ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

BACHELOR PROJECT

An Exploration of Quantum Satisfiability

Author:
Julien PERRENOUD

Supervisor:
Pr. Nicolas MACRIS

June, 2015

Abstract

Satisfiability is a widely studied problem of theoretical computer science, where one is given a set of constraints on a set of bits and asked if it is possible to satisfy all of them at once. This problem, due to its difficulty and close relation with the physical world, has become one of the cornerstone of classical complexity theory. In this study, I explore various interpretations of this problem in the frame of quantum computing, with the hope of finding similar properties. After providing some context on classical and quantum complexity theory, I investigate two quantum counterparts to this problem that use local hamiltonian interactions and quantum projectors, and discuss their role in the complexity hierarchy. Finally, I take a closer look at random quantum satisfiability with a study of random graphs and phase transitions.

Acknowledgements

I would like to deeply thank Pr. Nicolas Macris for all his passionate explanations and remarks about a lot of intriguing physical phenomena that led me among a lot of research ; for giving me the opportunity to work with him on such an interesting topic ; for the time he spent on reviewing all of this work ; and for contenting my curiosity about quantum computing in his third year course, without which this project wouldn't have happened.

Contents

Introduction **5**

1 Complexity Theory **6**
1.1 Classical Complexity Classes . . . 6
1.1.1 P and NP . . . 6
1.1.2 Probabilistic Algorithms . . . 8
1.1.3 Merlin-Arthur Protocol . . . 9
1.2 Quantum Computing . . . 9
1.2.1 Quantum Circuits . . . 9
1.2.2 Promise problems . . . 11
1.3 Quantum Complexity Classes . . . 11
1.3.1 BQP . . . 11
1.3.2 Quantum Merlin-Arthur . . . 12
1.4 The Complexity Hierarchy . . . 13
1.4.1 Hardness and Completeness . . . 13
1.4.2 Our Hierarchy . . . 14

2 The Classical SAT **16**
2.1 Boolean clauses & CNF formulas . . . 16
2.2 SAT . . . 17
2.3 The Importance of SAT . . . 17
2.3.1 In Artificial Intelligence . . . 17
2.3.2 In Cryptanalysis . . . 17
2.4 Complexity . . . 18
2.5 Variant : MAXSAT . . . 21
2.5.1 Definition . . . 21
2.5.2 Complexity . . . 22

3 A Quantum Analogue for SAT **23**
3.1 Quantum Satisfiability in Nature . . . 23
3.2 Projectors & Hamiltonians . . . 24
3.3 Quantum SAT . . . 25
3.3.1 Definition . . . 25
3.3.2 Product and entangled states . . . 25
3.4 The Local Hamiltonian Problem . . . 25
3.4.1 Definition . . . 26
3.4.2 Feynmann's Hamiltonian Computer . . . 26
3.5 Complexity . . . 28
3.5.1 Kitaev's Clock Construction . . . 28
3.5.2 Kempe and Regev on LH . . . 31
3.5.3 About 1LH . . . 32
3.5.4 Bravyi on QSAT . . . 32
3.5.5 Gosset and Nagaj on QSAT . . . 33

4 Random QSAT **34**
4.1 Introduction . . . 34
4.2 The Geometrization Theorem . . . 34
4.3 Erdös-Rényi Graphs . . . 35
4.3.1 Two Models . . . 35
4.3.2 Properties . . . 36
4.3.3 Apparition of Subgraphs . . . 36
4.3.4 The Giant Component . . . 38
4.3.5 The Hypercore . . . 39
4.4 Random 2QSAT . . . 39
4.4.1 Bravyi's Transfer Matrix . . . 39
4.4.2 Trees are PRODSAT . . . 41
4.4.3 Loops are PRODSAT . . . 43
4.4.4 Figure Eights are not PRODSAT . . . 44
4.4.5 Phase Transition for 2QSAT . . . 45
4.5 Random 3QSAT . . . 47
4.5.1 Upper Bounds on Satifiability . . . 47
4.5.2 Lower bounds on Unsatisfiability . . . 47
4.5.3 Product Versus Entangled . . . 47

Conclusion **48**

References **50**

Appendix **51**
A Quantum Computing . . . 51
Introduction . . . 51
Linear Algebra . . . 51
Dirac Notation . . . 53
Quantum Bit . . . 53
Composition . . . 54
Unitary Operators . . . 55
Measurements . . . 57
B Rank and Submatrices theorem . . . 58
C Probabilities . . . 58
D Stirling's Formula . . . 60
E Graph Properties . . . 60
F Hypergraphs . . . 61

Foreword

Since the dawn of the digital age and the birth of computer science, the speed and capacity of computers have grown exponentially, from room-sized machines with few bytes of memory to super tiny microprocessors able to treat billions of operations every second. However, this rapid growth comes in contrast with the more and more accepted idea that some problems will forever remain *too hard* for a computer to solve.

By too hard, I don't mean that it is not possible to find a solution, but that the resources required *explode* within a little growth of the input. The main topic of this study, the satisfiability problem, appears to be one of these problems which raises the fundamental question of the limitation, in terms of computing power, of the classical bit-oriented computer. This questioning is even strengthened by the fact that such a problem appears to be everywhere in nature. Already in 1981, Richard P. Feynmann, one of the most influential physicists of the second-half of the 20th century, asked the following question :

"Can physics be simulated by a universal computer ?"

