the length average of the codewords we must search for a balance of decoding capacity and practical coding efficiency. In other words, we seek instantaneous codes with the minimum of length average, called *optimal codes*.

Definition 9 *An **optimal r-ary code** for a source $\mathcal{S} = (S, P)$; with $S = \{s_1, ..., s_m\}$ and $P = \{p_1, ..., p_m\}$, is an r-ary instantaneous encoding scheme $C = \{w_1, ..., w_m\}$ with minimum average length $L(C) = \sum_{i=1}^{m} p_i|w_i|$ among all possible instantaneous codes for source $\mathcal{S}$.*

The concepts we introduced so far are sufficient to discuss the main results in noiseless coding. So far we know entropy, $H(\mathcal{S})$, of a source $\mathcal{S}$ is the amount of information contained in the source. Further, since an instantaneous encoding scheme for $\mathcal{S}$ captures the information in the source, it is not unreasonable to expect that the average codeword length of such a code must be at least as large as the entropy $H(\mathcal{S})$. In fact, this is what the Noiseless Coding theorem says, proved by Claude Shannon in 1948 [14]. The generalized version of this theorem, that will be discussed in a moment, also says that by clever encoding, we can make the average codeword length as close to the entropy as desired.

Theorem 4 ***(The Noiseless Coding Theorem)*** *For any probability distribution P of a source S, we have,*

$$H_r(S) \leq L(C) < H_r(S) + 1 \tag{7}$$

where C is an optimal code with probability distribution P and H_r is the entropy of an r-ary code, $H_r(X) = \sum_{i=1}^{n} p(x_i) \log_r \frac{1}{p(x_i)}$.

In the preceding theorem, there is an overhead that is at most one unit. However we may want to be more ambitious with our compression and try to compress as much as we can. We can reduce the overhead per symbol by spreading it out over many symbols. For this we consider a system in which we send a sequence of n symbols from the source. This idea is called extensions of a source.

Definition 10 *Let $\mathcal{S} = (S, P)$ be a source. The **n-th extension** of $\mathcal{S}$ is $S^n = (S^n, P^n)$, where S^n is the set of all words of length n over S, and P^n is the probability distribution defined for $x = x_1...x_n$ by*

$$P^n(x) = P(x_1)...P(x_n). \tag{8}$$

The symbols are assumed to be drawn from an *i.i.d.* (see appendix A) according to $p(x)$.

We can consider these n symbols to be a supersymbol from the alphabet S^n. Now we are able to introduced a generalization of Shannon Theorem. We can verify that $H(S^2) = 2H(S)$. Applying Shannon Theorem to the source S^2 we know there is an optimal code C^2 such that,

$$2H(S) = H(S^2) \leq L(C^2) \leq H(S^2) + 1 = 2H(S) + 1, \tag{9}$$

each symbol of S^2 contains 2 symbols of the original source, S. So, the average symbol (of S) lengths will be $\frac{L(C^2)}{2}$. The previous estimation leads to

$$H(S) \leq \frac{L(C^2)}{2} \leq H(S) + \frac{1}{2}, \tag{10}$$

Shannon Theorem can be generalized thus:

Theorem 5 *Consider S^n an extension of a source S. In terms of probability distributions,*

$$H_r(P^n) = nH_r(P) \tag{11}$$

Shannon Theorem and the previous theorem provide the following theorem,

Theorem 6 *Consider an **n-th extension** of a source $\mathcal{S} = (S, P)$, $S^n = (S^n, P^n)$. Then*

$$H_r(S) \leq \frac{L(C^n)}{n} < H_r(S) + \frac{1}{n}, \tag{12}$$

where C is an optimal code with probability distribution P and C^n its n-extension.

This theorems are stated in [4]. The last theorem says that to encode a sufficient long extension of $\mathcal{S}$, then we may make the minimum average codeword length per source symbol of $\mathcal{S}$ as close to the entropy $H_r(P)$ as desired. We now give an example of source extension.

Example 3 *Let $S = (a, b)$ and $P = (\frac{1}{4}, \frac{3}{4})$. Intuitively we expect that the encoding scheme (C,f):*

$$f(a) = 0, f(b) = 1 \tag{13}$$

couldn't be more compressed, in a sense that the minimal length of a word is one and both the code words have length one. So the average codeword length is $\frac{1}{4} \times 1 + \frac{3}{4} \times 1 = 1$. Constructing the extension S^2 we get a probability distribution, P^2:

$$f(aa) = 010, f(ab) = 011, f(ba) = 00 \text{ and } f(bb) = 1 \tag{14}$$

with average codeword length $\frac{1}{16} \times 3 + \frac{3}{16} \times 3 + \frac{3}{16} \times 2 + \frac{9}{16} \times 1 = \frac{27}{16}$. Per source symbol we have an average codeword length equal to $\frac{27}{32}$. Which is less than 1 and verify,

$$H(S^n) = \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3} = \frac{1}{2} + \frac{3}{4} 0.12 = 0.59 \leq \frac{27}{32} \leq 0.59 + \frac{1}{2}. \tag{15}$$

Now we have define the desired properties of a compression scheme we introduce in the next section four of the most important encoding schemes: the Huffman's code, the Lempel-Ziv code, the Arithmetic code and the enumerative code.

2.2 Classical Encoding Schemes

The idea of encoding a source is to obtain optimal codes and, in some applications, it is important to consider the time we take to encode and decode. There are many compression algorithms, different from each other in tiny details. The most commonly used are the Huffman's code [15], Lempel-Ziv code [16], Arithmetic code [17, 18] and Enumerative code [19]. We introduce these codes, discuss their compression limits and their applications.

2.2.1 Huffman's Code

The best known noiseless encoding method is the *Huffman Encoding*, published in 1952 [15]. This is an optimal encoding scheme, statement proven along the section. Huffman's encoding scheme gives an algorithm for minimizing the length average (*Min L*(C)). Without loss of generality and considering our goal, we present Huffman's algorithm for binary codewords. It operates in the following way. Let $\mathcal{S}$=(S,P) be a source, such that $S = (s_1, ..., s_m)$ with probabilities $P = (p_1, ..., p_m)$. Huffman's algorithm operates recursively as follows. Let us start with an informal description of Huffman's code. The description may get easier to understand if you follow at the same time example 4. At first we reorder, if necessary, the symbols of *S* in such way that the respective probabilities end up in a non-increasing order.

$$p_n \leq ... \leq p_2 \leq p_1 \tag{16}$$

Now we obtain a reduced source, which we call $S^{'}$, appending the two less probable symbols s_m and s_{m-1} in a single symbol $s' = s_{m-1} \vee s_m$ and attributing to it the probability $p' = p_{m-1} + p_m$. In case there is more than one symbol with the minimal probability, just pick two of them randomly. Following this idea we have constructed a source *S'* with $m - 1$ elements, $s_1, ..., s_{m-2}, s'$ with the respective probabilities, $p_1, ..., p_{m-2}, p'$.

Given a binary code C' for S' we can construct a binary code C for S as follows:

$$\begin{aligned} &For\ i = 1, ...m-2 : c_i = C'(s_i) => C(s_i) = c_i \\ &c' = C'(s') => C(s_{m-1}) = c'0 \text{ and } C(s_m) = c'1 \end{aligned} \tag{17}$$

The preceding explained how to construct a binary code C for a source S with m symbols from a binary code C' for a reduced source S' with $m - 1$ symbols. We could follow the some idea to construct C' from a binary code C'' of a source S'' with $m - 2$ symbols. Applying this several times we obtain the sequence $S, S', S'', ..., S^{m-2}$, each the reduced form from the previous one, with respectively $m, m - 1, m - 2, ..., 2$, elements:

$$S \to S' \to S'' \to ... \to S^{m-2} \tag{18}$$

S^{m-2} has two symbols, $s_2 \vee ... \vee s_{m-1}$ and s_1 with probabilities $p_2 + ... + p_{m-1}$ and p_1, respectively; we encode it with the symbols 0 and 1, and thus $C^{(m-2)} = \{0, 1\}$. Following the process just described we construct $C^{(m-3)}$ for the source with 3 symbols $S^{(m-3)}$. Applying the same process $m - 2$ times we obtain a sequence of codes

$$C^{(m-2)} \to ... \to C' \to C.$$

A formal way to describe Huffman's algorithm is:

Algorithm 1 *Consider the following algorithm $\mathcal{H}$ for producing binary encoding schemes C for probability distribution P.*

1. *If P=(p_1,...,p_n), where $n \leq 2$, then let $C = (0)$ if $n = 1$ and $C = (0, 1)$ if $n = 2$.*
2. *If $P = (p_1, ..., p_n)$, where $n > 2$, then*
   - *(a) Reorder P if necessary so that $p_1 \geq p_2 \geq ... \geq p_n$.*
   - *(b) Let $Q = (p_1, ..., p_{n-2}, q)$, where $q = p_{n-1} + p_n$.*
   - *(c) Perform the algorithm $\mathcal{H}$ on Q, obtaining an encoding scheme: $D = (c_1, ..., c_{n-2}, d)$*
   - *(d) Let $C = (c_1, ..., c_{n-2}, d0, d1)$*

We present an example of Huffman code.

Example 4 *Suppose we have $\mathcal{S} = (\{a, b, c, d\}, \{0.54, 0.31, 0.09, 0.06\})$ At first we write down the probability columns. For example, in the second probability column we summed the smallest two probabilities of the first probability column, the same construction scheme is used to obtain the other probability columns. Then we complete the table appending the codewords from the right to the left.*

Prob.	*Code*	*Prob.*	*Code*	*Prob.*	*Code*
0.54	*0*	*0.54*	*0*	*0.54*	*0*
0.31	*10*	*0.31*	*10*	*0.46*	*1*
0.09	*110*	*0.15*	*11*		
0.06	*111*				

This way we obtain $C = \{0, 10, 110, 111\}$.

Performance

The code constructed by Huffman's algorithm is optimal (proof in [4]), which implies that it is an instantaneous code with minimum average length code. Even so we care about its complexity. Complexity Theory is concerned with the inherent cost required to solve information processing problems. Unfortunately, Huffman's code, cannot be used in many application, thanks to the construction

time it requires. Storage space is also an important issue and Huffman's algorithm takes a considerable amount of construction space. This makes it only possible to use it in applications which require a small source alphabet.

To cover the gap left by Huffman's codes for large values of *N* source symbols we now present other data compression codes, namely the Lempel-Ziv Code, Arithmetic Code and Enumerative Code.

2.2.2 Lempel-Ziv Code

Huffman's algorithm may take a long construction time because it requires knowledge of the probabilities of the source symbols. When there is no knowledge of the source characteristics, and if statistical tests are either impossible or unreliable, the problem of data compression becomes considerably more complicated. In order to overcome these difficulties one must resort to universal coding schemes whereby the coding process is interlaced with a learning process for the varying source characteristics. Such coding schemes inevitably require a large working memory space and generally employ performance criteria that are appropriate for a wide variety of sources.

Here we describe a universal coding scheme which can be applied to any discrete source. Lempel-Ziv Codes are examples of *dictionary codes*. A dictionary code first partitions a data sequence into variable-length blocks (this procedure is called *parsing*). Then, each phrase in the parsing is represented by means of a pointer to that phrase in a dictionary of phrases constructed from previously processed data. The phrase dictionary changes dynamically as the data sequence is processed from left to right. A binary codeword is then assigned to the data sequence by encoding the sequence of dictionary pointers in some simple way. The most popular dictionary codes are the Lempel-Ziv codes. There are many versions of the Lempel-Ziv codes. The one we discuss here is called LZ78 [16]. Two widely-used compression algorithms on Unix systems are Compress and Gzip; Compress is based on LZ78 and Gzip is based on another popular Lempel-Ziv code, not discussed here, called LZ77 [20].

In the rest of this section, we discuss the parsing technique, the pointer formation technique, and the pointer encoding technique employed in Lempel-Ziv coding.

Lempel-Ziv Parsing. Let $(x_1, x_2, ..., x_n)$ be the data sequence to be compressed. Partitioning of this sequence into variable-length blocks via *Lempel-Ziv parsing* takes place as follows. The first variable-length blocks arising from the Lempel-Ziv parsing of $(x_1, x_2, ..., x_n)$ is the single sample x_1. The second block in the parsing is the shortest nonempty prefix of $(x_2, ..., x_n)$ which is not equal to x_1. Suppose this second block is $(x_2, ..., x_j)$. Then the third block in Lempel-Ziv parsing will be the shortest nonempty prefix of $(x_{j+1}, x_{j+2}, ..., x_n)$ which is not equal

to either x_1 or $(x_2, ..., x_j)$. In general, suppose the Lempel-Ziv parsing procedure has produced the first *K* variable-length blocks $B_1, B_2, ..., B_K$ in the parsing, and $x^{(k)}$ is that part left of $(x_1, ..., x_n)$ after $B_1, B_2, ..., B_K$ have been removed. Then the next block B_{k+1} in the parsing is the shortest nonempty prefix of $x^{(k)}$ which is not equal to any of the preceding blocks $B_1, B_2, ..., B_K$. (If there is no such prefix of x^k, then $B_{k+1} = x^{(k)}$ and the Lempel-Ziv parsing procedure terminates.)

By construction, the sequence of variable-length blocks $B_1, B_2, ..., B_t$ produced by the Lempel-Ziv parsing of $(x_1, x_2, ..., x_n)$ are distinct, except that the last block B_t could be equal to one of the preceding ones. The following example illustrates Lempel-Ziv parsing.

Example 5 *The Lempel-Ziv parsing of the data sequence*

$$(1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1) \tag{19}$$

is

B_1	*(1)*
B_2	*(1,0)*
B_3	*(1,1)*
B_4	*(0)*
B_5	*(0,0)*
B_6	*(1,1,0)*
B_7	*(1)*

Pointer Formation. We suppose that the alphabet from which the data sequence $(x_1, x_2, ..., x_n)$ is formed is $A = \{0, 1, ..., k-1\}$, where *k* is a positive integer. After obtaining the Lempel-Ziv parsing $B_1, B_2, ..., B_t$ of $(x_1, x_2, ..., x_n)$, the next step is to represent each block in the parsing as a pair of integers. The first block in the parsing, B_1, consists of single symbol. It is represented as the pair $(0, B_1)$. More generally, any block B_j of length one is represented as the pair $(0, B_j)$. If the block B_j is of length greater than one, then it is represented as the pair (i, s), where *s* is the last symbol on B_j and B_i is the unique previous block in the parsing which coincides with the block obtained by removing *s* from the end of B_j.

Example 6 *The sequence of pairs corresponding to the parsing of the previous example is*

$$(0, 1), (1, 0), (1, 1), (0, 0), (4, 0), (3, 0), (0, 1) \tag{20}$$

For example, $(4, 0)$ corresponds to the block $(0, 0)$ in the parsing. Since the last symbol of $(0, 0)$ is 0, the second component of the pair $(4, 0)$ is 0. The 4 in

the first entry refers to the fact that $B_4 = (0)$ is the preceding block in the parsing which is equal to what we get by deleting the last symbol of $(0, 0)$.

For our next step, we replace each pair (i, s) by the integer $ki + s$. Thus, the sequence of pairs (20) becomes the sequence of integers

$$\begin{aligned} I_1 &= 2 \times 0 + 1 = 1 \\ I_2 &= 2 \times 1 + 0 = 2 \\ I_3 &= 2 \times 1 + 1 = 3 \\ I_4 &= 2 \times 0 + 0 = 0 \\ I_5 &= 2 \times 4 + 0 = 8 \\ I_6 &= 2 \times 3 + 0 = 6 \\ I_7 &= 2 \times 0 + 1 = 1 \end{aligned} \tag{21}$$

Encoding the pointers. Let $I_1, I_2, ..., I_t$ denote the integer pointers corresponding to the blocks $B_1, B_2, ..., B_t$ in the Lempel-Ziv parsing of the data sequence $(x_1, x_2, ..., x_n)$. To finish our description of the Lempel-Ziv encoder, we discuss how the integer pointers $I_1, I_2, ..., I_t$ are converted into a stream of bits. Each integer I_j is expanded to base two and these binary expansions are "padded" with zeroes on the left so that the overall length of the string to be assigned to I_j is $\lceil \log_2(kj) \rceil$. The reason why these many bits is necessary is seen by examining the largest that I_j can possibly be. Let (i, s) be the pair associated with I_j. Then the biggest that *i* can be is *j-1* and the biggest than *s* can be is *k-1*. Thus the biggest that I_j can be is $k(j-1) + k - 1 = kj - 1$, and the number of bits in the binary expansion of $kj - 1$ is $\lceil \log_2(kj) \rceil$. Let W_j be the string of bits of length $\lceil \log_2(kj) \rceil$ assigned to I_j as described above. Then, the Lempel-Ziv encoder output is obtained by concatenating together the strings $W_1, W_2, ..., W_t$.