One of his major arguments involve the apparent continuity of space and time. As powerful as a computer can get, it will always contain a *finite* number of bits, and thus is forever reduced to only *approximate* nature. Some people have in fact taken this idea even further, with theories such as claiming that if we ever find the last digit of π, then it would probably mean that we are in some sort of Matrix-like computer simulation.

Another important matter is the existence of probabilities. As you might know, a classical computer cannot produce a random number, it can only provide the result of a function that *appears* random according to the input, but which is in fact totally determined from the very beginning. Similarly, if one tosses a coin and looks at the result, it seems random, but only provided that the initial position and speed are not known, in which case you can predict accurately if it is going to fall heads or tails.

Nature however, as far as we know, is driven by quantum mechanics, and it comes with this particularity that there exist phenomena which are *intrinsically* random. It means that it does not depend on the ignorance of the observer anymore, and one can thus never hope to predict the result before it has been observed.

Among others, these arguments will lead Feynmann to provide a definitive negative answer to the above question, which then raises an even more important one.

"How does nature calculate ?"

Indeed, if nature seems to have no trouble solving these problems while classical computers are backed into a corner, then there must exist, somewhere in nature, a superior computing power. This is precisely the role of Quantum Information Theory to analyse the power of quantum interactions as a computing resource, and in this study I will explore and compare their efficiency at solving these difficult tasks I described earlier.

1 Complexity Theory

Complexity theory is a subdiscipline of theoretical computer sciences that aims at classifying various problems[1] according to their *difficulty*. This term being quite wide, it is expressed in mathematical terms as the amount of *time* (time complexity) or *space* (space complexity) that a computer requires in order to solve the problem, according to the length of the given input.

Instead of the common word *computer*, which possess a sense that is far too large for any mathematical purpose, complexity theory defines various models[2] of theoretical computational machines, among which one can find the popular *Turing Machines* − deterministic or not − and *Boolean Circuits*. If the reader is not familiar with these notions, as well as the proper meaning of an *algorithm*, I recommend reading [10] before jumping into this chapter.

1.1 Classical Complexity Classes

As an introduction and in order to help transition into quantum algorithms, we will first present the main classical complexity classes, before trying to understand their quantum equivalent. Note as well that we will here only talk about time complexity, and what is called *decision problems* − We are given an input from an universal alphabet Σ^* (we often use $\Sigma = \mathbb{B} = \{0, 1\}$ because of how actual computers work) and want to decide if it belongs to a subset L defined by the problem. For example,

1. The problem of finding if a given number is prime is defined by the language

$$L_{even} = \{2, 3, 5, 7, \dots\}$$

2. Deciding if a $n \times n$ matrix has a non-zero determinant yields

$$L_{det} = \{A \in \mathcal{M}_{n\times n} |\ \det A \neq 0\}$$

Before we begin, please note that since the computational power of boolean circuits is equivalent to those of Turing machines (complete proof can be found in [10]), I chose to express in this chapter each class in terms of the latter, because it more convenient to do so.

1.1.1 P and NP

Beginning with P and NP seems like an obvious choice, because they are the "superstars" and foundation of complexity theory. Indeed, these two classes encompass a lot of everyday problems and are really easy to define. The first one, P, contains every problem that can be solved by a Turing machine in polynomial time.

Definition 1 (P)**.**
A decision problem $L \subset \Sigma^*$ belongs to P if there exists a deterministic Turing machine M such that

(i) M runs in polynomial time on all inputs.

(ii) For all $x \in L$, M accepts x.

[1] tasks that can be given to a computer, thus solvable with an algorithm

[2] A brief introduction about other models can be found on http://zoo.cs.yale.edu/classes/cs460/Spring98/chap1/machine.html

(iii) For all $x \notin L$, M rejects x.

Remark 1. *"Accepts" and "Rejects" usually means that the Turing Machine outputs* 1 *or* 0 *respectively.*

Remark 2. *"Polynomial time" will always refer to a number of steps that is polynomial in* $n = |x|$.

The idea behind P is that it contains all problems we can define as *easy*[3] such as

1. Deciding if a number is prime
2. Testing if a number is the minimum in a given list
3. Finding if a graph contains an Eulerian path[4]

That being said, the polynomial bound seems rather arbitrary, and it is. Why should we consider a problem that runs in n^{100} easier than one that runs in 1.1^n ? There is no perfect answer to this question, and a lot of mathematicians have expressed their opinion about the absurdity of such a classification. However, a polynomial bound expresses the idea that if we increase the size of a problem, then the difficulty of the problem should grow accordingly, and not explode in time like an exponential growth would.

Now that our idea of an easy problem is set, we will call *hard* [5] problems for which no polynomial-time solving algorithm is known, but that can be *verified* in polynomial time if a solution is provided. Such problems are contained in the complexity class NP.

Definition 2 (NP : Certificate Definition)**.**
A decision problem $L \subset \Sigma^*$ is in NP if there exists a deterministic Turing Machine M such that

(i) M runs in polynomial time on all inputs.

(ii) For all $x \in L$, there exists a certificate $c \in \mathbb{B}^{poly(n)}$ such that M accepts (x, c).

(iii) For all $x \notin L$, M rejects (x, c) for any c provided.

The reader might want to note that NP refers to "Non-deterministic Polynomial time" instead of "Non-Polynomial" as one might think at first thought. This is due to another equivalent definition of NP using non-deterministic Turing machines[6], which I will state here as well.

Definition 3 (NP : Non-Deterministic TM Definition)**.**
A problem $L \subset \Sigma^*$ is in NP if there exists a non-deterministic Turing Machine M such that

(i) M runs in polynomial time on all inputs.

(ii) For all $x \in L$, M accepts x.

(iii) For all $x \notin L$, M rejects x.

Some examples of these problems are

[3] *efficiently solvable* or *tractable* in mathematical terms
[4] a path that uses every edge
[5] *intractable*
[6] if you don't know what a non-deterministic Turing machine is, either read [10] or consider it as being a Turing machine that can explore every computational path at the same time

1. The classical satisfiability problem (which I'll explain in the next section).
2. The Travelling Salesman problem.
3. Finding if a graph contains an Hamiltonian path.[7]

1.1.2 Probabilistic Algorithms

Another interesting chapter of complexity theory, and which will seem to be more appropriate when it comes to translating it into quantum complexity, relates to the study probabilistic algorithms.

The thought behind probabilistic algorithms is that, in practice, there is no such thing as an *exact* answer. A computer, as efficient as it can be, will always have a small risk of having a bug somewhere that leads to an erroneous answer. Moreover, there is no need to have a exact solution at every try − providing that the probability of having a correct answer is high enough, we only need to run the algorithm multiple time to have an answer that is *almost certainly* true.

This behaviour can be translated into Turing machines by considering that, in addition to the input x, the machine provides a random number y that will be used for the computations. This type of Turing Machine is often referred to as *Probabilistic Turing Machine*

Definition 4 (BPP)**.**
A language $L \subset \Sigma^*$ belongs to BPP if there exists a polynomial p and a Probabilistic Turing Machine M such that

(i) M runs in polynomial time on all inputs.

(ii) For all $x \in L$,

$$\Pr_{y \in \Sigma^{p(|x|)}}[M \text{ accepts } (x, y)] \geq \frac{2}{3}$$

(iii) For all $x \notin L$,

$$\Pr_{y \in \Sigma^{p(|x|)}}[M \text{ accepts } (x, y)] \leq \frac{1}{3}$$

Remark 3. *Property (ii) is often called Completeness − the machine accepts good inputs with recognizable probability − while property (iii) is called Soundness − the machine doesn't accept bad inputs.*

Remark 4. *The bound $\frac{2}{3}$ (resp. $\frac{1}{3}$) is completely arbitrary. In fact, the definition is equivalent for any value greater than $\frac{1}{2} + \varepsilon$ (resp. less than $\frac{1}{2} - \varepsilon$), where $\varepsilon = \frac{1}{2^c}$ for a constant c. Basically, it means that when we can get exponentially close to a probability of 1 (resp. 0) by simply repeating the experiment a polynomial number of times.*

Remark 5. *Property (iii) is equivalent to*

$$\Pr_{y \in \Sigma^{p(|x|)}}[M \text{ rejects } (x, y)] \geq \frac{2}{3}$$

[7] a path that passes through each vertex only once

1.1.3 Merlin-Arthur Protocol

In order to conclude this very brief journey into the world of classical computation, I will take a look at what is called *Arthur-Merlin protocols*, introduced in 1985 by László Babai and Shlomo Moran in [1]. They yield interesting results while using a different approach on complexity and will have a great role to play in the study of quantum complexity.

The setup is as follows : Arthur, being a simple human, can only make calculations in polynomial time. He does however possess golden coins in his pocket and therefore is allowed to use random numbers. On the other side, the almighty oracle Merlin can transcend time and thus owns an infinite computational power. This gives birth to a system of interactive proofs where Merlin's goal is to convince Arthur of something, knowing that Arthur is no fool and do not trust Merlin. Since Merlin might lie, he will then need to analyse any information he receives.

The nature of the complexity class results from the number of messages exchanged between Arthur and Merlin. The only one we will look at here is called MA, and restricts the interaction between Arthur and Merlin to a single message from the latter. Intuitively, MA contains all languages such that for every word in the language, there exists a polynomial proof that Merlin can send Arthur in order to convince him. On the other hand, there should be no way that Merlin convinces Arthur to accept a false proof.

Definition 5 (MA)**.**
A decision problem $L \subset \Sigma^*$ belongs to MA if there exists a probabilistic polynomial-time Turing Machine M (Arthur) and polynomials p, q such that

(i) *M runs in polynomial time on all inputs*

(ii) *For all $x \in L$, there exists a certificate $c \in \Sigma^{q(n)}$ such that*

$$\underset{y \in \Sigma^{p(|x|)}}{Pr}[M \text{ accepts } (x, y, c)] \geq \frac{2}{3}$$

(iii) *For all $x \notin L$, any $c \in \{0,1\}^{q(n)}$ is such that,*

$$\underset{y \in \Sigma^{p(|x|)}}{Pr}[M \text{ accepts } (x, y, c)] \leq \frac{1}{3}$$

Remark 6. *The class remains the same even if we require perfect completeness, i.e that Arthur always accepts with probability 1.*

1.2 Quantum Computing

Now that we have a relatively clear idea of what classical complexity looks like, let us go into the strange world of quantum mechanics. In this chapter, I will however not provide readers a complete introduction and I redirect unexperienced ones towards [10].