To illustrate, suppose the data sequence $(x_1, x_2, ..., x_n)$ is binary (i.e., k=2), and has seven blocks $B_1, B_2, ..., B_7$ in its Lempel-Ziv parsing. These blocks are assigned, respectively, string of code bits $W_1, W_2, ..., W_7$ of lengths $\lceil \log_2(2) \rceil = 1$, $\lceil \log_2(4) \rceil = 2$, $\lceil \log_2(6) \rceil = 3$, $\lceil \log_2(8) \rceil = 3$, $\lceil \log_2(10) \rceil = 4$, $\lceil \log_2(12) \rceil = 4$, $\lceil \log_2(14) \rceil = 4$. Therefore, any binary data sequence with seven blocks in its Lempel-Ziv parsing would result in an encoder output of length $1 + 2 + 3 + 3 + 4 + 4 + 4 = 21$ code bits. In particular, for the data sequence (19), the seven string

$W_1, ..., W_7$ are (21):

$$\begin{aligned} W_1 &= (1) \\ W_2 &= (1,0) \\ W_3 &= (0,1,1) \\ W_4 &= (0,0,0) \\ W_5 &= (1,0,0,0) \\ W_6 &= (0,1,1,0) \\ W_7 &= (0,0,0,1) \end{aligned} \tag{22}$$

Concatenating, we see that the codeword assigned to data sequence (19) is

$$(1,1,0,0,1,1,0,0,0,1,0,0,0,0,1,1,0,0,0,0,1) \tag{23}$$

Decoding can be performed simply by reversing the encoding process. We omit a detailed description of the Lempel-Ziv decoder, but it is easy to see what the decoder would do. For example, it would be able to break up the codeword (23) into the separate codewords for the phrases, because, from the size *k* of the data alphabet, it is known how many code bits are allocated to the encoding of each Lempel-Ziv phrase. From the separate codewords, the decoder recovers the integer representing each phrase; dividing each of these integers by *k* to obtain the quotient and remainder, the pairs representing the phrases are obtained. Finally, these pairs yields the phrases, which are concatenated together to obtain the original data sequence.

Performance

Let S be a source with a source alphabet of size k. We denote by n the size of the data sequence to compress. It is known [16] that there is a positive constant C_k (depending on *k* but not on *n*) such that

$$H(S) \leq \frac{C^n}{n} \leq H(S) + \frac{C_k}{\log_2 n}. \tag{24}$$

Thus, the Lempel-Ziv code is not quite as good as the Huffman's code. But, there is an important difference, the Huffman's code require knowledge of the source. The preceding performance bounds is valid regardless of the source. Thus, one can use the same Lempel-Ziv code for all sources - such a code is called a *universal code* [16].

2.2.3 Arithmetic Codes

We consider an arithmetic coding [17, 18] of a single source. Consider the sample *X* generated by the source, modeled as a random variable taking its values in

the finite alphabet A and probability $p(a) = Pr[X = a]$ (for each *a* $\in$ A). The arithmetic code works on the following way for each *a*:

1. Consider *I(a)* the interval assigned to letter *a*. At first the interval [0,1] is divided into subintervals of $I(a)$ ($a \in A$), where $I(a)$ is taken to be of length $p(a)$.

2. To each interval $I(a)$ calculate the midpoint x_a.

3. The codeword $B(a)$ assigned to *a* by the arithmetic encoder is of length $L(a) = 1 + \lceil -\log p(a) \rceil$ and is obtained as the first *L(a)* digits to the right of the decimal point in the infinite binary expansion of the real number x_a. Letting $L = L(a)$,

   $x_a = .b_1b_2b_3...b_Lb_{L+1}...$

   $x_a = \left(\frac{b_1}{2}\right) + \left(\frac{b_2}{4}\right) + \left(\frac{b_3}{8}\right) + ... + \left(\frac{b_L}{2^L}\right) + \left(\frac{b_{L+1}}{2^{L+1}}\right) + ...$

   $2^L x_a = \left(2^{L-1}b_1 + 2^{L-2}b_2 + ... + 2b_{L-1} + b_L\right) + fraction$

   $\lfloor 2^L x_a \rfloor = 2^{L-1}b_1 + 2^{L-2}b_2 + ... + 2b_{L-1} + b_L$

   $B(a) = b_1b_2b_3...b_L$ is thus the *L(a)*-digits binary expansion of the positive integer $\lfloor 2^{L(a)} x_a \rfloor$.

The steps above describe the arithmetic encoding scheme. Now we discuss the way the arithmetic decoder works. The distance between x_a and the point

$$\left(\frac{b_1}{2}\right) + \left(\frac{b_2}{4}\right) + \left(\frac{b_3}{8}\right) + ... + \left(\frac{b_L}{2^L}\right) \tag{25}$$

is less than

$$(\frac{1}{2^{L+1}}) + (\frac{1}{2^{L+2}}) + (\frac{1}{2^{L+3}}) + ... = \frac{1}{2^L} \leq \frac{p(a)}{2} \tag{26}$$

by choice of *L(a)*. Since *I(a)* is of length *p(a)* and the point of equation (25) is at most half this length from the centre point of the interval *I(a)*, the point of equation (25) must lie in this interval, too. The decoder computes the point of equation (25) from the received codeword *B(a)*, and then decodes *B(a)* into *a* by finding the unique *I(a)* containing the point of equation (25).

Example 7 *Let {a,b,c,d} be the alphabet with probabilities:*

$$p(a)=\tfrac{1}{2},\ p(b)=\tfrac{1}{4},\ p(c)=\tfrac{1}{8},\ p(d)=\tfrac{1}{8}.$$

Step 1. and 2.

A	***I***	***midpoint x***
a	$[0, \frac{1}{2}]$	$\frac{1}{4}$
b	$[\frac{1}{2}, \frac{3}{4}]$	$\frac{5}{8}$
c	$[\frac{3}{4}, \frac{7}{8}]$	$\frac{13}{16}$
d	$[\frac{7}{8}, 1]$	$\frac{15}{16}$

Step 3. Let $[j]_k$ denote the k-bit expansion of integer j. Then:

$$L(a) = 1 + \lceil -\log_2(\tfrac{1}{2}) \rceil = 2$$
$$\lfloor 2^{L(a)} x_a \rfloor = 1$$
$$B(a) = [1]_2 = 01$$
$$L(b) = 1 + \lceil -\log_2(\tfrac{1}{4}) \rceil = 3$$
$$\lfloor 2^{L(b)} x_b \rfloor = 5$$
$$B(b) = [5]_3 = 101$$
$$L(c) = 1 + \lceil -\log_2(\tfrac{1}{8}) \rceil = 4$$
$$\lfloor 2^{L(c)} x_c \rfloor = 13$$
$$B(c) = [13]_4 = 1101$$
$$L(d) = 1 + \lceil -\log_2(\tfrac{1}{8}) \rceil = 4$$
$$\lfloor 2^{L(d)} x_d \rfloor = 15$$
$$B(d) = [15]_4 = 1111$$

The arithmetic encoder table is therefore

letter	***codeword***
a	*01*
b	*101*
c	*1101*
d	*1111*

Note that if we remove the rightmost bit from each codeword, we obtain another prefix codeword set $\{0, 10, 110, 1111\}$. This will happen any time the interval in Step1 are chosen to never increase in length as one goes from left to right in the unit interval. In this case, one can use the shorter codewords $\{0, 10, 110, 1111\}$ as the codewords generated by the arithmetic encoder, thereby saving one code bit per source letter. (If the intervals sometimes increase in length in going from left to right, then one may not be able to remove the rightmost bit from the codewords.)

Performance

It can be shown [17] that the resulting compression rate satisfies

$$H(S) \leq \frac{C^n}{n} \leq \frac{(H(S^n) + 2)}{n}, \tag{27}$$

where *n* is the length of the source. For large n, we therefore have $\frac{(H(S^n)+2)}{n} \approx H_n$. This is the best one can possibly hope to do. If the source is memoryless and *n* is large one obtains the very good arithmetic code compression rate performance just described but, at the same time, the arithmetic code of low complexity. As discussed for the Huffman code will also achieve a very good compression rate performance but this Huffman code will be very complex. For this reason, arithmetic codes are preferred over Huffman codes in many data compression applications.

2.2.4 Enumerative Codes

Enumerative coding, in its present state of development, is due to Thomas Cover [5]. Enumerative coding is used for a source in which the data sequence are equally likely; the best lossless code for such a source is one which assigns codewords of equal length. Here is Cover's approach to enumerative code designs:

1. Let N be the number of sequences in S_n ($S_n = \{s_1 s_2 ... s_n | s_i \in S, 1 \leq i \leq n\}$). Construct the rooted tree *T* with *N* leaves, such that the edges emanating from each internal vertex have distinct labels from *S*, and such that the *N* sequences in S_n are found by writing down the labels along the *N* root-to leaf paths of *T*.

2. Locate all paths in *T* which visit only unary vertices in between and which are not subpaths of other such paths. Collapse each of these paths to single edges, labelling each such single edge that result with the sequence of labels along the collapsed path. This yields a tree $T^{\times}$ with *N* leaves (the same as the leaves of *T*). Label each leaf of $T^{\times}$ with the sequence obtained by concatenating together the labels on the root-to-leaf path to that leaf; these leaf labels are just the sequence in S_n.

3. Assign an integer weight to each vertex *v* of $T^{\times}$ as follows. If *v* has no siblings or is further to the left than its siblings, assigns *v* a weight of zero. If *v* has sibling or is further to the left, assigns *v* a weight equal to the number of leaves of $T^{\times}$ that are equal to or subordinate to the siblings of *v* that are to the left of *v*.

4. To encode $x^n \in S_n$, follow the root-to-leaf path in $T^{\times}$ terminating in the leaf of $T^{\times}$ labelled by x^n. Let *I* be the sum of the weights of the vertices along this root-to-leaf path. The integer *I* is called the index of x^n, and satisfies

$$0 \leq I \leq N - 1. \tag{28}$$

Encode x^n into the binary codeword of lenght $\lceil \log_2 N \rceil$ obtained by finding the binary expansion of *I* and them padding that expansion (if necessary) to $\lceil \log_2 N \rceil$ bits by appending a prefix of zeroes.

Example 8 *For example, suppose the source satisfies*

$$S_n = \{aaa, aba, abb, abc, bba, bac, caa, cba, cbb, cca\}. \tag{29}$$

Then step 1-3 yield the tree $T^{\times}$ in Figure 1

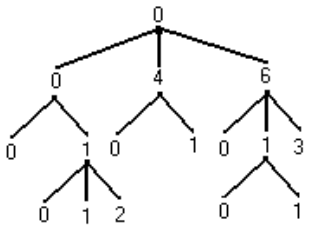

Figure 1: Example of classical enumerative code.

in which every vertex is labelled with its weight from step 3. (The 10 leaves of this tree, from left to right, correspond to the 10 sequences in 29, from left to right.) The I values along the 10 path are just the cumulative sums of the weights, which are seen to give I=0,1,2,3,4,5,6,7,8,9. The codeword length is $\lceil \log_2 10 \rceil = 4$ and the respective codewords are

$$0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001. \tag{30}$$

Sometimes, the tree $T^{\times}$ has such a regular structure that one can obtain an explicit formula relating a data sequence and its index I , thereby dispensing with the need for the tree $T^{\times}$ altogether.

3 Quantum Information Overview

Quantum Information is a new paradigm based on the laws of Quantum Mechanics, it is a completely different way of understanding and codifying information. Rolf Landauer pointed out the importance of the physical systems when dealing with information. According to Deutsch [21], "Landauer was telling everyone that computation is physics and that you can't understand the limits of computation without saying what the physical implementation [i.e. type of hardware] is. He was a lone voice in the wilderness. No one really understood what he was talking about-and certainly not *why*." Information is physical and any processing of information is always performed by physical means - an innocent statement, but its consequences are anything but trivial. When quantum effects become important, for example at the level of atoms and photons, the existing, classical theory of computation become fundamentally inadequate. Entirely new modes of computation and information processing become possible. In the last few years there has been an explosion of theoretical and experimental research in Quantum Computation.

In this section we present the basics of Quantum Computation needed to learn Quantum Information processing. A reader familiar with basic Linear Algebra will presumably have no difficulties in following this section, but for a reader feeling lost we recommend appendix B and an introductory book on Linear Algebra, such as [22].

Quantum Computation [23] is based on the laws of Quantum Mechanics [24], but we present the fundamental notions of quantum information processing from the point of view of a computer scientist, like a mathematical framework. Our main interest is to introduce Quantum Computation and Quantum Information processing. For this reason we are primarily interested in representing a finite set by using a quantum mechanical system. For the sake of simplicity we assume that all the quantum systems handled in this section are *finite-dimensional*.

In section 3.1 we introduce the basic notions and notation of Quantum Information, for a better understanding of the following section about Quantum Information Theory. Then, in section 3.2.1, we introduce an important feature of Quantum Computation, quantum parallelism, whose power is based on the *superposition principle*. Moreover, in section 3.2.2 we describe Grover's algorithm, a search algorithm for a particular element of a long unsorted list. It is an example of an algorithm which achieves a speedup over any classical algorithm, as we shall see. It will be used in our extension of the quantum Huffman algorithm. Finally, we introduce entanglement (in section 3.3), a powerful feature of Quantum Information, as well as one of its applications: dense coding.

3.1 Quantum Information

The purpose of this section is to introduce the basic concepts of Quantum Information. Appendix B introduces some mathematical notions for a better understanding of this section.

Quantum mechanical systems behave in a much different way than classical ones. But, what is it that separates the quantum and the classical world from a computer science point of view? Can Quantum Mechanics improve in any way information processing and manipulation?

In section 2 we have introduced Classical Information Theory where the fundamental unit is the *bit*. The quantum counterpart of the bit is called quantum bit or *qubit*. They have very different properties. Like a bit can be either in the state 0 or 1, a qubit also has two possible states (that correspond to the classical states 0 and 1) that are usually denoted $|1\rangle$ and $|0\rangle$. This "$|\rangle$" is called the *Dirac notation* (appendix B) and we use it in future sections, as it is the standard notation for states in Quantum Mechanics. We present here Quantum Mechanics as in [23], based on four postulates. To study these postulates from a more physical point of view a good reference is [24]. In section 3 we consider only systems whose state is perfectly known, *pure state*. We show how to study their time evolution and how to predict the result of various observations performed on them. However, in practice, the state of the system is not perfectly determined, *mixed state*. When one has incomplete information about a system, one typically appeals to the concept of probability. It is usually presented in the following way: the state of this system may be either the state $|\psi_1\rangle$ with probability p_1 or the state $|\psi_2\rangle$ with probability p_2, etc ... Such that:

$$p_1 + p_2 + ... = \sum_k p_k = 1 \tag{31}$$

The first postulate defines where the quantum world takes place.

Postulate 1 *Associated to any physical system is a complex vector space with 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.*

The simplest quantum mechanical system is the qubit, which is the one we consider in the report.

Definition 11 *A qubit is a quantum state*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{32}$$

where $\alpha, \beta \in \mathsf{C}$ and $|\alpha|^2 + |\beta|^2 = 1$ (normalization condition). The qubit is a state of a quantum two level system. It has two distinguishable basis vectors that form an orthonormal basis for this vector space.

The special states $|0\rangle$ and $|1\rangle$ are known as *computational basis states*, and they are two possible distinguishable states.

For the notions of normalization condition, orthonormal basis, vector space and Hilbert space see in appendix B. Postulate 1 includes one of the most important features of Quantum Information. Notice that from postulate 1 we know that a qubit can be in a state $|\psi\rangle$ either than $|1\rangle$ or $|0\rangle$ and it is possible to form linear combination of states, called **superposition**, according to the *superposition principle*, a fundamental property of Quantum Mechanics. In equation (32) $|\psi\rangle$ is in a superposition of states $|0\rangle$ and $|1\rangle$. Thus, the state of a qubit is a vector (see appendix B) in a two-dimensional complex vector space. But what do α and β in equation (32) mean? A peculiar difference between Classical and Quantum Computation is the information we get after one observes (*measures*) the system. Quantum Mechanics tell us that if one measures a quantum bit in the state (32) then one gets either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. An interesting point is that α and β behave in a different way of classical probabilities. We call the coefficients of the basis states **amplitudes**. So, consider the state

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}}, \tag{33}$$

it is a superposition of the state $|0\rangle$ and $|1\rangle$ with amplitudes $\frac{1}{\sqrt{2}}$ for the state $|0\rangle$, and amplitude $-\frac{1}{\sqrt{2}}$ for the state $|1\rangle$.