1.2.1 Quantum Circuits

A first task when one enters the world of quantum computation would be to try and define a quantum equivalent for the Turing Machine. However, such a computational model is very hard to work

with and contains a lot of assumptions that should be met in order to work properly. As a result, I will focus on a more natural way of defining quantum computation which imitates classical boolean circuits. More informations about quantum Turing machines can still be found in [10]. Note also that, as in the classical case, the model of Quantum Circuits has been proven to have the same computational power as that of quantum Turing machines by H. Nishimura and M. Ozawa in [15].

Quantum Circuits, as introduced by David Deutsch in 1985, are an intuitive counterpart of classical boolean circuits, where the gates operate on bits instead of qubits. The only difference is the unitary nature of quantum operators, which makes this model in fact closer to the that of reversible boolean circuits as imagined by Fredkin and Toffoli.

This section will only give a general overview of quantum circuits in order to understand further definitions. For a deeper explanation of quantum circuits, please refer to [10].

Definition 6 (Quantum Circuit)**.**
A quantum circuit $U = U_L \dots U_1$ transform an input state $|\psi_{in}\rangle$ into an output state $|\psi_{out}\rangle = U|\psi_{in}\rangle$ via a series of successive quantum gates $U_1, \dots, U_L$.

Remark 7. *L denotes the "length" or "depth" of the circuit, while the size of input, witness and ancilla states is called the "size".*

Remark 8. *Because of the reversibility of the systems, we often need to "borrow" extra qubits in the process, initialized in the $|0\rangle$ state. These qubits, often denoted as "ancillas" qubits, will be considered as part of the input in our definition.*

Remark 9. *Because we can always use extra gates at the beginning of the circuit, some definitions require every input qubit to be in state $|0\rangle$. In this study, it is sometimes simpler to think of a non-zero input state so I'll keep the definition above, although it is not hard to see that these notions are equivalent here.*

Given this definition and provided the random nature of quantum observations, we still need to determine what *accepting an input* means. This is defined rather intuitively with the following acceptance probability.

Definition 7 (Acceptance Probability)**.**
Given a quantum circuit U that runs on input state $|\psi\rangle$, the acceptance probability of U is the probability of measuring $|1\rangle$ in the first bit of the output state.

$$Pr[U \text{ accepts } |\psi\rangle] = \langle\psi|U^\dagger\big(|1\rangle\langle 1| \otimes \mathbb{I}^{\otimes(n-1)}\big)U|\psi\rangle$$

Remark 10. *Some definitions require to have all bits of $|\psi_{out}\rangle$ in state $|1\rangle$. It is not hard to see that these definitions are equivalent.*

Remark 11. *In most cases, we will simply define our own "answer" qubit and define the acceptance probability as the probability of having it in state $|1\rangle$.*

Now before going into the next chapter, there is an important concept that remains. What exactly is a *quantum algorithm* ? Is it the circuit that performs the computation ? That is not a very good answer, because we would need to talk of different sized circuits as implementing different algorithms, though they all solve the same problem. Then what ? Well, I will here use a definition that was introduced by Kitaev in [10].

Definition 8 (Quantum Algorithm)**.**
A quantum algorithm for the computation of a function $f : \Sigma* \to \Sigma*$ is a classical algorithm (i.e. a Turing machine) that computes a function of the form $x \to D(U_x)$ where $D(U_x)$ is a description of a quantum circuit U_x that computes $f(x)$ on empty input[8]*.*

Remark 12. *According to Kitaev in [10], there is no difference if we decide to use a quantum algorithm to construct our circuit.*

1.2.2 Promise problems

In the classical case, we looked at decision problem which are defined as finding whether an element of Σ belongs to a related language L. In quantum computation, things are a bit different. Because of the random nature of computation, we often require a *promise* that allows us to distinguish efficiently the possible outputs. Such problems are defined in the following way.

Definition 9 (Promise Problem)**.**
A promise problem is associated two disjoint languages L_{yes} and L_{no}, where all inputs in L_{yes} should be accepted while rejecting those in L_{no}. The fact that the input always belongs in $L_{yes} \cup L_{no}$ constitutes what we call the promise of the problem.

1.3 Quantum Complexity Classes

Now that our arsenal is ready, let us see how we can use these previous definitions to provide an equivalent set of classes for quantum algorithms.

1.3.1 BQP

An expected first result when trying to translate classical classes into quantum classes is that, due to the intrinsic random nature of quantum measurements, P and BPP will know no difference in the quantum world. The resulting class, called BQP, has a similar definition and meaning as BPP and stands for "Bounded-error Quantum Polynomial-time".

Definition 10 (BQP)**.**
A promise problem $L_{yes} \cup L_{no} \in \Sigma^*$ belongs to BQP if there exists a polynomial-time quantum algorithm such that

- *For all $x \in L_{yes}$,*
$$Pr[U_x \text{ accepts } |0\rangle] \geq \frac{2}{3}$$
- *For all $x \in L_{no}$,*
$$Pr[U_x \text{ accepts } |0\rangle] \leq \frac{1}{3}$$

Remark 13. *The discussion around $\frac{2}{3}$ and $\frac{1}{3}$ is the same as in BPP*

For all we know at this point, the behaviour of BQP among the complexity classes is the same as its classical counterpart. It contains most of the famous quantum algorithms such as

- Shor's algorithm for integer factorization that runs in $O(n^3)$.
- The discrete logarithm problem.
- Feynmann's problem of the simulation of quantum systems as discussed in the introduction.

[8] with input state $|0, \ldots, 0\rangle$

1.3.2 Quantum Merlin-Arthur

Now that we have an equivalent class for P, the next step is to find the quantum counterpart of NP. This was first done by Kitaev in [10], which introduced the complexity class BQNP. Several years of research later, mathematicians came to the conclusion that it was in fact equivalent to the quantum equivalent of MA, and that it made more sense to call it QMA. The idea is pretty much the same as with classical Merlin-Arthur protocols, but with quantum circuits instead of Turing machines.

Definition 11 (QMA)**.**
A promise problem $L_{yes} \cup L_{no} \in \Sigma^*$ belongs to QMA if there exists a polynomial-time quantum algorithm such that

(i) For all $x \in L_{yes}$, there exists a certificate state $|\psi_c\rangle$ such that

$$Pr[U_x \text{ accepts } |\psi_c, 0\rangle] \geq \frac{2}{3}$$

(ii) For all $x \in L_{no}$ and any certificate $|\psi_c\rangle$,

$$Pr[U_x \text{ accepts } |\psi_c, 0\rangle] \leq \frac{1}{3}$$

Finally, there is also another class of complexity which we'll use in upcoming chapters. It corresponds to the class of Quantum Merlin Arthur protocols with one-sided error, denoted as QMA_1.

Definition 12 (QMA_1)**.**
A promise problem $L_{yes} \cup L_{no} \in \Sigma^*$ belongs to QMA_1 if there exists a polynomial-time quantum algorithm such that

(i) For all $x \in L_{yes}$, there exists a certificate state $|\psi_c\rangle$ such that

$$Pr[U_x \text{ accepts } |\psi_c, 0\rangle] = 1$$

(ii) For all $x \in L_{no}$ and any certificate $|\psi_c\rangle$,

$$Pr[U_x \text{ accepts } |\psi_c, 0\rangle] \leq \frac{1}{3}$$

(iii) Every U_x is built from a fixed universal gate set $\mathcal{G}$.

Remark 14. *An important point here is that in the classical case, there is no distinction between MA and what would be MA_1, whereas in the quantum world this distinction has to be made since there is no proof that these classes are equivalent.*

Remark 15. *Condition (iii) appears from technical issues when trying to have an exact measurement for point (i). There is at this moment no reason to think that the choice of $\mathcal{G}$ has an impact on the resulting class.*

1.4 The Complexity Hierarchy

1.4.1 Hardness and Completeness

After having defined all these complexity classes, an important point would be to organize them and thus define exactly what it means for a complexity class to contain problems that are *harder* than those of another. It is mathematically and intuitively formulated by the notions of *hardness* and *completeness* with respect to the so-called *Karp reduction*, which I will all recall here.

The idea behind Karp reduction is that if a problem can, in polynomial time, be transformed into[9] another problem, then it must be *at most as hard*, since a solution for the latter will provide a solution for the first. This is nicely translated into the following definition

Definition 13 (Karp reduction)**.**
A Karp reduction from a problem L_1 to a problem L_2 is a function f such that

(i) f *is computable in polynomial-time for all inputs*

(ii) f *produces instances of L_2 when given instances of L_1*

(iii) $w_1 \in L_1 \Leftrightarrow f(w_2) \in L_2$

If such a function exists, we will say that L_1 can be reduced to L_2 (in polynomial time).

Notation 1. *It will be denoted as $L_1 \leq_p L_2$.*

Considering this relation, our goal now is to find problems that can represent a given class, in the sense that every problem in the class can be reduced to the representative. Such a property is denoted as hardness and is mathematically formulated in the following way.

Definition 14 (C-hard)**.**
Given a complexity class C and a decision problem L, L is said to be C-hard (w.r.t the Karp reduction) if

$$L \leq_p L' \quad \forall L' \in C$$

Finally, if the problem is both contained in C and C-hard, then we call it C-complete.

Definition 15 (C-complete)**.**
Given a complexity class C and a decision problem L, L is said to be C-complete (w.r.t. the Karp reduction) if

(i) $L \in C$

(ii) L *is C-hard*

The notion of completeness is very strong. Basically, it means that if you can provide an algorithm that solves a C-complete language in a way that it now belongs to another complexity class D, automatically it means that all languages in C belong to D as well, and thus provide an evidence of the fact $C \subset D$. For this reason, complete problems are widely studied in modern complexity theory in order to try and collapse classes that appear to have different levels of complexity.

[9] *reduced* to

1.4.2 Our Hierarchy

Using the Karp reduction I just defined, I will try to give an overview of the hierarchy of the complexity classes that we just saw. It is important to keep in mind that a lot of relations are not known to this day, therefore some classes that are represented as strict inclusions may in fact well be equal. I will not provide any proof in this section because most of the arguments used are pretty much self-explanatory.