An important feature of quantum amplitudes is that they can be negative or more generally complex. Furthermore, when two quantum states overlap, the rule is that one adds the amplitudes rather than the probabilities. These facts have profound consequences. To understand why, it is helpful to recall that, in the macroscopic world, whenever one calculates the probability of something happening in different independent ways, one always add probabilities. Imagine one bet in two different card players at the same time, the probability of winning is the sum of the probabilities for each player. In the quantum world, thanks to the "amplitude quantum rule", one should think twice before betting in two independent events, because it turns out that this rule enables probabilities to subtract from one another rather than always adding up. Quantum rules are not applicable to the macroscopic world, therefore the card players bet would not be a problem. This is an example to understand the concept behind the complex amplitudes. This property is called *quantum interference* and is one of the fundamental properties that differentiates quantum and classical world. In section 3.2 we introduce an exam-

ple where quantum interference is used to speedup an algorithm when compared with the classical counterpart.

Unfortunately, an equally fundamental property of Quantum Mechanics is the *measurement rule*, it describes and severely restricts the way one can observe a quantum state. For example, if we have a superposition of states $|0\rangle$ and $|1\rangle$ we only can get the answer $|0\rangle$ or $|1\rangle$. The measurement, or observation, of a state yields information in a classical form which induces an irrevocable destruction of some remaining information, making further measurements less informative or even completely useless. Therefore we must perform a computation that somehow takes advantage from the superposition principle considering measurement loss. Quantum interference helps overcome the measurement restriction imposed by Quantum Mechanics as we shall see in section 3.2.

Postulate 2 *Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occur is given by*

$$p(m) = \langle\psi| M_m^* M_m |\psi\rangle , \tag{34}$$

and the state of the system after the measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^* M_m |\psi\rangle}}. \tag{35}$$

The measurement operators satisfy the completeness equation (see appendix B),

$$\sum_m M_m^* M_m = I, \tag{36}$$

where I is the identity operator.

The completeness equation expresses the fact that the probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^* M_m |\psi\rangle . \tag{37}$$

Let us consider an example. When we measure a qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ in the computational basis there are two possible outcomes defined by the two measurement operators, $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. We may verify that $I = M_0^* M_0 + M_1^* M_1 = M_0 + M_1$, so the completeness equation is satisfied. Then, when we measure the state $|\psi\rangle$ the probability of obtaining the outcome 0 is

$$p(0) = \langle\psi| M_0^* M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = |a|^2 . \tag{38}$$

And the state after the measurement is

$$\frac{M_0 |\psi\rangle}{|a|} = \frac{a}{|a|} |0\rangle . \tag{39}$$

A similar conclusions would be applicable to the measurement of the outcome 1, with probability $p(1) = |b|^2$ and the state after the measurement being $\frac{M_1|\psi\rangle}{|b|} = \frac{b}{|b|} |1\rangle$. The measurement rule is of great importance to quantum states. In the classical world, different states of an object are usually distinguishable, at least in principle. For example imagine we drop a dice, we can easily identify in which number the dice has landed up, at least in the ideal limit. On the other hand, in quantum systems it is not always possible to distinguish between arbitrary states, only if they are perpendicular. This statement can by proved by contradiction [23]. Consider we have two non orthogonal states $|\psi_1\rangle$ and $|\psi_2\rangle$. So, by contradiction we assume that these states are distinguishable by some measurement. If state $|\psi_1\rangle$ ($|\psi_2\rangle$) is prepared then the probability of measuring j such that $f(j) = 1$ ($f(j) = 2$) must be one. Defining $E_i \equiv \sum_{j:f(j)=i} M_j^* M_j$, these observations may be written as

$$\langle\psi_1| E_1 |\psi_1\rangle = 1;\ \langle\psi_2| E_2 |\psi_2\rangle = 1 \tag{40}$$

From $\sum_i E_i = I$ follows that $\sum_i \langle\psi_1| E_i |\psi_1\rangle = 1$, and since $\langle\psi_1| E_1 |\psi_1\rangle = 1$ we must have $\langle\psi_1| E_2 |\psi_1\rangle = 0$, and thus $\sqrt{E_2} |\psi_1\rangle = 0$. Consider $|\psi_2\rangle = \alpha |\psi_1\rangle + \beta |\varphi\rangle$, where $|\varphi\rangle$ is orthonormal to $|\psi_1\rangle$, $|\alpha|^2 + |\beta|^2 = 1$, and $|\beta| < 1$ since $|\psi_1\rangle$ and $|\psi_2\rangle$ are not orthogonal. Then $\sqrt{E_2} |\psi_2\rangle = \beta\sqrt{E_2} |\varphi\rangle$, which implies a contradition with equation (40), as

$$\langle\psi_2| E_2 |\psi_2\rangle = |\beta|^2 \langle\varphi| E_2 |\varphi\rangle \leq |\beta|^2 < 1, \tag{41}$$

where the second last inequality follows from the observation that

$$\langle\varphi| E_2 |\varphi\rangle \leq \sum_i \langle\varphi| E_i |\varphi\rangle = \langle\varphi|\varphi\rangle = 1. \tag{42}$$

So, by contradiction we have concluded that if we have two non orthogonal states it is not possible to distinguish between them.

The difference between the fully unobservable state of a qubit and the observations we can make is at the heart of Quantum Computation and Quantum Information. The gap between this direct correspondence in Quantum Mechanics makes it difficult to have any intuition about the behavior of quantum systems. However, there is an indirect correspondence: qubit states can be manipulated and transformed in ways which lead to measurement outcomes which depend distinctly on the different properties of the state.

Transformations on qubits are represented by *unitary* matrices (see appendix B). Any quantum evolution on a qubit is described by a unitary matrix, U:

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \tag{43}$$

then,

$$U \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{bmatrix}, \tag{44}$$

which transforms any qubit state $\alpha |0\rangle + \beta |1\rangle$ into the state $(a\alpha + b\beta) |0\rangle + (c\alpha + d\beta) |1\rangle$.

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

$$\left|\psi'\right\rangle = U |\psi\rangle . \tag{45}$$

A closed system is a system which does not interact with any other system. These postulates assures that the evolution of any closed quantum system may be defined by a unitary operator. However there are some unitary operators that are mostly used and natural to consider. We introduce some of them. As we will see in later sections, a very useful evolution matrix is the **Hadamard matrix** (transformation):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{46}$$

which transforms the computational basis $\{|0\rangle , |1\rangle\}$ into the *dual basis* $\{|+\rangle , |-\rangle\}$, where

$$|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ and } |-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{47}$$

as follows:

$$\begin{aligned} H |0\rangle = |+\rangle , H |+\rangle = |0\rangle , \\ H |1\rangle = |-\rangle , H |-\rangle = |1\rangle . \end{aligned} \tag{48}$$

Applying H we can change between the standard and dual basis. It holds $HH^{+} = I$ (where I is the identity matrix), that is H is unitary.

Another very useful transformation is the **NOT** transformation, represented by the matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \tag{49}$$

we would expect this gate to transform the state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ in $|\phi\rangle = \beta|0\rangle + \alpha|1\rangle$. As we easily check X is unitary, $X^+X = I$.

The **Pauli-Y** matrix is also very used:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{50}$$

it transforms $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ into $|\phi\rangle = i(\beta|0\rangle - \alpha|1\rangle)$

The so-called **Pauli-Z** matrix keeps the coefficients of $|0\rangle$ unchanged and flips the sign of that of $|1\rangle$:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{51}$$

However, already finitely many of them are sufficient to perform all quantum computations with an arbitrary precision. A gate is said *universal for Quantum Computation* if any unitary operation may be aproximated to arbitrary accuracy by a quantum circuit involving only those gates. For example, Hadamard, phase, $\frac{\pi}{8}$ and NOT gates are enough to build any unitary operation approximated with arbitrary accuracy (see proof in section 4.5 of [23]), where $\frac{\pi}{8} = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$, $phase = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ and NOT, defined in equation (49). The following theorem generalize the representation of all matrices of degree 2.

Theorem 7 *For each unitary matrix U of degree 2 there exist real numbers α, β and θ such that (page 175 of [23]):*

$$U = \exp i\alpha \begin{pmatrix} \exp i\alpha & 0 \\ 0 & \exp i\alpha \end{pmatrix} \begin{pmatrix} \cos\theta & i\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \exp i\beta & 0 \\ 0 & \exp -i\beta \end{pmatrix}.$$

In principle, there is a continuous range of rotations (by α), phase shifts (with respect to θ) and scale matrices (with respect to β).

So far we know how to describe a quantum system and its evolution. If we have a quantum system made up of two or more distinct quantum systems then it is called composite quantum systems. Our approach is over composite systems of qubits.

Postulate 4 *The state space of a physical system is the tensor product (see appendix B) of the state space of the component physical system. Moreover, if we*

have systems numbered 1 *through* n*, and system numbered* i *prepared in the state* $\left|\psi^{'}\right\rangle$*, then the joint state of the total system is* $|\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$.

It is common to use the notation $|\psi\phi\rangle$ instead of $|\psi\rangle \otimes |\phi\rangle$. "$\otimes$" represents the tensor product, see appendix B.

Suppose we have a system of two qubits. In the classical world there would be four possible states: 00, 01, 10, 11. The quantum counterpart instead of having only four possible states has four *computational basis* vectors: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. This system is a four-dimensional Hilbert space $H_4 = H_A \otimes H_B$. The general state $|\psi\rangle$ of two qubits is a superposition of the states $|00\rangle_{AB}$, $|01\rangle_{AB}$, $|10\rangle_{AB}$ and $|11\rangle_{AB}$:

$$|\psi\rangle = \alpha_{00}\,|00\rangle_{AB} + \alpha_{01}\,|01\rangle_{AB} + \alpha_{10}\,|10\rangle_{AB} + \alpha_{11}\,|11\rangle_{AB} \tag{52}$$

with the constraint that

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{53}$$

The measurement of a state of equation (52) results in $x(= 00, 01, 10 \text{ or } 11)$ and occurs with probability $|\alpha_x|^2$, with the state of the qubit after the measurement being $|x\rangle$. For a two qubit system we could measure just a subset of the qubits. If we measure just the first qubit, the result will be 0 with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, leaving the post-measurement state

$$\left|\psi^{'}\right\rangle = \frac{\alpha_{00}\,|00\rangle_{AB} + \alpha_{01}\,|01\rangle_{AB}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}. \tag{54}$$

$\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$ is the *re-normalization* factor, so it still satisfies the normalization condition.

It is usual to represent states of the computational basis in one of the following forms:

$$|0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |1\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$
$$|2\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |3\rangle = |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{55}$$

The generalization of 2-qubit register of the case of n-qubit register is straightforward.

To deal with n-qubit register we work within a 2^n- dimensional Hilbert space with the following set of basis, called *computational basis*:

$$\mathcal{B} = \{|i\rangle \, | 0 \leq i < 2^n\} \, . \tag{56}$$

The general state of the n-qubit register is

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i \, |i\rangle \text{ with } \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \tag{57}$$

A very important set of basis states is given by the *Bell states*. This states are responsible for many properties of Quantum Computation and Quantum Information, mentioned in section 3.3.

$$\begin{aligned}
\left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}(|00\rangle_{AB} + |11\rangle_{AB}), \\
\left|\Phi^-\right\rangle &= \frac{1}{\sqrt{2}}(|00\rangle_{AB} - |11\rangle_{AB}), \\
\left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}(|01\rangle_{AB} + |10\rangle_{AB}), \\
\left|\Phi^-\right\rangle &= \frac{1}{\sqrt{2}}(|01\rangle_{AB} - |10\rangle_{AB}).
\end{aligned} \tag{58}$$

Among unitary transformations on two qubits states the following transformation has a special role

$$XOR : |x, y\rangle \rightarrow |x, x \oplus y\rangle \, , \tag{59}$$

where $\oplus$ is the exclusive or operation.

In the XOR transformation the first input qubit is called the control qubit and the second input qubit is called the target qubit.

XOR is represented by the matrix:

$$XOR = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{60}$$

Observe that XOR $|00\rangle = |00\rangle$, XOR $|01\rangle = |01\rangle$, XOR $|10\rangle = |11\rangle$ and XOR $|11\rangle = |10\rangle$ XOR is called the CONTROL-NOT (CNOT), since the second qubit (*target qubit*) is flipped if and only if the first (*control qubit*) is 1.

Indeed, inputs $|0\rangle$ and $|1\rangle$ on the control qubit come out on the target qubit output, but a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ on the control qubit is transferred into the Bell state $|\Phi\rangle^+ = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. $|\Phi\rangle^+$ is an *entangled* state, these states are important features of Quantum Computation, we shall introduce entanglement in section 3.3.

Now we introduce a surprising result of Quantum Information Theory: it is the "No-cloning Theorem" due to W. K. Wootters and W. H. Zurek [25]. This result states that it is impossible to clone an arbitrary unknown quantum state.

Theorem 8 ***(No-cloning theorem)*** *An unknown quantum state cannot be cloned. (Namely, there is no unitary transformation U, such that for any one-qubit state* $|\psi\rangle$ *,* $U(|\psi, 0\rangle) = |\psi, \psi\rangle$ *.) The no-cloning theorem holds for any Hilbert space.*

Proof 1 *Assume that such a U exists and for two different orthogonal states* $|\alpha\rangle$ *and* $|\beta\rangle$*,* $U(|\alpha, 0\rangle) = |\alpha, \alpha\rangle$ *,* $U(|\beta, 0\rangle) = |\beta, \beta\rangle$ *. Let* $|\gamma\rangle = \frac{1}{\sqrt{2}}(|\alpha\rangle + |\beta\rangle)$*. Then by linearity and our assumptions,* $U|\gamma, 0\rangle = \frac{1}{\sqrt{2}}(|\alpha, \alpha\rangle + |\beta, \beta\rangle) \neq |\gamma, \gamma\rangle = \frac{1}{2}(|\alpha, \alpha\rangle + |\beta, \beta\rangle + |\alpha, \beta\rangle + |\beta, \alpha\rangle)$.

The no-cloning theorem implies that there is no general unitary transformations for perfect copying of quantum information without destroying the original copy of information. This makes a big difference between classical and quantum information, since we are enable to copy information. We mention some consequences in section 4.

3.2 Parallelism and Grover's Algorithm

Quantum computation is a new paradigm that brings new challenges to computation. The question is: Are we are able to do better in computing using Quantum Mechanics? This new paradigm can not take us any further as a computational power, by other words, all that we can compute with quantum computers we can also compute with classical computers and vice versa. The thing that make us feel interested is whether we can do some computations significantly faster with a quantum computer. The most suitable feature of Quantum Computation that could make it possible is the quantum superposition. This makes it possible to break up a unique task into several subtasks, each of which could be performed in "parallel". This property unique of quantum systems is exemplified in the "Deutsch problem" (section 3.2.1) and in the Grover's algorithm (section 3.2.2).

3.2.1 Quantum Parallelism

The first quantum method to demonstrate the potential of quantum methods over classical ones was proposed by Deutsch [26] in his 1985 paper on quantum par-

allelism. It concerned the calculation of a mathematical function $f(x)$. Suppose, Deutsch asked, you wanted to know whether this function f took the same value for two different values for two different input values, $x = 0$ and $x = 1$. Now, since this was only a simple example, not only was the input restricted to the two values 0 and 1, so also was the output. The function thus operates on a single qubit (although it requires a second qubit to perform the calculation). Deutsch invited us to imagine a function predicting tomorrow's stock exchange movements. Now, suppose it took 24 hours to work out f for each value of x and you needed to know for your investment strategy whether the two values of the function - $f(x)$ and $f(1)$ - gave the same answer, without necessarily needing to know what the answers were. The scenario could be that you will buy some stock in a company, but only if two economic indicators $f(0)$ and $f(1)$ agree with each other.

On a classical computer you would need to do two calculations (one each for 0 and 1) and compare the answers to find out. Unhappily, these two computations would take 48 hours to complete - hardly useful for tomorrow's investment decision. Deutsch showed how you could cut the computation time to 24 hours using quantum parallelism.

The idea is as follows. Imagine we have a quantum oracle (black box) that computes $f(x)$. The function f will take as its input x a number placed in one qubit. In a quantum computer it is now possible to "rotate" the state of the input into a superposition of 0 and 1 using a Hadamard gate. Consider the transformation U_f that takes two qubits to two:

$$U_f : |x\rangle\,|y\rangle \rightarrow |x\rangle\,|y \oplus f(x)\rangle\,. \tag{61}$$

It flips the second qubit if $f(x)$ acting on the first qubit is 1, and does not do anything if f acting on the first qubit is 0. So, this transformation applied twice will tell us if f is balanced or constant. But this does not solve our time problem! Can we get the answer by running the quantum box only once? This is known as the "*Deutsch's problem*".

Since this is a quantum computer, we may take as an input the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then:

$$\begin{aligned} U_f : |x\rangle\,\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow |x\rangle\,\frac{1}{\sqrt{2}}\left(|f(x)\rangle - |1 \oplus f(x)\rangle\right) \\ = |x\rangle\,(-1)^{f(x)}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{62}$$

At this stage f is isolated as an x-dependent phase. If the first qubit is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ we get

$$U_f : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow$$
$$\frac{1}{\sqrt{2}}[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle]\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{63}$$

Now we can perform a measurement that projects the first qubit onto the basis

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle). \tag{64}$$

We obtain $|+\rangle$ if the function is balanced, and $|-\rangle$ if the function is constant.

This is the solution of Deutsch's problem. It is also a separation between what a classical computer and a quantum computer could achieve. The quantum computer only has to run the black box once, while the classical computer must run it twice. This happens because the quantum computer extracts "global" information about the function by acting on a superposition of $|0\rangle$ and $|1\rangle$, this is called "**quantum parallelism**" [26], combined with a property of Quantum Machanics called **interference**. Notice that in a classical computer if we have two alternatives, $f(0)$ and $f(1)$, they forever exclude one another. In a quantum computer it is possible for the two alternatives to *interfere* with one another to yield some global property of the function f, by using the Hadamard gate to recombine the different alternatives, as was done in equation (63). What is more, the program would not need to be limited to just two inputs. We can apply the same idea to a function acting on a N register. According to:

$$U_f : |x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle, \tag{65}$$

choosing the input register to be in the state

$$[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)]^N = \frac{1}{2^{\frac{N}{2}}}\sum_{x=0}^{2^N-1}|x\rangle, \tag{66}$$

and by computing $f(x)$ only once, we can generate a state

$$\frac{1}{2^{\frac{N}{2}}}\sum_{x=0}^{2^N-1}|x\rangle|f(x)\rangle. \tag{67}$$

This state encodes the global properties of f.

If this sounds too good to be true, it is in a sense. Even if we produce exponentially many outputs for the price of one, Quantum Mechanics severely restricts the way in which you can look at the result. The measurement rules of Quantum

Mechanics state that the measurement of the output yields information in a classical form which induces an irrevocable destruction of some remaining information. It is the quantum parallelism that Shor invokes in his factoring algorithm [1]. So although quantum computing offers seemingly unlimited amounts of massive parallelism without the need of extra hardware, there is also a price to be paid: It is impossible to read all the information contained on the final state. In section 3.2.2 we present an algorithm that uses quantum parallelism to exploit the speed of a search.

3.2.2 Grover's Algorithm

Grover's search algorithm [27] is an excellent example of quantum parallelism. It enable us to get a complexity speedup when compared with the classical search algorithms. We will use it later to decrease the complexity time of a particular quantum encoding scheme. This algorithm is important because of the existence of many computer science problems based on search problems. For example, imagine we are given a search space of size N with no prior knowledge about the structure of the information in it. Without loss of generality, suppose that the elements are numbers from 0 to $N-1$. Classically we would test each element at a time, until we find the one we were looking for. This takes an average of $\frac{N}{2}$ attempts, and N in the worst case, therefore the complexity is $O(N)$. In this section we prove that using Quantum Mechanics we only require $O(\sqrt{N})$ steps.

For simplicity, assume that $N = 2^n$, for some integer n. Grover's algorithm has two registers, the first one with n qubits and the second one with one qubit. We start by creating a superposition of all 2^n computational basis states $\{|0\rangle, ..., |2^n - 1\rangle\}$ of the first register. To do that we proceed as follows. We initialize the first register in the state $|0, ..., 0\rangle$ and apply the operator Hadamard $H^{\otimes n}$:

$$\begin{aligned} |\psi\rangle &= H^{\otimes n} |0, ..., 0\rangle \\ &= (H|0\rangle)^{\otimes n} \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle . \end{aligned} \tag{68}$$

where $|\psi\rangle$ is a superposition of all basis states with equal amplitudes given by $\frac{1}{\sqrt{N}}$. The second register begins with $|1\rangle$ and, after applying a Hadamard gate, it will be in state $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$.

We define, $f : \{0, ..., N-1\} \rightarrow \{0, 1\}$, $f(x) = 1$ if x is a solution to the search problem, otherwise $f(x) = 0$. This function is used in the classical algorithm. In the quantum algorithm, let us assume that it is possible to build a linear unitary operator also dependent on f, U_f , such that:

$$U_f \left(|i\rangle \, |j\rangle \right) = |i\rangle \, |j \oplus f(i)\rangle \, . \tag{69}$$

We call U_f *oracle* or a black box. In the above equation, $|i\rangle$ stands for a state of the first register, so i is in $\{0, ..., 2^n - 1\}$, and $|j\rangle$ stands for a state of the second register, so j is in $\{0, 1\}$, and the sum is modulo 2. The action of the oracle is:

$$\begin{aligned} U_f(|i\rangle \, |-\rangle) &= \frac{U_f(|i\rangle \, |0\rangle) - U_f(|i\rangle \, |1\rangle)}{\sqrt{2}} \\ &= \frac{|i\rangle \, |f(i)\rangle - |i\rangle \, |1 \oplus f(i)\rangle}{\sqrt{2}} \\ &= (-1)^{f(i)} \, |i\rangle \, |-\rangle \, . \end{aligned} \tag{70}$$

In the last equality, we have used the fact that

$$1 \oplus f(i) = \begin{cases} 0, i = i_0 \\ 1, i \neq i_0 \end{cases} \tag{71}$$

where i_0 stands for the searched element. We say that the oracle marks the solutions to the search problems by shifting the phase of the solution.

Let us call $|\psi_1\rangle$ the resulting state of the first register Now look at what happens when we apply U_f to the superposition state coming from the first step, $|\psi_1\rangle \, |-\rangle$. The state of the second register does not change.

$$\begin{aligned} |\psi_1\rangle \, |-\rangle &= U_f(|\psi\rangle \, |-\rangle) \\ &= \frac{1}{\sqrt{2}} \sum_{i=0}^{N-1} U_f(|i\rangle \, |-\rangle) \\ &= \frac{1}{\sqrt{2}} \sum_{i=0}^{N-1} (-1)^{f(i)} \, |i\rangle \, |-\rangle \, . \end{aligned} \tag{72}$$

$|\psi_1\rangle$ is a superposition of all basis elements, but the amplitude of the searched element is negative while the amplitude of all others are positive. The searched element has been marked with a minus sign. This result is obtained using a feature called *quantum parallelism*. At the quantum level, we have a superposition of all database elements. The position of the searched element is known: it is

the value of *i* of the term with negative amplitude in equation (72). This quantum information is not fully available at the classical level. To obtain information out of a quantum state we make measurements, and, at this point, it does not help if we measure the state of the first register, because it is much more likely that we obtain a non-desired element, instead of the searched one. Before we can perform a measure, the next step should be to increase the amplitude of the searched element while decreasing the amplitude of the others. This is quite a general approach: quantum algorithms work by increasing the amplitude of the states which carry the desired result. After that, a measurement will hit the solution with high probability. Imagine we have a superposition like the one in equation (72). Figure 2 depicts the superposition idea, with all coefficients, lets call them c_i ($0 \leq i \leq 2^n - 1$), equal to $\frac{1}{\sqrt{2^n}}$.

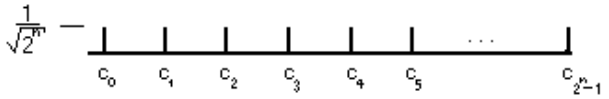

Figure 2: Initial state.

The operator U_f flips the sign of the amplitudes of all elements that are the one searched for (i_0), by other words the elements such that $i = i_0$. Imagine that element 3 is the search one. Then, c_3 becomes $-\frac{1}{\sqrt{2^n}}$. This is depicted in figure 3.

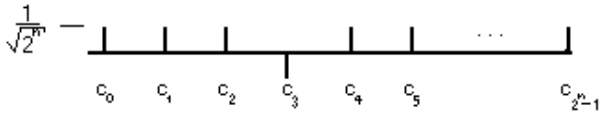

Figure 3: Amplitudes after oracle query.

Now we shall work out the details by introducing the circuit for Grover's algorithm and analyzing it step by step.

The circuit for one Grover iteration *G* is given in figure 4. Each interaction of the Grover's algorithm increases the coeficient of the searched element, this way we are increasing the probability of getting this element after a measurement. In figure 4 The states $|\psi\rangle$ and $|\psi_1\rangle$ are given by equations (73) and (72), respectively, and the operator $2|\psi\rangle\langle\psi| - I$ is called inversion (explained in a moment). We will also show how each Grover operator application raises the amplitude of the searched element. $|\psi_1\rangle$ can be rewritten as:

$$|\psi_1\rangle = |\psi\rangle - \frac{2}{\sqrt{2^n}}|i_0\rangle\,, \tag{73}$$

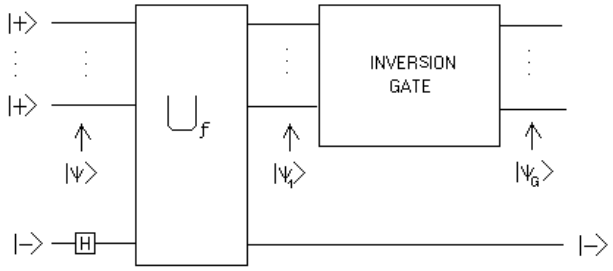

Figure 4: Outline of each interaction of the Grover's algorithm.

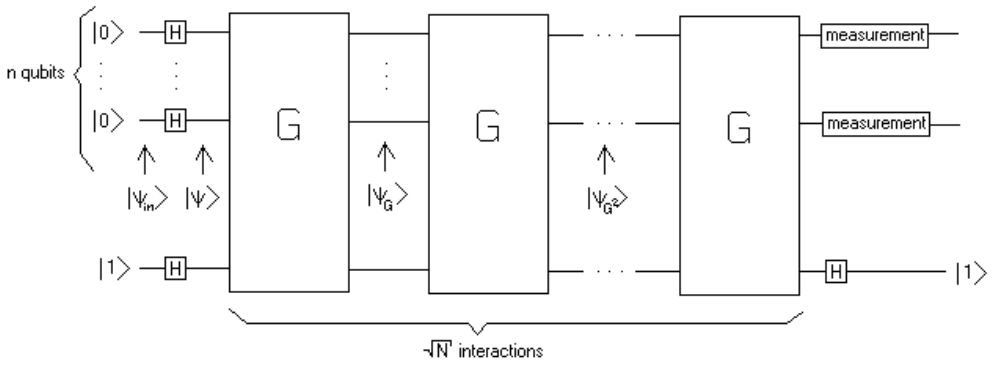

Figure 5: Amplitudes after one iteration of Grover's algorithm.

where $|i_0\rangle$ is the searched element. $|i_0\rangle$ is an element of the computational basis. Note that

$$\langle\psi|i_0\rangle = \frac{1}{\sqrt{2^n}}. \tag{74}$$

Let us calculate $|\psi_G\rangle$ of figure 5. Using equations (73) and (74), we obtain

$$\begin{aligned} |\psi_G\rangle &= (2\,|\psi\rangle\,\langle\psi| - I)\,|\psi_1\rangle \\ &= \frac{2^{n-2}-1}{2^{n-2}}\,|\psi\rangle + \frac{2}{\sqrt{2^n}}\,|i_0\rangle\,. \end{aligned} \tag{75}$$

This is the state of the first register after one application of *G*. The second register is in the state $|-\rangle$. Notice in equation (75) the action of the inversion operator. It duplicates the coeficient of searched element. After one application of Grover's algorithm we get the amplitudes depicted in figure 6. Applying the operator G more times increases the probability of obtaining the searched element as close to

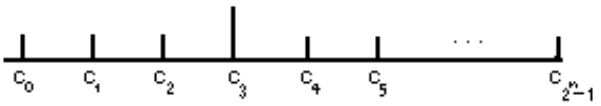

Figure 6: Outline of each interaction of the Grover's algorithm.

one as we desire. The operator G is applied $O(\sqrt{N})$ times, as depicted in figure 6. This way, when measuring the final state, we obtain with a high probability the searched state.

Grover's algorithm may be improved in linear time, those methods are described in [28].

The following theorem sets the complexity of the Grover's algorithm (page 90 of [29]).

Theorem 9 *By using a quantum circuit making $O(\sqrt{2^n})$ queries to a blackbox function f, one can decide with nonvanishing correctness probability if there is an element $x \in H^{\otimes n}$ such that $f(x) = 1$.*

The theorem above is only applicable when we search a single element. Grover's algorithm has been generalized [30]. In the case we are searching more than one element we apply $O\left(\sqrt{\frac{N}{M}}\right)$ interactions of the Grover's algorithm, for M solutions out of N possibilities. If we do not know the number of solutions, M, in the classical case need $O(N)$ steps, which in the quantum case we achieve a speed up to $O(\sqrt{(M+1)(N-M+1)})$ steps. In case of counting the number of solutions with error $\sqrt{M}$ it takes $O(\sqrt{N})$. Thanks to the usefulness of searching in many problems, Grover's algorithm has been applied with some other purposes. Two examples are the quantum counting algorithm [31] and minimum finding algorithm [32]. The latter example is an algorithm to find the minimum of an unsorted list $T[0, \cdot, N-1]$, each holding a value from an ordered set. We use this results in section 4.2.3 to do a search in our quantum encoding scheme.

3.3 Entanglement

One of the most specific and important concepts for Quantum Computation and Quantum Information Theory is quantum entanglement which is also one of the most puzzling concepts of Quantum Physics.

Entanglement plays a central role in Quantum Information Theory that extends Classical Information Theory. Three important possible applications of entanglement are: teleportation (see [23]), dense coding (described in section 3.3.1) and quantum key distribution (well explained in [23]).

Let's start with an example. A 2-qubit register can be in the state

$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{76}$$

If we observe the first bit of this state using the standard observable M$=$ $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, then we get the value 0 with probability $\frac{1}{2}$ and the value 1 also with probability $\frac{1}{2}$ (hence the outcome is completely random). After such an observation the state $|\psi\rangle$ collapses into the state $|00\rangle$ in the first case and into the state $|11\rangle$ on the second case. If, afterwards, we measure the second qubit its value is determined uniquely, with probability 1. We see that if a quantum register is in the above state $|\psi\rangle$, then the two qubits are not independent. In addition, in such a case particular qubits of the quantum register no longer have an identity! Each of them is actually in a mixed state with probability $\frac{1}{2}$ in the state $|0\rangle$ and with probability $\frac{1}{2}$ in the state $|1\rangle$. Notice that this correlations only exit in the quantum theory, they have no analog in the classical theory.

How specific and important is this example? We naturally expect that there should be cases where the qubits are completely independent and therefore they can be separately acted on.

This is actually the case if a state $|\psi\rangle$ of a 2-qubit register is the tensor product $|\psi_1\rangle \otimes |\psi_2\rangle$ of two 1-qubit states $|\psi_1\rangle = \alpha_0 |0\rangle_A + \alpha_1 |1\rangle_A$ and $|\psi_2\rangle = \beta_0 |0\rangle_B + \beta_1 |1\rangle_B$, i.e.,

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \left(\sum_{i=0}^{1} \alpha_i |i\rangle_A\right) \otimes \left(\sum_{j=0}^{1} \beta_j |j\rangle_B\right). \tag{77}$$

If now we observe the first qubit of the state $|\psi\rangle$, we get,

$$\begin{aligned} &0 \text{ with probability } |\alpha_0\beta_0|^2 + |\alpha_0\beta_1|^2 = |\alpha_0|^2, \\ &1 \text{ with probability } |\alpha_1\beta_0|^2 + |\alpha_1\beta_1|^2 = |\alpha_1|^2. \end{aligned} \tag{78}$$

Moreover, after the observation of the first qubit the state $|\psi\rangle$ is reduced to $|\psi_2\rangle$; after the observation of the second qubit to $|\psi_1\rangle$.