The first example, and the most famous, is the question of *P versus NP*. It is one of the most fundamental yet unsolved problem of theoretical computer science. Basically, it is trivial to see that

Theorem 1. $P \subset NP$

On the other hand, the converse has been resisting uncountably many attempts from thousands of mathematicians for the past 40 years. In fact, this problem is part of the 7 Millennium Prize Problems of the Clay Mathematics Institute[10], which are considered to be the hardest tasks of modern mathematics.

Apart from all that, another easy inclusion can be made when thinking of probabilistic algorithms. Every classical non-probabilistic algorithm can be realized by a probabilistic Turing machine with perfect soundness and completeness when one just ignores the random bits. Therefore,

Theorem 2. $P \subset BPP$

Finally, the question of BPP versus NP is also still open.

Turning our eyes onto Merlin and Arthur, we can first see that Arthur can always chose to ignore Merlin's advice and try to compute an answer on his own. This yields

Theorem 3. $BPP \subset MA$

On the other hand, Arthur can take the advice from Merlin while making no use of his random coins, therefore

Theorem 4. $NP \subset MA$

As for quantum classes. First of all, it is easy to see that a quantum circuit can always simulate a classical one if we only allow pure state without linear combinations. This gives the expect results

Theorem 5. $BPP \subset BQP$

and

Theorem 6. $MA \subset QMA$

Moreover, we can use the same argument as with BPP and MA, i.e. that Quantum Arthur can ignore the advice from Quantum Merlin and computes on his own, and convince ourselves that

Theorem 7. $BQP \subset QMA$

[10] More informations on http://www.claymath.org/millennium-problems

Finally, finding if MA and NP are contained in BQP is probably one of the most exciting question when one tries to grasp the power of quantum computing. Unfortunately, there is no definitive answer at this time. In fact, several algorithms that belong to NP can be realized with a polynomial-time quantum computing machine, but it is still unclear whether all NP problems can. More important than that, the implications of $NP \in BQP$ would mean that quantum computers can provide an exponential speed-up to *every* classical algorithm, which is today still only a hope.

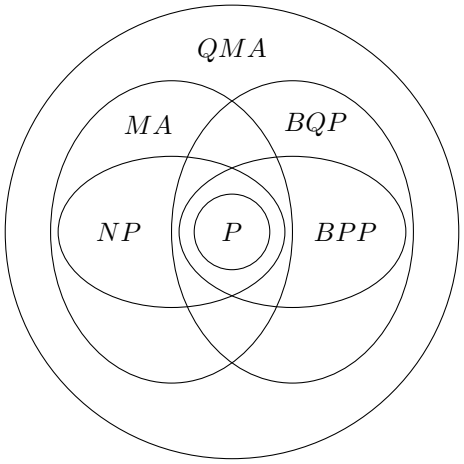

Figure 1: Our Complexity Hierarchy

2 The Classical SAT

In the previous chapter, I mentioned that some problems, denoted as NP-complete, were of major importance in theoretical computer science because they could be used to express every classical problem. The *Boolean Satisfiability Problem* (abbreviated SAT), is in fact one of them, and maybe the most important of all because it appears in a lot research areas. For this reason, it has been widely studied for the past 50 years, and still no polynomial algorithm has been found to this day.

The idea behind SAT is the following. Given any mathematical or physical object that can be represented as a state with multiple degrees of freedom[11], we want to add several *constraints* on this state and find out if there is a way for our state to satisfy all of them.

In this chapter, I will provide a clear mathematical definition of this problem, followed by a few examples of fields where it appears. Finally, I will discuss its place among the complexity hierarchy with the famous Cook-Levin theorem.

2.1 Boolean clauses & CNF formulas

In the classical version of SAT, degrees of freedoms are represented as a set of n boolean variables. From there, one can find multiple ways of defining a constraint, but the one that will be used here is the standard boolean *clause* representation. Clauses are nothing else but a special form of boolean functions, defined as follows.

Definition 16 (Clause)**.**
A clause is a disjunction of literals, i.e. of the form

$$\phi(x_1, ..., x_n) = \bigvee_{i=1}^{k} l_i$$

where each literal l_i represents the variable x_i or its negation $\overline{x}_i$.

Remark 16. *What is important to notice about a clause is that this form allows to think about the constraint in a very specific way : each clause will rule out one and only one possible state of the bits it acts upon. For instance, a clause of the form $x_1 \vee \overline{x}_2 \vee x_3$ will accept any state except $(x_1, x_2, x_3) = (0, 1, 0)$.*

In order to now be able to define multiple constraint $-$ and get a boolean function that takes the value 1 if all of them are satisfied $-$ we will use a conjunction[12] of clauses, which correspond to what is called the Conjunctive Normal Form of a boolean formula.

Definition 17 (CNF formula)**.**
A CNF (Conjunctive Normal Form) formula is a conjunction of clauses, i.e. of the form

$$\phi(x_1, ..., x_n) = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{i,j}$$

Remark 17. *If every clause contains at most k literals, we will talk about a kCNF formula.*

[11] A degree of freedom can be seen as a variable with multiple values
[12] A series of AND operators

2.2 SAT

Using all these notions together, a CNF formula $\phi(x_1, \ldots, x_n)$ will be denoted as *satisfiable* if and only if there exists an assignment of the variables $(a_1, \ldots, a_n)$ such that

$$\phi(a_1, \ldots, a_n) = 1$$

From there, the question of satisfiability is simply defined as the existence of such an assignment.