If a quantum pure state of a Hilbert space H cannot be obtained as a tensor product of two quantum states from Hilbert spaces of dimensions smaller than that of H, then the state is called **entangled**. This is the case of the above example of $|\Phi^{+}\rangle$, where the two qubits are not independent.

Entanglement arises in a natural way as a result of interactions between quantum systems. In addition, some quantum operations create entangled states out of separable states. For example, if the XOR operation is applied to the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$, the entangled state $|\Phi^{+}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ (one of the Bell

states) is created. Entangled states do not exist in classical systems. The Bell states are often called EPR pairs-"EPR" stands here for "Einstein, Podolsky and Rosen" and that fact that the qubits are not independent creates a so-called "EPR" channel. This is called a channel because, since we can have two particles physically far apart and represented by a state (with preserved attributes), by applying some specific transformation on one particle the global state changes.

The source of various paradoxes related to entangled states is the fact that a pair of particles in an entangled state can be physically separated. However, each measurement on one particle of such an entangled pair immediately determines the state of another one, no matter how far apart they are. A measurement of an entangled particle exhibits therefore the so-called "non-local property", something that cannot happen from the point of view of classical physics without an instantaneous communication among the particles. In section 3.3.1 we described an entanglement application, the dense coding protocol.

According to [33], entanglement between a pair of quantum systems in a maximally entangled state is the purest form of inherently quantum information: it is capable of interconnecting two particles far apart, it cannot be copied, eavesdropped without disturbance, nor can it be used by itself to send classical messages. At the same time it can assist, in some sense (mention on section 3.3.1), in improving both classical and quantum communication.

3.3.1 Dense Coding

The aim of dense coding is to transmit more than one bit of information in each qubit transmitted. Like in many ideas in Quantum Computation and quantum information, it is more easily understood using the metaphor of a game involving two parties, Alice and Bob. Imagine Alice and Bob share two particles on the *Bell state* $|\phi^+\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. They could have met a long time ago to generate it or it could have been a third person shipping one particle to Bob and another to Alice.

If Alice wants to send two classical bits ($00, 01, 10$ or 11) of information to Bob she performs on her particle one of the Pauli rotations (described in section 3) as shown in the second column of the table below. By doing this $|\phi^+\rangle_{AB}$ is transformed into one of the mutually orthogonal *Bell states*. At this stage she sends her qubit to Bob, who receives it, and then performs an orthogonal collective measurement on the pair that projects it onto the maximally entangled basis. From the outcome Bob can, unambiguously, distinguish between the four possible rotations Alice could have done. Suppose Alice and Bob have shared also a protocol with a relation one-to-one, by this we refer to having each pair of bit corresponding to one and only one rotation. Bob, by now, would be able to know which bits Alice wanted to send.

Alice's bits	Pauli's rotations	Alice transforms $\|\phi^+\rangle_{AB}$ to the Bell state:
00	I	$\|\phi^+\rangle_{AB} = \frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$
01	X	$\|\psi^+\rangle_{AB} = \frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$
10	Y	$\|\phi^-\rangle_{AB} = \frac{1}{\sqrt{2}}(\|01\rangle - \|10\rangle)$
11	Z	$\|\psi^-\rangle_{AB} = \frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$

Dense coding has a very nice advantage, if the message is confidential, Alice does not need to worry that an eavesdropper can intercept the transmitted qubit and decipher her message. The qubit transmitted by Alice contains no information by its own. All the information is on the correlations between qubit A and B. They used entanglement as a resource.

We could argue that Alice and Bob still need to use the channel twice to send two bits of information. The initial qubits must be transmitted, as well as the qubit Alice wants to send. But, anyway, the *Bell pair* could have been exchanged a long time ago. So in an emergency dense coding would be very useful if there is a need to be faster. Quantum dense coding was recently put into practice in a channel of 100 km [3].

4 Quantum Data Compression

The purpose of this section is to study quantum data compression. A compressing, studies the tangible resources necessary to represent a certain information.

In section 4.1 we introduce some properties of Quantum Information Theory, we mention the quantum encoding schemes developed so far and we present in detail a particular quantum encoding scheme 4.2.1. The following three sections, 4.2.2, 4.2.3 and 4.2.4, are three different proposals to improve that encoding scheme.

4.1 Quantum Information Theory and Data Compression

In this section we introduce the main properties of Quantum Information Theory and a framework to a particular quantum encoding scheme.

We may think about Quantum Information Theory as a parallel of the Classical Information Theory, but it is more peculiar, the aim is also to understand the difference between the quantum and the classical world, from a computer science point of view. What resources, unavailable in a classical world, are being utilized in a Quantum Computation? How could they improve in any way information processing and manipulation? The answer to this questions are in the heart of Quantum Information Theory research and they will clarify the potential of this topic and bring a new way of understanding and using information, with a great change to result in important transformations in the Information Society we live in. Experimental demonstrations of Quantum Information Theory are of great importance in topics like, security systems, communication systems, etc, and may have in a close future a revolutionary impact in the society.

Quantum Information Theory in fundamentally richer than Classical Information Theory, one of the central differences is the nature of the information itself.

In [34] R. Jozsa has introduced an overview of the main properties of quantum information. The classical compression scheme can not be used to compress quantum data because of quantum systems behavior. The no-cloning theorem (see section 3.1) is one property of quantum systems that point a difference between classical and quantum information. This theorem states the impossibility of copying unknown quantum information, providing a gap on accessible quantum information when compared to the classical case, which can obviously be copied. But now you may be asking yourself, if we said that the classical information is a special case of a quantum information. How is it possible that we can actually copy classical information? The detail here is that the no-cloning theorem do not prohibit all quantum states to be copied, it only forbid non-orthogonal quantum states. Classical information can be thought as merely orthogonal quantum states.

Another example of the difference between quantum and classical information is the way we distinguish different items of information. One of the quantum systems properties with a great importance to *distinguishing quantum states* is the *measurement rule*, introduced in section 3.1. In the classical world, distinct states of an object are usually distinguishable, at least in principle. For example imagine we drop a dice, we can easily identify in which number the dice has landed up, at least in the ideal limit. On the other hand in quantum systems it is not always possible to distinguish between arbitrary states. We explain it in section 3.1.

Quantum information is really different from classical one. At first, from the possibility of superposition of quantum states it seem that we had an infinite way of representing information with only one qubits. But the impossibility of accessing all the quantum information makes some restrictions on that. We only can read classical information and it would be interesting to quantify the accessable information. Unfortunately there is no general methods, but there are some bounds. The most important one is the *Holevo bound* (see page 531 of [23]), it plays an important role in many applications of Quantum Information Theory. The Holevo bound is an upper bound for the accessible information. A direct consequence of the Holevo bound is that one bit cannot be sent or represented with less than one qubit and it also manifests itself in the observation that the quantum channel capacity is no bigger than the classical channel capacity.

To encode quantum data we can not just go ahead and use the same techniques used to compress classical information. We must adapt or create quantum encoding schemes capable of deal with quantum information properties.

One approach to construct quantum compression schemes is to begin with the classical encoding schemes and investigate how this algorithms must be reinterpreted or modified to fit with quantum data. For example, quantum codes must allow superpositions of different codewords.

In section 3 we have introduced Quantum Mechanics using Dirac's notation. Alternatively we can use a tool called *density operator*. This tool is mathematically equivalent to the Dirac's notation, but it provides a more convenient language to describe some scenarios in Quantum Mechanics. It turns out that all the postulates of Quatum Mechanics can be reformulated in terms of the density operator language. We use the density operator language from now on for its simplicity to describe a quantum data source. Our quantum source would be described as an *ensemble of pure states*. More precisely, suppose a quantum system is in one of a number of states $|\psi_i\rangle$, where *i* is the index of the different states, with respective probabilities $p(i)$. We denote the ensemble by $\Sigma = \{p_i, |\psi_i\rangle\}$. The density operator of the system is defined by

$$\rho \equiv \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{79}$$

The first result in Classical Coding Theory was the *noiseless coding theorem*, it shows the importance of the Shannon entropy for Classical Information Theory as a measure of the tangible resources necessary to represent the information produced by message source. In quantum systems, the expression for entropy was proposed by von Neumann [35] in 1955. The von Neumann entropy $S(\rho)$ is

$$S(\rho) = -Tr\rho \log_2 \rho, \tag{80}$$

for an ensemble of states, of a quantum system, described by a density operator ρ. In the quantum case the probabilities are replaced by the density operator. If we denote the eigenvalues of ρ as λ_x then the von Neumann entropy can be re-expressed by,

$$S(\rho) = -\sum_x \lambda_x \log_2 \lambda_x. \tag{81}$$

The classical and quantum entropies are formaly similar, but actualy they are quite different. Consider we have a source X that produces a message x with probability $p(x)$. If we have a device that codes each message x from a quantum signal source, then the ensemble of this quantum source is represented by the density operator

$$\rho = \sum_i p(x_i)\pi_{x_i}, \tag{82}$$

where π_i are the projection $\pi_{x_i} = |x_i\rangle \langle x_i|$. Notice that the states $|x_i\rangle$ $(1 \leq i \leq n)$ are not necessarily orthogonal. Formally, an i.i.d. quantum source is described by a Hilbert space H, and a density matrix ρ on that Hilbert space. The two entropies are equal only when the states $|x_i\rangle$ are orthogonal, otherwise $S(\rho) < H(X)$ [36]. In the decoding process if we do not have orthogonal states, then we are not able to distinguish between them, so we can not recover the entire information perfectly. The idea of quantum compressing scheme is to take states in the Hilbert space $H^{\otimes n}$ of the source to states in a $H^{nR} - dimensional$ state space, where R is the compression rate of the source. The approach between the input and output state is denoted fidelity. It is defined, mathematically, by $F = \langle x_{input}| \rho_{output} |x_{input}\rangle$.

As in the classical case, the quantum entropy represents the mean number of qubits necessary to encode the states in the ensemble in an ideal encoding [5].

Theorem 10 ***(Quantum noiseless coding theorem)*** *Let M be a quantum signal source with a signal ensemble described by the density operator ρ and let $\delta, \epsilon > 0$.*

1. *Suppose that $S(\rho) + \delta$ qubits are available per M signal. Then for sufficiently large N, groups of N signals from the signal source M can be transposed via the available qubits with fidelity $F > 1 - \epsilon$.*

2. *Suppose that $S(\rho) - \delta$ qubits are available per M signal. Then for sufficiently large N, if groups of N signals from the signal source M are transposed via the available qubits, then the fidelity $F < \epsilon$.*

Notice that the state ρ in theorem 10 can be interpreted as a part of a larger system which is in a pure state. In this case ρ would be a mixed state which could be due to the entanglement between the Hilbert space H and the remainder of the system. Schumacker's theorem 10 proves that the von Neumann entropy is the lower bound limit of a lossless compression. He also gives an idea of how to do the compression, but it does not ensures optimal compression. The fact that Schumacher's data compression scheme only can achive a lossless data compression scheme is the asymptotic limit and that it is very inefficient for a small number of qubits. However it has inspired a number of other encoding schemes [8, 7, 9, 10, 11, 12, 13]. We explain one encoding scheme in more detail in section 4.2.1.

Quantum analogues have been proposed to the classical encoding schemes that we have presented in section 2.2. The quantum analogue of the classical Arithmetic coding (see section 2.2.3) was proposed by I. Chuang and S. Modha [11]. They studied the problem of compressing a block of symbols emitted by a quantum source. R. Cleve and DiVincenzo [13] have proposed a block coding algorithm, which is, in fact, a generalization of the classical enumerative coding (see section 2.2.4). Recently, S. Braunstein *et al.* [9] have proposed a quantum analogue of the classical Huffman's code (see section 2.2.1). However, it was done in an way that unecessary information have been carried. Some studies have been done considering universal quantum compression. The quantum encoding schemes we have mentioned so far are not applicable to the case where we do not know the average density operator, which the construction of the protocols is dependent on. Is there a protocol which faithfully compress quantum information even if we do not know the density matrix of the source? R. Jozsa and the Horodecki [8] constructed a quantum universal fixed-length code for the case that all we know about the source is that its von Neumann entropy does not exceed some given value S. Their protocol is efficient in the i.i.d. case when the entropy rate of the source is less than the rate of the code. Otherwise, this protocol demolish the state the state unrecoverably. The optimality of their code among quantum fixed-length codes is proven in a sense of compression rate. M. Hayashi and K. Matsumoto have proposed [12] a quantum universal variable-length source coding applicable for any probability distribution of quantum states. Moreover R. Jozsa and S. Presnell proposed another quantum universal data compression scheme based on the classical Lempel-Ziv code presented in section 2.2.2.

B. Schumacher and M. Westmoreland have sketched a framework to discuss those codes [37]. However, the encoding scheme we introduce in this report does

not follow that framework.

From now on we describe the framework of the encoding scheme proposed by K. Böstrom and T. Felbinger [6]. During its description we nake consideration on the physical realization of such a scheme. Let Ω be a quantum source. We interpret all its valid objects as normalized vectors of a Hilbert space ν. The encoding task is basically a mapping (see definition in a book of Linear Algebra, *e.g.* [22]) to another Hilbert space $\mathcal{M}$. The mapping must be linear and isometric, in order to preserve linearity and norm, respectively. Now we consider the structure of $\mathcal{M}$. Like in the classical countepart it has a *quantum alphabet*, which are orthogonal vectors spanning the Hilbert space $\mathcal{M}$. For example $|1\rangle, |0\rangle$. We denote the size of $\mathcal{M}$ as *k*. A quantum word, or a *message*, is a sequence of the alphabet symbols (quantum letters). So the messages are composed by the tensor product of the quantum letters. The space of all the message of size n, $|x_1\rangle ... |x_n\rangle$ is

$$\mathcal{H}^{\otimes n} = \bigotimes_{i=1}^{n} \mathcal{H} = \mathcal{H} \otimes ... \otimes \mathcal{H}. \tag{83}$$

These kind of spaces, which all the words have a fixed number of qubits are called *block spaces*. In the classical case the fixed length codes were less efficient than the variable length codes. In quantum codes the same happends. Therefore we define a space that enables quantum messages of different length, *indeterminate-length space*, to be in superposition.

$$\mathcal{H}^{\oplus} = \bigoplus_{n=0}^{\infty} \mathcal{H}^{\otimes} = \mathcal{H}^{\otimes 0} \oplus \mathcal{H} \oplus \mathcal{H}^{\otimes 2} \oplus .... \tag{84}$$

where $\oplus$ is explained in appendix B. This space looks perfectly acceptable from the theoretical point of view, but how could we actually implement such a space? We get into this consideration later in this section. This way a message may contain a superposition of messages of distinct length, for example, if $|1\rangle$ and $|0\rangle$ $\in \mathcal{H}$,

$$\frac{1}{\sqrt{2}} \left(|10\rangle + |110101\rangle\right) \in \mathcal{H}^{\oplus}. \tag{85}$$

A message with components of distinct length is called a *indeterminate-length message*. So, block spaces are particular cases of indeterminate-length spaces.

The message space $\mathcal{H}^{\oplus}$ contains every quantum messages that can be composed using quantum alphabet words from $\mathcal{H}$ and the laws of Quantum Mechanics. However, since we have a system we only can realize a finite dimensional subspace of the general message space. We define the *r-bounded message space*

$$\mathcal{H}^{\oplus r} = \bigoplus_{n=0}^{r} \mathcal{H}^{\otimes n}, \tag{86}$$

containing all superpositions of messages of maximal length r. We may think about a space $\mathcal{R} = \mathcal{D}^{\otimes s}$ which is the physical realization of $\mathcal{H}^{\oplus r}$. The physical space $\mathcal{R}$ represents the space of all pysical states of the register, while the message space $\mathcal{H}^{\oplus r}$ represents the space of valid codewords that can held by the register and it is isomorphic to a subspace $\mathcal{H}_p^{\oplus r}$ of the physical space $\mathcal{R}$. Let dim ($\mathcal{H}$)$= r$, then s must satisfy the following.

$$\begin{aligned} & dim(\mathcal{H}^{\oplus r}) \leq dim(\mathcal{D}^{\otimes s}) \\ \rightarrow \sum_{n=0}^{r} k^n = \frac{k^{r+1}-1}{k-1} & \leq k^s \\ & \rightarrow r+1 \leq s. \end{aligned} \tag{87}$$

This way, to implement a message space $\mathcal{H}^{\oplus r}$ in a space $\mathcal{R} = \mathcal{D}^{\otimes(r+1)}$.