Definition 18 (SAT)**.**
In terms of input / output, SAT is defined as
Input : *a CNF formula ϕ acting on n bits*
Output : *Yes if ϕ is satisfiable, No otherwise.*

Remark 18. *In terms of language, we define $L_{SAT} = \{\phi$ a CNF formula $\mid \phi$ is satisfiable$\}$*

Remark 19 (kSAT)**.** *kSAT is an instance of SAT where all the clauses are kCNF formulas.*

2.3 The Importance of SAT

At this point, a question may arise. Why is this problem so important ? Well, the importance of SAT as a problem is a consequence of two things. First, a lot of different areas of mathematics and physics can be represented and solved as constraint satisfaction problems. Secondly, it is the first problem that was found to be NP-complete. In this chapter I will try to explain and give some insight about phenomena that can be represented as a simple satisfiability problem.

2.3.1 In Artificial Intelligence

The field of Artificial Intelligence is composed with a lot of different tasks that we want to find efficient algorithms for a robot to solve. On the opposite of *data*, which is usually understood as any form of ordered information, *knowledge* is disordered and is often represented as a set of logical formulas. Whether they represent conclusions that one might make from a situation − if it rains, then the floor is probably wet − or propositions that can not go together − I cannot be in San-Francisco and Los Angeles at the same time − these formulas can (almost) always be transformed CNF formulas.

A good example is the problem of *planification*, which consists, given an initial state and a set of possible operations, in finding a way to attain one or more *goals* through a clever ordering of actions. This type of task is typically really easy to transform into a satisfiability problem, which emphases the importance of finding an efficient algorithm to solve such problems.

2.3.2 In Cryptanalysis

The goal of everyday cryptography is to ensure that the communication between two end-points verify two properties : *confidentiality*[13] and *authenticity*[14]. In order to do that, modern cryptography is heavily based on mathematical assumptions about the computational hardness of problems and the use of so-called *one-way functions*[15]

[13] Nobody should be able to read the messages.
[14] One must make sure that the other person is really who he claims to be.
[15] functions that are efficiently computable in a way but intractable in the other. Their whole existence rely on the assumption P $\neq$ NP.

An everyday example of this is the widely used RSA algorithm, which is based on the fact that nobody knows how to efficiently factorize big numbers (although it might change when someone builds a big enough quantum computer that implements Shor's algorithm). RSA is typically used everytime you purchase something online, or log into a secure account on an https website.

In [13], Mironov and Zhang explain how some of the widely used cryptographic hash functions got broken in 2005. The process behind the attack was to encode the cyphers as boolean formulas and then use *SAT solvers* in order to find a possible solution, which leads to the secret key. After this attack, a lot of other typical cryptographic functions were tested and modern cryptanalysis now uses SAT solvers in order to determine the effective difficulty of computational problems they rely on.

2.4 Complexity

Another reason why SAT is such an important piece of theoretical computer science is its place among the complexity hierarchy. It was in fact the very first prototype of a NP-complete problem. The proof, discovered by Cook and Levin in 1971, is in my opinion quite a beautiful one and for this reason it will be provided below.

Now since 3SAT $\leq_p$ SAT (proof can be read in [5]), I will here only focus on providing a suitable complexity class for 2SAT and 3SAT.

Beginning with the first, the use of two-variable interactions only allows to fully determine the state of adjacent variables once a value is attributed. Using such a method, one can easily find a proper polynomial algorithm and classify 2SAT accordingly. Numerous proofs exist and many algorithms find a solution in faster time, but since we only care about asymptotic complexity I will provide a short and easy proof.

Theorem 8. *2-SAT* $\in$ *P*

Proof. The idea here is to construct a implication graph out of ϕ and to observe how the assignments propagates to the other variables. Given a formula $\phi(x_1, \ldots, x_n)$, we construct the following directed graph $G = (V, E)$.

1. For any variable and its negative, we create a vertex. thus $V = \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\}$.
2. Starting with $E = \emptyset$, we then add the directed edges $(\overline{a}, b)$ and $(a, \overline{b})$ into E for every clause $a \vee b$.

What it means, is that if we assign any value 0 or 1 to a vertex x_i, then we must propagate this value to all its neighbours in order to satisfy the constraints. We will thus iterate on every vertex and look for paths from x_i to $\overline{x}_i$ and vice-versa, using any graph search algorithm such as BFS or DFS.

If such a path exists, then it means that both x_i and $\overline{x}_i$ must take the same value, which yields a contradiction. On the other hand, if we cannot find any contradiction then we can simply take the value on every vertex as a satisfying assignment for ϕ.

Last but not least, we need to verify that all this takes polynomial time. The maximum number of constraint is given by $\frac{n(n-1)}{2} = O(n^2)$, and since we iterate on n variables performing two searches for each, the total time of this algorithm is $O(n^3)$, which is polynomial as requested. $\square$

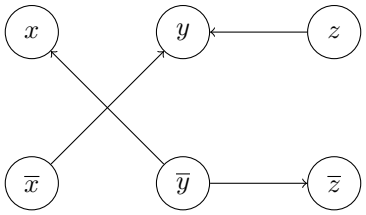

Figure 2: The implication graph for $(x \vee y) \wedge (y \vee \overline{z})$

Theorem 9 (Cook-Levin '71)**.** *3-SAT is NP-complete*

Proof. In order to prove the NP-completeness of 3-SAT, we need to provide two things, the fact that it belongs to NP and the NP-hardness. We will do this in two parts.