To implement this in a protocol we use a k-ary representation of the natural number. In the next section we explain how to it.

4.2 Quantum Encoding Schemes

The problem of compressing is central to storage and transmission of data. In this correspondence, we introduce in section 4.2.1 a quantum compressing scheme and in sections 4.2.2, 4.2.3 and 4.2.4 we investigate more efficient extensions of that scheme.

4.2.1 Lossless Quantum Data Compression Scheme

Let us introduce a lossless quantum encoding scheme developed by Kim Böstrom and Timo Felbinger [6].

An encoding is a mapping from a source space to a code space. So, its specification is just a linear mapping, which transforms the basis element of the source alphabet, $|w_i\rangle \in \nu$, into the basis elements of the code space, $|v_i\rangle \in \mathcal{M}$. Notice that the vectors $|w_i\rangle$ are orthogonal between each other, as well as the $|v_i\rangle$ basis vectors.

$$|w_i\rangle \longmapsto |v_i\rangle \,. \tag{88}$$

Since the mapping from one space to another is isometric the coding is lossless, which is the type of compression we study in this report.

The compression operator can be defined as

$$C = \sum_{w \in v} |c(w)\rangle \langle w| \,. \tag{89}$$

Since the encoder is lossless the inverse of C, C^{-1}, is the decoder, which we denote by

$$D = \sum_{v \in \mathcal{M}} |w\rangle \langle c(w)| . \tag{90}$$

Recently some proposals have been made to which code space we should use. In the encoding scheme that we will describe moreover the code space is called *Neutral-prefix space* and it is defined as follows. Consider $Z_k(i)$ the *k*-ary representation of a natural number i. Consider $Z_k(0) = \phi$. Define an orthonormal basis message:

$$\mathcal{B}_r := \{|Z_k(0)\rangle , ..., |Z_k(k^r - 1)\rangle\} \tag{91}$$

with variable-length of maximal lenght *r*.

The length of the $|Z_r(i)\rangle$ is given by

$$|Z_r(i)| = \lceil \log_k(i+1) \rceil, \tag{92}$$

$\lceil x \rceil$ denotes the largest integer larger or equal to x.

Notice that, for example if we choose $r = 3$ and $k = 2$, there are messages of length less than 3 that do not belong to the space spaned by the basis $\mathcal{B}_r$. An example is the message $|10\rangle$. Let us denote the space spaned by $\mathcal{B}_r$, $\mathcal{N}_r$. So we have the relation

$$\mathcal{N}_r \subset \mathcal{H}^{\oplus r}. \tag{93}$$

The reason why it turns out to be not so clear to define a code space is because of its physical realization. For example the state $|01\rangle + |1101\rangle$ stills a discussion whether it could possibly be implemented or not. Therefore we define other space, which we will use to implement the space $\mathcal{N}_r$. The problem we have in hands is the physical implementations of states with a superposition of messages with different length. Therefore we define a space $\mathcal{B}_R$ of the same size as $\mathcal{B}_r$ but with all the elements with equal size. This space instead of the $|Z_k(i)\rangle$ has an *r*-extention of those numbers by adding as much leading zeros as necessary to achieve the length r, *e.g.* $Z_k^r(i) := 0 \cdots 0 Z_k(i)$. More precisely,

$$\mathcal{B}_R := \{|Z_k^r(0)\rangle , ..., |Z_k^r(k^r - 1)\rangle\}. \tag{94}$$

Notice that the basis of $\mathcal{B}_R$ are orthonormal.

Now it is about time to Alice and Bob appear. Imagine that Alice wants to send some encoded quantum data to Bob. We consider that they both have a quantum computer and they have a quantum and a classical channel connecting them. So, Alice prepares each message $|x_i\rangle$, $1 \leq i \leq n$, from a source set χ, each with a probability $p(x_i)$. The source is represented by the ensemble $\Sigma := \{p, \chi\}$. She

encodes each of this words into variable length codewords $c(x_i) \in \mathcal{N}_r$ of maximal length r. If we denote the dimension of the source space by d, then r must be at least as large as the number of qubits necessary to represent d, so $r \geq \lceil \log_r d \rceil$. To define the encoder operator C, see equation (89), Alice check if the messages of χ are linerly dependet. If so she creates a new set $\Xi = \chi$ and removes the most probable message from Ξ, which she appends to a list $\mathcal{L}$. Then she again takes the most probable message from Ξ and checks whether it is linear dependent with the words of $\mathcal{L}$. If not she adds the element to $\mathcal{L}$, otherwise she follows taking elements of Ξ until there is no elements left. This procedure leads to a linear independent list, $\mathcal{L}$, of elements from χ. The probability of the messages in $\mathcal{L}$ are in a decreasing order. Now Alice performs a *Gram-Schmidt* (see appendix B) orthonormalization of the elements of the list $\mathcal{L}$, starting with the first element of the list, which is the one with higher probability.

$$\begin{aligned} |w_1\rangle &:= |x_1\rangle \,, \\ |w_i\rangle &:= N_i \left[I - \sum_{j=1}^{i-1} |w_j\rangle \langle w_j| \right] |x_i\rangle \,, \end{aligned} \tag{95}$$

with $2 \leq i \leq d$ (considering d the number of elements in $\mathcal{L}$), N_i are normalization constants and I is the identity matrix. Consider

$$\mathcal{B} = \{|w_1\rangle \,, \cdots \,, \, |w_d\rangle\}, \tag{96}$$

then the elements of $\mathcal{B}$ form an orthonormal basis for the source set χ. Now we encode the elements of χ into the code space $\mathcal{B}_R$,

$$|c(w_i)\rangle := |Z_k^r(i-1)\rangle \,, \tag{97}$$

with $1 \leq i \leq d$, with increasing significant lenght

$$L_c(w_i) = \lceil \log_k(i) \rceil. \tag{98}$$

The length of each message is given by

$$L_c^{max}(x) = \max_{1 \leq i \leq d} \{ L_c(w_i) || \langle w_i | x \rangle |^2 > 0 \}, \tag{99}$$

for all the messages of the set χ. Each codeword is in space $\mathcal{B}_R$. At this point Alice has quantum data, which are the encoded messages. The respective length of each message is represended in classical data and encoded by a classical encoding scheme. Now Alice is ready to send the data to Bob. She sends the classical data to Bob through a classical channel. The quantum data is sent in the following way. Alice reads the length of the message that she wants to send and removes the

leading zeros from that message. Then the codeword would be in space spanned by $\mathcal{B}_r$. The quantum message is now ready to be sent to Bob.

Bob decodes the data in a similar way. The classical data has the information about the length of the quantum codewords. So Bob picks each quantum codeword and appends the leading zeros needed to fulfil the r required zeros to map the codewords to the space $\mathcal{B}_R$. He applies the decoder D, equation (90), to every codeword and gets the original messages.

The following algorithms describe three different ways to choose the axes of $\mathcal{B}$, equation (96). The first, called *Brute Force* algorithm, just goes around all the possible set of axes and choose the best one. The second one, *Improved BF* algorithm, uses Grover's algorithm idea to speed up Brute Force algorithm. Finally, the *Adapted Algorithm*, gives another idea how to order probabilities in the lossless quantum data compression scheme just described.

This compression protocol has a codewords length average, I_c, upper bounded by [6]

$$I_c \leq \log_2(dim\nu) + log_2 k. \tag{100}$$

where ν is the source space.

4.2.2 Extension - Brute Force

In this section we describe a lossless quantum encoding scheme. Our aim is to compress more our data by changing a bit the previous algorithm.

We begin with an example where the above algorithm could be improved.

Example 9 *A source $\mathcal{S} = (S, P)$ has a probability distribution*

P=$\{\frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{11}{168}\}$ *and S=* $\{(1,0,0), (0,1,0), (0,0,1), \frac{1}{\sqrt{2}}(0,1,1),$ $\frac{1}{\sqrt{3}}(0,1,2), \frac{1}{\sqrt{3}}(0,2,1), \frac{1}{\sqrt{4}}(0,1,3), \frac{1}{\sqrt{4}}(0,3,1)\}$. *The source has only real vectors to be easier to illustrate it on a figure.*

Figure 7 shows S, the quantum states we want to encode, that are decribed by vectors with a color different of black. The codewords are represented with "(codeword,probability)" on one side.

The graphic on the left has the codewords we get with the scheme described in the previous section. With the previous scheme the smaller codeword correspond to the most probable word to encode, represented in blue on that graphic.

The codewords length average of the quantum scheme of the previous section is:

$$\mathcal{L}(C) = \frac{1}{6} \times 0 + \frac{1}{7} \times 1 + \frac{1}{8} \times 5 \times 2 + \frac{11}{168} \times 2 = \frac{32}{21} \tag{101}$$

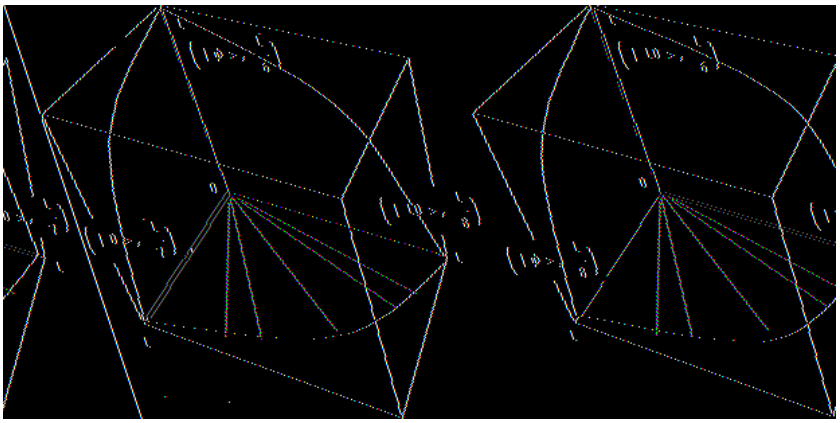

Figure 7: Two diferent codeword assignments.

In this example the axes chose to the coding are not the ones that minimizes the length average. The graphic on the right side of the figure our example of axes that give a smaller length average:

$$\mathcal{L}(C) = \frac{1}{6} \times 2 + \frac{1}{7} \times 1 + \frac{1}{8} \times 0 + \frac{1}{8} \times 4 \times 1 + \frac{11}{168} \times 1 = \frac{25}{24} \qquad (102)$$

The value obtained is almost half of the previous one! The length average was decreased by just changing the correspondence between the codewords and the axes.

In the encoding scheme described in the previous section, the *Gram-Schmidt* decomposition, equation (95), begins by choosing the most probable word to encode. Set this one as an axes to the encoding scheme and encode it with the smallest codeword. But not always this procedure is the way to get the smaller length average.

Now we describe a different way to choose the axes to get the smallest length average. We call the following algorithm *Brute Force algorithm* because it is based on going through all possible set of axes and choose the one with the lowest $\mathcal{L}$. Its description provides a good understanding of what we could improve in the lossless quantum encoding scheme of the previous section algorithm. Moreover we describe how to improve the Brute Force algorithm, in a sense to avoid going through all possible sets and this way save some time.

So, how do we construct this Brute Force algorithm? In algorithm 2 we have its description. Consider we want to encode a source $\mathcal{A} = \{|a_1\rangle, ..., |a_n\rangle\}$, where the vectors $|a_i\rangle$ $(1 \leq i \leq n)$ belong to a Hilbert space of dimention d (first

item of algorithm 2). To all possible subsets $\wp$=$\{|a_{i_1}\rangle ... |a_{i_d}\rangle\}$ (items 2 and 4) of $\mathcal{A}$ ($\wp$ is selected to have all the elements different of each other) we apply the folowing. We check if the elements of $\wp$ are linearly independents (item 5). If not this set could not be a base of our data. If so we apply the *Gram-Schmidt* normalization to normalize the elements of $\wp$ (items 6 and 7). To be possible to encode as we have done in the lossless compression scheme algorithm, we assign $|c(a)_j\rangle := |Z(j)\rangle$, being $|a\rangle$ the word to encode and $1 \leq j \leq d$ (item 8). This way we have encoded the axes. Since $\wp$ as the same dimension of $\mathcal{A}$, all the source alphabet can be expressed in the normalized axes. Our next move is to encode the source alphabet. The length of the codewords is $l(b_j) = \lceil \log_k(j) \rceil$. Now we calculate the $\mathcal{L}$ of $\mathcal{A}$. The $\wp$ is a base of $\mathcal{A}$, each element of $\mathcal{A}$ can be represented as a linear combination of the elements of $\wp$, or its respective codewords. We go through all elements of $\mathcal{A}$ (item 9) and sets the length of each element equal to the length of the longest word belonging to $\wp$ of its linear combination (item 10). Its respective codeword length is used to calculate the average codewords length L. We save the smaller $\mathcal{L}$ and its respective bases in item 11.

We go through all the elements of $\mathcal{A}$ and apply the encoder C, equation (89), to each element of $\mathcal{A}$. In item we output the smallest $\mathcal{L}$ and its respective base, codewords and the elements of $\mathcal{A}$ encoded.

Algorithm 2 ***Brute Force:***

1. *To encode:* $\mathcal{A} = \{|a_1\rangle, ..., |a_n\rangle\}$*, where* $|a_i\rangle$ *belong to a Hilbert space of dimension d.*
2. $\wp$=$\{|a_{i_1}\rangle, ..., |a_{i_d}\rangle \mid a_{i_1}, ..., a_{i_d} \in \mathcal{A} \wedge a_{i_j} \neq a_{i_l} \wedge \forall_{i,j}(1 \leq i, j \leq d)\}$
3. *L*=∞;
4. *For every* $|a_{i_1}\rangle ... |a_{i_d}\rangle \in \wp$:
5. *If* $\mathcal{B} = \{|a_{i_1}\rangle, ..., |a_{i_d}\rangle\}$ *are linearly independent:*
6. $|b_1\rangle := |a_{i_1}\rangle$,
7. $|b_j\rangle := N_j[1 - \sum_{k=1}^{k-1} |a_{i_k}\rangle \langle a_{i_k}| \, |a_{i_j}\rangle]$, *j=2,...,d;* N_j *are normalization constants;*
8. *assign to every b*$\in \mathcal{B}$: $|c(b_j)\rangle := |Z_K^r(j)\rangle$, *j=1,...,n;* *and l(*$b_j$*)=* $\lceil \log_k(j) \rceil$ *(length of the codeword)*
9. *For every* $|v\rangle \in \mathcal{A}$ *, s=d;*
10. *while* ($\langle v|b_s\rangle = 0$, *s=s-1, l=l+l(*b_s*));*

11. *If l<L, L=l and $\mathcal{B}_f = \mathcal{B}$;*

12. output L, $\mathcal{B}$, C and $C\left|a_i\right\rangle$.

Performance

The Brute Force algorithm, as the name mention, goes around all possible set of axes and chose the best one (the one with lowest $\mathcal{L}$). Even having by grand that it gets be best solution it can not always be applied, thanks to the time it takes to explore all the possible sets of axes. To count the number of step it takes we must consider that for this algorithm it is different to have the base $\wp_1 = \{\left|1\right\rangle, \left|2\right\rangle, \left|3\right\rangle\}$ or $\wp_2 = \{\left|1\right\rangle, \left|3\right\rangle, \left|2\right\rangle\}$, since the Gram-Schmidt normalization gives different results in each case. Thus, we must examine every set of d-elements (dimension of the Hilbert space), so in this setp the time it takes is upper-bounded by a linear function on n^d. Then to check linear independence, to normalize and assing codewords take time upper bounded by $k \times d$ (where k is a constant). To calculate the length of the source encoded we take time upper bounded by $n \times d$. Finally we need more time n to calculate output the encoded source. The complexity of the algorithm only considers the factor with bigger power. In our case the final complexity is upper bounded by a linear function on n^{d+k}, where k is a constant. The performance of the algorithm depends mostly on the dimention of the Hilbert space of the source, since the complexity time depends in power on the dimention.

4.2.3 Extension - Improved BF

The Brute Force algorithm is not applicable in many cases thanks to its complexity time. The *Improved BF* is based on the Brute Force algorithm, speeding up its search time using the Grover's algorithm and its extensions. Grover's algorithm search for a particular element of a long unsorted list. As described in section 3.2.2 it is based on a function f that:

$$f(x) = \begin{cases} 0, x \neq x_0 \\ 1, x = x_0 \end{cases} \tag{103}$$

where x_0 is the searched element.

In the *Improved BF* algorithm we are not just searching for a single element, but for a basis, $\mathcal{B}$. So, to adapt to the search quantum algorithm we consider each possible basis as an element. The problem to use this function in our case is to define which element or elements we are searching for. In Brute Force algorithm we seached for the basis with lowest $\mathcal{L}$. Or, in other words, the basis with $\mathcal{L}$ as closest to the entropy value as possible. But it is not always true that the lowest $\mathcal{L}$ is equal to the entropy, so we must find a way of selecting the elements with lowest

$\mathcal{L}$ to use it as the range of function f. To this purpose we use the upper bound of $\mathcal{L}$, stated in equation (100). This way, function f instead of considering $x = x_0$, just search all the basis with $\mathcal{L}$ lower than the upper bound, which we denote by $\mathcal{L}_{superior}$. In this case we are not seaching a single element. Section 3.2.2 describes how the Grover's algorithm works in a case of searching a single element. At the end of that section is stated the speedup we get seaching M elements out of N. The function f, in our case, becames:

$$f(x) = \begin{cases} 1, \mathcal{L}(x) \leq \mathcal{L}_{superior} \\ 0, \mathcal{L}_{superior} \leq \mathcal{L}(x) \end{cases} \tag{104}$$

In the end of section 3.2.2 we mentioned the complexity time speed up of some extentions of Grover's algorithm [30]. The complexity time of the quantum algorithms of the extensions of Grover's algorithm are in the following table.

Problem	Quantum Complexity	Classical Complexity
Searching M elements	$\mathrm{O}\left(\sqrt{\frac{N}{M}}\right)$	$\mathrm{O}\left(\frac{N}{M}\right)$
Minimum Finding	$O\left(\sqrt{N}\right)$	$\mathrm{O}(N)$

Now we apply these quantum search algorithms and the oracle f, equation (104), to find the basis with lowest codeword length average. The algorithm that comes up to our minds as a solution for this problem is the "Minimum Finding". Algorithm 3, *Improved BF 1*, only applies the "Minimum Finding" algorithm to find the basis with lowest $\mathcal{L}$. It has complexity time of $\mathrm{O}\left(\sqrt{N}\right)$.

Now we introduce Algorithm 4, *Improved BF 2*, it decreases the complexity time of the search, but the probability of finding the minimum also decreases. So, depending on our goals we may balance between time and optimal results. The idea is as follows. At first we search the elements with $\mathcal{L}(x) \leq \mathcal{L}_{superior}$ (item 1 of the algorithms). This search only can be applied if we have some previous information about the number of basis with $\mathcal{L}(x) \leq \mathcal{L}_{superior}$. Then we search between those elements which has smaller codeword length average by applying the "Minimum Finding" algorithm. Algorithm 4 has a complexity time of $O(\sqrt{M} + \sqrt{\frac{N}{M}})$. Notice that between the two items we cannot make measurements. Otherwise we would loose the superposition of the states found with that property.

Algorithm 3 ***Improve BF 1***

1. *Apply the "Minimum Finding" algorithm to find the basis with minimum* $\mathcal{L}(x)$.

Algorithm 4 ***Improve BF 2***

1. *Searching* M *elements;*
2. *Finding the minimum between this elements.*

Performance To study the complexity time of an algorithm that is composed of a sequence of more than one algorithm, we must consider the cemplexity time of all those algorithm. The one that is defined by a funcion with biggest power is the complexity time of our main algorithm. To study the complexity of algorithm 4 we consider the complexity of the algorithms we used in items 1 and 2. The algorithms have complexity time equal to the subalgorithm with bigger complexity time. From the table above we conclude that algorithm 4 has complexity time equal to $O(\sqrt{M} + \sqrt{\frac{N}{M}})$. As we have seen, the complexity time of algorithm 3 is $O\left(\sqrt{N}\right)$. Therefore algorithm 4 has a smaller complexity time. So, we can choose more speedup or optimality of the result.

4.2.4 Extension - Adapted Algorithm

As we described, the lossless compression scheme of the previous section, similarly to the classical Huffman's algorithm, order probabilities in a decreasing order to start the construction of the algorithm. The *Adapted Algorithm*, introduces a slightly difference, it order probabilities depending on its *densities*. What do we mean by *densities* and why does such idea could improve our axes search? At first we should remember again that we are dealing with vectors in a Hilbert space, instead of just real numbers as in the classical case. Therefore, choosing one vector to be an axes influences all the vectors that have a projection on him. So, to choose the first axes vector we pick the one that sum up more probabilities projected on him, because vectors are encoded by a linear combination of its basis. So, how do we do that? To make it explicit let us return to example 9 and apply the *Adapted Algorithm* (in example 10). At the beginning we make a table with each probability as the label of each column celland then we do the same to the line labels. We fill in only the cells below and including table diagonal, since the rest would repeat information. Then, to every p_i and p_j ($i < j$ and $1 < i, j < 8$), if $\langle i|j\rangle \neq 0$ fill in the cell with the number 1, otherwise fill in with number 0 (following table). This is the same as saying that if the vectors are orthogonal, there is

no projection on this basis, we set the cell equal to zero, otherwise, there is a projection on this basis, so we set it equal to one. After filling in this quadratic table we add one more line with the sum of the probabilities projected in each vector. In our example, the line and column with a different type of letter is respective to the vector with probability p_3. The sequence $(0, 0, 1, 1, 1, 1, 1, 1)$ means the vector $|3\rangle$ has a projection in every vectors of the source except p_1 and p_2. Therefore in the last line we have $\frac{41}{56}$, that is the sum of all the probabilities except p_1 and p_2. See example 10.

Example 10 *$\mathcal{S}$=(S,P)*

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
p_1	*1*							
p_2	*0*	*1*						
p_3	*0*	*0*	1					
p_4	*0*	*1*	1	*1*				
p_5	*0*	*1*	1	*1*	*1*			
p_6	*0*	*1*	1	*1*	*1*	*1*		
p_7	*0*	*1*	1	*1*	*1*	*1*	*1*	
p_8	*0*	*0*	1	*1*	*1*	*1*	*1*	*1*
$\sum prob$	$\frac{1}{6}$	$\frac{17}{24}$	$\frac{41}{56}$	$\frac{5}{6}$	$\frac{5}{6}$	$\frac{5}{6}$	$\frac{5}{6}$	$\frac{5}{6}$

$\frac{1}{6} < \frac{17}{24} < \frac{41}{56} < \frac{5}{6}$. *Therefore we choose $|4\rangle$ to be one axes and the one with lowest codeword length.*

Moreover we choose the one with bigger sum of probabilities to be the basis axes with lowest codeword length. In example 10 it is vector $|4\rangle$. Now the algorithm follows in a similar way but with a difference concerning the table construction. Since we have chosen one axes, this one will no longer be concerned in the table. Now we must consider that the other axes we will choose need to be orthogonal to the one already chosen. So, we must project each of the vectors from the source to the space orthogonal to the axes already chose. It is a similar idea to the *Gram-schimdt* procedure (described in appendix B). So, to get another axes vector we construct another table concerning each vector from the source and the ones orthogonal to the axes. Before introducing the table continuing the previous example we would like to state two remarks. Firstly it is important to note that in our example $P|2\rangle$, $P|3\rangle$, $P|5\rangle$, $P|6\rangle$, $P|7\rangle$ and $P|8\rangle$, are the same. This happens because $P|i\rangle$ is a projection of $|i\rangle$ over the plan orthogonal to the axes already found. In our example the projection of all the vectors enumerated before over the plan orthogonal to $|4\rangle$ are the same.

	$P\|1\rangle$	$P\|2\rangle$=$P\|3\rangle$=...=$P\|8\rangle$
$p_{\|1\rangle}$	1	
$p_{\|2\rangle}$	0	1
$p_{\|3\rangle}$	0	0
$p_{\|5\rangle}$	0	1
$p_{\|6\rangle}$	0	1
$p_{\|7\rangle}$	0	1
$p_{\|8\rangle}$	0	0
$\sum$prob	$\frac{1}{6}$	$\frac{17}{24}$

So, since $\frac{1}{6} < \frac{17}{24}$, the new axes just found is $P\,|2\rangle = \frac{1}{\sqrt{2}}(0,1,-1)$. The last axes only can be $P\,|1\rangle$, in our case is equal to $|1\rangle = (1,0,0)$.

In the previous example we got the axes: $|4\rangle = \frac{1}{\sqrt{2}}(0,1,1)$; $P\,|2\rangle = \frac{1}{\sqrt{2}}(0,1,-1)$; and $|1\rangle = (1,0,0)$. With respectively decreasing codeword length. Therefore $l(c(|4\rangle)) = 0$, $l(c(P\,|2\rangle)) = 1$ and $l(c(|1\rangle)) = 2$. So, the average codeword average of the source is:

$$L(C) = \frac{1}{6} \times 2 + \frac{1}{7} \times 1 + \frac{1}{8} \times 4 \times 1 + \frac{11}{168} \times 1 + \frac{1}{8} \times 0 = \frac{25}{24} \qquad (105)$$

Performance

The time the *Adapted Algorithm* takes is upper bounded by a linear function on n^2, being n the number of elements to encode. n^2 is the complexity of the first table we make. The other tables have lower complexity. Therefore, since the complexity time of our protocol is the sum of the complexity time we take to calculte the numbers that are in the tables, the complexity time that counts is only the one of the first table n^2. Notice that the complexity of the sum of functions with different power is always upper bounded by the complexity time of the one with bigger power.

5 Outlook and Conclusion

Our report introduced the main concepts of classical and quantum data compression for noiseless and lossless channels and presented some improvements to a particular quantum encoding scheme.

Firstly, in section 2, we introduced the main concepts of Classical Coding Theory, as well as the four of the most currently used classical encoding schemes: Huffman's Code, the Lempel-Ziv Code, the Arithmetic Code and the Enumerative Code. In section 3, we presented a brief overview of the essentials of quantum information. In section 3.1 we explained the main properties of quantum information and its differences when comparing to classical information. Quantum compression algorithms are mainly based on well known classical encoding schemes adapted to the properties of quantum information, which may actually imply fundamental modifications in the scheme. For example, we are not able to clone unkown quantum states because of the unitary evolution of quantum systems. Related to this and to the measurement rule is the fundamental impossibility of distinguishing two unkown quantum states. Another quantum rule, the possibility of having coherent superposition of states, can also be used to decrease the complexity time of some algorithms. It provides "quantum parallelism" (see section 3.2.1), used in Grover's (see section 3.2.2) and Shor's algorithms. Finally, when we are dealing with composite quantum systems, the tensor product structure of the Hilbert space is instrumental in the definition of entangled particles, a powerful and exclusive property of quantum systems (see section 3.3) which offers correlations that are impossible to simulate classicaly. The Classical Coding Theory and the properties of quantum information are then used in section 4 to study Quantum Data Compression. In section 4.1 we introduce some concepts of Quantum Information Theory. Moreover, we describe three improvements of the lossless quantum data compression scheme of K. Boström and T. Felbinger [6] (see section 4.2). In this scheme the Hilbert space of the source message we want to encode is spanned by a set of vectors. Those vectors are encoded into codewords. So, each source message is encoded in a linear combination of the codewords. The length average is calculated on the size of the longest basis codeword of each encoded source word. Our aim is to achieve a codeword length average as lower as possible. In [6] they choose the most probable source word to be encoded with the smaller codeword. However, in this specific scheme, this procedure does not lead us to the smaller codeword length average we could achieve. This happens because the source messages can be spanned in more than one codeword. In the first extension, *Brute Force* (see section 4.2.2), we point out an example where the algorithm could be improved and we construct an extension to achieve the optimal codeword length average to this scheme. Basically it goes around all possible solutions. It implies a drawback in complexity time, which is a exponential

function of N (the number of source messages). Therefore in section 4.2.3 we describe a quantum algorithm to find the optimal solution, *Improved BF*, which uses Grover's algorithm and its extensions to do the search and this way it decreases the complexity time. This way we achieve a complexity time of $O(\sqrt{M} + \sqrt{\frac{N}{M}})$. At last we describe the *Adapted Algorithm*. In this algorithm we just search the optimal solution in a different way. Instead of choosing the most probable source word to be encoded with the smaller codeword, like in [6], we choose the source word which has the biggest probability of having another source word spanned on itself to be encoded with the smallest codeword. The *Adapted Algorithm* is the one with the lowest complexity time. However we did not manage to proof its optimality in this specific scheme. Therefore, since the *Brute Force* algorithm and the *Improved BF* algorithms find the optimal solution to this scheme we would recommend this ones. The *Improved BF* has the advantage of having a polinomial complexity time.

5.1 Future Work

The *Adapted Algorithm* algorithm could be improved by finding an upper bound related to the von Neumann entropy for the compression that can be achieved.

Future work may also extend the study of compression of entangled states, since this states are very useful to some Quantum Information applications, like quantum criptography.

The idea of a classical channel to send the codewords length information could have been studied from the point of view of its effectiveness. One mistake most probably would damage all the quantum codewords encoded. In this sequence it would be interesting to study a reliable extension of this scheme to a noisy channel.

6 Acknowledgements

This report was carried out in Lisbon, in my home university, Instituto Superior Técnico, under the guidance of my supervisor, Yasser Omar, whom I wish to thank for fruitful discussions about the topic of this report. I had also a long distance co-supervisor, Luca Aceto, from Aalborg University, who I am grateful for his valuable comments on my work. I am also thankful to Vlatko Vedral for his helpful suggestion to the topic of my report. I gratefully acknowledge the hospitality of the people of GoLP.

A Independent Identical Distributed Variables

We introduce some statistics [38] to describe the type of variable we are using.

Definition 12 *Two random variables X and Y are said to be independent if for any two subsets A and B of R,*

$$\begin{aligned} P(X \in A \text{and} Y \in B) = \\ P(X \in A; Y \in B) = \\ P(\{X \in A\} \cap \{Y \in B\}) \\ = P(X \in A) \times P(Y \in B). \end{aligned} \tag{106}$$

More generally, $X_1, ..., X_n$ are said to be (mutually) independent if for any subsets $A_1, ..., A_n$ of R, the events $\{X_1 \in A_1\}$, $\{X_2 \in A_2\}$, ..., $\{X_n \in A_n\}$ are mutually independent: i.e.

$$\begin{aligned} P(X_1 \in A_1, X_2 \in A_2, ..., X_n \in A_n) = \\ P(\{X_1 \in A_1\} \cap \{X_2 \in A_2\} \cap ... \cap \{X_n \in A_n\}) \\ = P(X_1 \in A_1) \times P(X_2 \in A_2) \times ... \times P(X_n \in A_n). \end{aligned} \tag{107}$$

In particular, if $X_1, ..., X_n$ are independent discrete random variables with probability mass function $f_1, ..., f_n$, respectively, then

$$P(X_1 = k_1, ..., X_n = k_n) = f_1(k_1) \times ... \times f_n(k_n) \tag{108}$$

Definition 13 *If $X_1, ..., X_n$ each have the same distribution function, then they are said to be identically distributed. If their common distribution function is F and they are independent as well, we say that $X_1, ..., X_n$ are independent and identically distributed (i.i.d.) with distribution F.*

B Hilbert Spaces

Quantum Mechanics is based on a set of postulates (presented in section 3) as its own rules, for its understanding is entirely necessary a good assimilation of Linear Algebra. For this reason we introduce here the basic and fundamental concepts of Linear algebra. If you feel lost while reading this appendix we recommend [22]. Linear Algebra is presented in Quantum Mechanics with the Dirac notation, that's why it may look a bit fearsome, but it is just algebra! Therefore we will introduce Dirac's notation.

The basic elements of Linear Algebra are **vector spaces**. We study the vector space in C^n, the space of all n-tuples of complex numbers, $(z_1, ..., z_n)$. We denote the elements of a vector space as *vectors* and we present them in a column matrix representation:

$$\begin{bmatrix} z_1 \\ . \\ . \\ . \\ z_n \end{bmatrix}. \tag{109}$$

The Dirac notation for a vector in a vector space is $|\psi\rangle$. The addition operation in C is defined as:

$$\begin{bmatrix} z_1 \\ . \\ . \\ . \\ z_n \end{bmatrix} + \begin{bmatrix} z_1^{'} \\ . \\ . \\ . \\ z_n^{'} \end{bmatrix} \equiv \begin{bmatrix} z_1 + z_1^{'} \\ . \\ . \\ . \\ z_n + z_n^{'} \end{bmatrix}, \tag{110}$$

where $z_i + z_i^{'}$ are the common addition of complex numbers. Moreover, the *multiplication by a scalar* operation is defined as:

$$z \begin{bmatrix} z_1 \\ . \\ . \\ . \\ z_n \end{bmatrix} \equiv \begin{bmatrix} zz_1 \\ . \\ . \\ . \\ zz_n \end{bmatrix}, \tag{111}$$

where z is a complex number.

A vector space also contains a *zero vector*, denoted by 0. It satisfies $|v\rangle + 0 = |v\rangle$, where $|v\rangle$ is any other vector. A *vector subspace* of a vector space V is a subset W of V such that W is also a vector space.

Now we introduce some definitions for the complex matrices. Let U be a matrix such that:

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \tag{112}$$

then

$$\begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{113}$$

i.e., $U^+U = I$. a^* stands for the complex conjugate of complex number a and $U^+ = (U^*)^T$. The notation U^T indicates the transpose of U. U^T and U^* are

$$U^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \text{ and } U^* = \begin{pmatrix} a^* & b^* \\ c^* & d^* \end{pmatrix}, \text{ respectively.} \tag{114}$$

A *spanning* set for a vector space is a set of vectors $|w_1\rangle, ..., |w_n\rangle$ such that any vector in the vector space can be written as a linear combination $|w\rangle = \sum_i a_i |w_i\rangle$ of vectors in that set. An example of a spanning set for C^2 are the vectors:

$$|w_1\rangle \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \; |w_2\rangle \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{115}$$

because any vector in C can be written as a linear combination of $|w_1\rangle$ and $|w_2\rangle$. A vector space have several spanning sets.

We say that a set of vectors is linearly dependent if there exists a set of complex number $a_1, ..., a_n$, for at least one a_i different from zero, such that:

$$a_1 |w_1\rangle + ... + a_n |w_n\rangle = 0. \tag{116}$$

If a set of vectors is not *linearly dependent*, then it is *linearly independent*. We call a set of vectors that can span a vector space V, the *basis* of V. The number of elements in the basis is denoted as the *dimension* of the vector space.

Now we shall introduce the *linear operators*. A linear operator between two vector spaces W and V is defined to be any function A:

$$A\left(\sum_i a_i |w_i\rangle\right) = \sum_i a_i A\left(|w_i\rangle\right). \tag{117}$$

The viewpoint of linear operators and matrix are known to be equivalent. Therefore we may use both notation on the report.

With these conventions, the inner product on a Hilbert space can be given by a matrix representation.

$$\langle v|w\rangle = \begin{bmatrix} v_1^* & . & . & . & v_n^* \end{bmatrix} \begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_n \end{bmatrix}. \tag{118}$$

The vector $\langle v|$ has a nice interpretation as a row vector whose components are complex conjugates of the corresponding components of the column vector representation of $|v\rangle$.

Consider now a vector $|v\rangle$ in a vector space V, and a vector $|w\rangle$ in a vector space W. A very useful linear operator from V to W is the so called **outer product**. It is represented by:

$$\left(|w\rangle \langle v|\right)\left(\left|v'\right\rangle\right) \equiv |w\rangle \left\langle v|v'\right\rangle |w\rangle. \tag{119}$$

We can take linear combination of outer product operators $|w\rangle\langle v|$, by definition $\sum ia_i |w_i\rangle\langle v_i|$ is a linear operator which, when acting on $\left|v^{'}\right\rangle$, produces $\sum ia_i |w_i\rangle\left\langle v_i|v^{'}\right\rangle$ as output.

An important result of the outer product is the **completeness relation** for orthonormal vectors. Let $|i\rangle$ be any orthonormal basis for the vector space V, so an arbitrary vector $|v\rangle$ can be written as $|v\rangle\sum iv_i|i\rangle$ fro some set of complex numbers v_i. Note that $\langle i|v\rangle = v_i$ and therefore

$$\left(\sum i|i\rangle\langle i|\right)|v\rangle = \sum i|i\rangle\langle i|v\rangle = \sum iv_i|i\rangle = |v\rangle. \tag{120}$$

From the last equation it follows that

$$\sum i|i\rangle\langle i| = I. \tag{121}$$

This is the *completeness relation*. **Hilbert space** A Hilbert space is a mathematical framework suitable for describing the concepts, principles, processes and laws of Quantum Mechanics. We can say that to each isolated quantum system corresponds a Hilbert space.

A finite-dimensional Hilbert space H is a complete (defined later in this section) vector space over complex numbers which is equipped with an inner-product (defined in a moment) $H \times H \rightarrow C, (x, y) \rightarrow \langle x|y\rangle$. All n-dimensional Hilbert spaces are isomorphic (defined in a moment), and we can, therefore, denote any such space by H_n.

Definition 14 *An **inner-product space** H is a complex vector space, equipped with an inner-product* $< .|. >$*: H*$\times$ *H* $\rightarrow$ *C satisfying the following axioms for all vectors* ϕ, ψ, ϕ_1 *and* $\phi_2 \in$ *H, and any* c_1, $c_2 \in$ *C.*

$$\begin{gathered} <\phi|\psi> = <\psi|\phi>^*, \\ 0 \leq <\psi|\psi> \text{ and } <\psi|\psi> = 0 \text{ if and only if } \psi = 0, \\ <\psi|c_1\phi_1 + c_2\phi_2> = c_1 <\psi|\phi_1> + c_2 <\psi|\phi_2>, \end{gathered}$$

where * stands for the conjugate.

The inner-product mostly used in finite quantum systems is:

$$\langle x|y\rangle = x_1^* y_1 + ... + x_n^* y_n. \tag{122}$$

where $x = (x_1, ..., x_n)$, $y = (y_1, ..., y_n) \in H$.

The inner product induces a *vector norm*:

$$||x|| = \sqrt{\langle x|x\rangle}. \tag{123}$$

Unit norm vectors of an inner-product space are also called **(pure) states** of *H*. Pure states of quantum systems are said to be vectors of a Hilbert space.

Definition 15 *An inner-product space H is called **complete**, if for any sequence* $\{\phi_i\}_{i=1}^{\infty}$, *with* $\phi_i \in H$ *and there is a vector* $\phi \in H$ *with the property that* $lim_{i\to\infty} \|\phi - \phi_i\| = 0$. *A complete inner-product space is also called a **Hilbert space**. Two Hilbert spaces are said to be **isomorphic**, notation* $H_1 \cong H_2$, *if the underlying vector spaces are isomorphic and their isomorphism preserves the inner-product.*

You can find this definition in [22].

The concepts of an orthogonal basis and of an orthogonal decomposition of a Hilbert space are fundamental for the Hilbert space theory.

Definition 16 *Two vectors* ϕ *and* ψ *of a Hilbert space are called **orthogonal,** notation* $\phi \perp \psi$, *if* $\langle\phi|\psi\rangle = 0$. *A set* $S \subseteq H$ *is orthogonal if any two distinct elements of* S *are orthogonal. S is **orthonormal** if it is orthogonal and all its elements have norm 1.*

If $E = \{e_1, ..., e_n\}$ is an orthonormal basis of H, then we can represent each vector of H by $x = x_1e_1 + ... + x_ne_n$, using this fixed basis axes, the vector x can be represented as $x = (x_1, ..., x_n)$.

Definition 17 *A **subspace** G of an inner-product space H is a subset of* H *closed under addition and scalar multiplication.*

An important property of Hilbert spaces is their decomposability into mutually orthogonal subspaces. It holds [23]:

Theorem 11 *For each closed subspace W of a Hilbert space H there exists a unique subspace* $W^{\perp}$ *such that* $\langle\phi|\psi\rangle = 0$, *whenever* $\phi \in W$ *and* $\psi \in W^{\perp}$ *and each* $\psi' \in H$ *can be uniquely expressed in the form* $\psi' = \phi_1 + \phi_2$, *with* $\phi_1 \in W$ *and* $\phi_2 \in W^{\perp}$. *In such a case we write* $H = W \oplus W^{\perp}$ *and we say that* W *and* $W^{\perp}$ *form an orthogonal decomposition of H.*

The symbol $\oplus$ is the Tensor product, defined in a moment.

We can make a generalization of an orthogonal decomposition

$$H = W_1 \oplus W_2 \oplus ... \oplus W_n, \tag{124}$$

of H into mutually orthogonal subspaces W_1,...,W_n, such that each $\psi \in H$ has a unique representation as $\psi = \phi_1 + \phi_2 + ... + \phi_n$, with $\phi_i \in W_i 1 \leq i \leq n$.

A very used method to construct orthonormal basis is called **Gram-Schmidt** procedure. Suppose we have an orthogonal basis set $|w_1\rangle$,...,$|w_d\rangle$ belonging to

a vector space V. To construct an orthonormal basis set $|v_1\rangle$,...,$|v_d\rangle$ for V we proceed as follows. Define $|v_1\rangle = \frac{|w_1\rangle}{|||w_1\rangle||}$. For $1 \leq k \leq d-1$ define

$$|v_{k+1}\rangle \equiv \frac{|w_{k+1}\rangle - \sum_{i=0} k \langle v_i|w_{k+1}\rangle |v_i\rangle}{|||w_{k+1}\rangle - \sum_{i=0} k \langle v_i|w_{k+1}\rangle |v_i\rangle||} \tag{125}$$

Tensor product

The so-called **Tensor product** , or *Kronecker product*, of H_1 and H_2, is written as

$$H = H_1 \otimes H_2. \tag{126}$$

The tensor product for vectors $x = (x_1, ..., x_m)$ and $y = (y_1, ..., y_n)$ is

$$x \otimes y = (x_1y_1, ..., x_1y_n, x_2y_1, ..., x_2y_n, ..., x_my_1, ...x_my_n) \tag{127}$$

and for matrices

$$A = \begin{bmatrix} a_{11} & ... & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & ... & a_{nn} \end{bmatrix} B = \begin{bmatrix} b_{11} & ... & b_{1n} \\ \vdots & & \vdots \\ b_{n1} & ... & b_{nn} \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & ... & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & ... & a_{nn}B \end{bmatrix} \tag{128}$$

References

[1] P. Shor, ”Algorithms for quantum computation: discrete logarithms and factoring,” in *Proc. 35th Annual Symposium on Foundations of Computer Science, Edited by S. Goldwasser, IEEE Computer Society Press*, Los Alamitos, 1994, p.124. – later revised and expanded as ”Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, *SIAM Journal of Computing*, 26, 1484, 1997.

[2] D. Stinson, ”Cryptography: theory and practice”, *CRC Press LLC*, 1995.

[3] H. Kosaka, A. Tomita, *et al.*, ”Single-photon interference experiment over 100 km for quantum cryptography system using a balanced gated-mode photon detector,” Electronically available at quant-ph/0306066.

[4] T. Cover and J. Thomas, ”Elements of Information Theory”, *Wiley series in telecommunications*, 1991.

[5] B. Schumacher, ”Quantum coding”, *Phys. Rev. A*, Vol. 51, number 4, 1995.

[6] K. Boström and T. Felbinger, ”Lossless quantum data compression and variable-lenght coding”, *Phys. Rev. A* 65, 032313, 2002. Electronically available in quant-ph/0105026.

[7] R. Jozsa and S. Presnell, ”Universal quantum information compression and degree of prior knowledge”. A preliminary version of this work was presented at EQIS ’02, Tokyo, September 2002. Electronically available at quant-ph/0210196.

[8] R. Jozsa, M. Horodecki, R. Horodecki and P. Horodecki ”Universal Quantum Information Compression”, *Phys. Rev. Lett.*, Vol. 21, number 8, 1714-1717, 1998.

[9] M. Hayashi and K. Matsumoto, ”Simple construction of quantum universal variable-length source coding”, 2002. Electronically available at quant-ph/0209124.

[10] S. Braunstein *et al.*, ”A quantum analog of Huffman coding”, *IEEE Transactions in Information Theory 46, 1644 - 1649* 2001. Electronically available at quant-ph/9805080.

[11] I. Chuang and S. Modha, ”Reversible Arithmetic Coding for Quantum Data Compression”, *IEEE Trans. Inform. Theory*, Vol. 46, No 3, pp. 1104-1116, 2000.

[12] M. Hayashi and K. Matsumoto, "Quantum universal variable-length source coding", *Phys. Rev. A* 66 022311, 2002. Electronically available at quant-ph/0202001.

[13] R. Cleve and D. DiVincenzo, "Schumacher's quantum data compression as a quantum computation," *Phys. Rev. A*, Vol. 54, pp. 2636-2650, 1996.

[14] C. Shannon, "A Mathematical Theory of Communication," *Urbana, III.: University of Illinois Press*, 1963. Reprinted in *C. E. Shannon and Warren Weaver (eds.)*.

[15] D. Huffman, "A method for the construction of minimun redundancy codes", *Preceedings IRE*, Vol.40, pp. 1098-1101, 1952.

[16] J. Ziv and A. Lempel, "A Universal algorithm for data compression," *IEEE Trans. Inform. Theory*, Vol. 23, pp. 337-343, 1977.

[17] R. Pasco, "Source coding algorithms for fast data compression", Ph.D. dissertation, Dep. Elec, Eng., Stanford Univ., Stanford, CA, 1976.

[18] J. Rissanen, "Generalized Kraft inequality and arithmetic coding", *IBM, J. Res. Devel.*, Vol. 20, no. 3, pp. 198-203, 1976.

[19] T. Cover, "Enumerative source encoding", *IEEE Trans. Inform. Theory*, Vol. 19, pp. 73-77, 1973.

[20] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding." *IEEE Trans. Inform. Theory*, Vol. 24, pp. 530-536, 1978.

[21] J. Brown, "The Quest for the quantum computer", *Simon & Schuster*, 2000.

[22] S. Lang, "Linear Algebra (Undergraduate Texts in Mathematics)", *Springer Verlag*, 3^{rd} edition, 1996.

[23] M Nielsen and I. Chuang, "Quantum Computation and Quantum Information", *Cambridge University Press*, 2000.

[24] C. Tannoudji, *et al.*, "Quantum Mechanics", *John Wiley & Sons*, vol. 1, 1992.

[25] W. Wootters, Wojciech H. Zurek, "A single quantum cannot be cloned", *Nature* 299, 802-803, 1982.

[26] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," Proc. R. Soc. Lond. A 400, 97 (1985).

[27] L. Grover, ”A fast quantum mechanical algorithm for database search,” *Proc. 28th Annual ACM Symposium on the Theory of computing*, ACM Press, New York, p.212, 1996.

[28] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp: ”Tight bound on quantum searching”, *Fourth Workshop on Physics and Computation - PhysComp’96*, Ed.: Toffoli, M. Biaford, J. Lean, New England Complex Systems Institute, 36-43 (1996). Electronically available at quant-ph/9605034.

[29] M. Hirvensalo, ”Quantum Computing”, *Springer*, 2001.

[30] G. Brassard *et al.*, ”Quantum Amplitude Amplification and Estimation”, 2000. To appear in AMS Contemporary Mathematics Series Millennium Volume entitled ”Quantum Computation & Information”. Electronically available at quant-ph/0005055.

[31] G. Brassard, P. Hoyer *et al.*, ”Quantum counting, Automata, Languages and Programming”, *Proceeding of the 25th International Colloquium, ICALP’98*, Lecture Notes in Computer Science 1443, 820-831, Springer (1998). Electronically available at quanth-ph/9805082.

[32] C. Dürr and P. Hoyer: ”A quantum algorithm for finding the minimum”. Electronically available at quant-ph/9607014.

[33] C. Bennett, *Quantum information*, *Physica Scripta*, T76:210-217, 1998a.

[34] R. Jozsa, ”Illustrating the concept of quantum information”, *IBM J. Res. Dev*, 2003. Article for Charles H. Bennett 60th Birthday Symposium. Electronically available at quant-ph/0305114.

[35] von Neumann, ”Mathematical Foundations of Quantum Mechanics”, English translation by R. Beyer (Princeton University Press, Princeton, NJ, 1955).

[36] A. Wehrl, *Rev. Mod. Phys.*, 50, 221, 1978.

[37] B. Schumacher and M. Westmoreland, ”Indeterminate-length quantum coding”, *Physical Review A* 64, 2304-2316, 2001. Electronically available at quant-ph/0011014.

[38] D. Montgomery and G. Runger, ”Applied Statistics and Probability for Engineers”, *John Wiley & Sons, Inc.*, 1994.