3SAT $\in$ NP

This part is easy. Given a satisfiable assignment $f : \{1, n\} \to \mathbb{B}$, we can simply affect every value to the corresponding apparitions of the variable, which takes $O(m)$ (where $m \leq \binom{n}{3}$ is the number of clauses), then iterate on each clause and verify that is yield true, which takes $O(m) = O(n^3)$ time again.

3SAT is NP-hard

The idea here is pretty simple and quite remarkable. For any language $L \in$ NP, there exists a non-deterministic Turing machine M that can decide it in n^c. Given M and the input w of the problem, what Cook and Levin did was provide a 3CNF formula ϕ with the following properties

1. If M accepts w, then ϕ is satisfiable.
2. If M doesn't accept w, then ϕ isn't.

Now if M accepts w, then it means that there must exist a computational path of M that leads to an accepting state. Thus their idea was to represent any path possible of M run with w as a *tableau* with the following layout. Each row represents a state of the tape at a given step, and the head of the Turing machine is the bold character at the right of the cell containing its current state.

q_0	$\mathbf{w}_1$	w_2	$\dots$	w_{n-1}	w_n	$\square$	$\square$	$\dots$	$\square$
w_1	q_1	$\mathbf{w}_2$	$\dots$	w_{n-1}	w_n	$\square$	$\square$	$\dots$	$\square$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
w_1	w_2	w_3	$\dots$	w_n	$s_{n,j}$	q_j	$\mathbf{s}_{n+2,j}$	$\dots$	$\square$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$

Figure 3: A possible tableau for M

Now, we say that a tableau T is an accepting tableau if it represents a valid series of states in the corresponding NTM and if the last row is an accepting configuration of M. Given the properties

of non-deterministic Turing machines and the one-to-one relation between a tableau and the corresponding computational path, it follows that

$$M \textit{ accepts } x \Leftrightarrow \exists \textit{ an accepting table } T \textit{ for } M$$

At this point, all we have to do left is find a 3CNF formula ϕ such that

$$\exists \textit{ an accepting table } T \textit{ for } M \Leftrightarrow \phi \textit{ is SAT}$$

First, let us define the variables of ϕ. Since our tableau T has n^{2c} entries, and the set of possible symbols on the tape is given by

$$S = Q \cup \Sigma \cup \{\square\}$$

it means that the whole state of T can be described by $|S|n^{2k} = O(n^{2c})$ variables $x_{i,j,s} \in \mathbb{B}$ such that

$$x_{i,j,s} = 1 \Leftrightarrow T[i,j] = s$$

Now that we have our variables, let us define the constraints. There are in total four type of constraints

1. $\phi_{in}-$ It aims at ensuring that the starting state of the TM is a proper starting state. Such a constraint can be translated into

$$\phi_{in} = x_{0,0,q_0} \wedge x_{0,1,w_1} \wedge \cdots \wedge x_{0,n,w_n} \wedge x_{1,n+1,\square} \wedge \cdots \wedge x_{1,n^c-1,\square}$$

2. $\phi_{out}-$ Same idea, but here we want to describe the fact that at the end of the computation, M is in an accepting state. Note that we can only check the last row here since if the accepting state comes before we can just tell M to do nothing for the rest of the computation.

$$\phi_{out} = \bigvee_{i=0}^{n^c-1} x_{i,n^c-1,q_{accept}}$$

3. $\phi_{mutex}-$ Here what we want to do is to forbids any state of our Turing machine where more than one symbol is written in the same cell.

$$\phi_{mutex} = \bigwedge_{1 \leq i,j \leq n^c} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\overline{x}_{i,j,s} \vee \overline{x}_{i,j,t}) \right) \right]$$

4. $\phi_{prop}-$ At this point, the only condition left to be checked is that the computational path is valid. In order to do this, we must ensure that every row $-$ which again represents a state of the TM $-$ logically follows from the precedent one, according to the rules of its transition function δ.

   In order to do that, what Cook and Levin did was divide the tableau into 2×3 *windows*, and use the following lemma, which I will not prove here.

   Lemma 1. *If the first row of the table is valid and every 2×3 window is a legal $-$ i.e. it respects the transition function $-$ then every row yields the next one in the computation of the TM.*

With this tool, all we need to do is provide a formula that goes over all windows and checks if they are legal.

$$\phi_{prop} = \bigwedge_{0 \leq i,j \leq n^c - 3} \textit{"window [i, j] is legal"}$$

In order to transform this into a real formula, we need a little trick. Since every legal or illegal window can be represented as a tuple $(a_1, \ldots, a_n)$, we can simply go over all legal combinations and check if one corresponds to our current window. This yields

$$\begin{aligned}\phi_{prop} &= \bigwedge_{0 \leq i,j \leq n^c - 3} \left[\bigvee_{(a_1,\ldots,a_n)\ \textit{legal}} \left(x_{i,j,a_1} \wedge \cdots \wedge x_{i+1,j+2,a_6} \right) \right] \\ &= \bigwedge_{0 \leq i,j \leq n^c - 3} \left[\bigwedge_{(a_1,\ldots,a_n)\ \textit{illegal}} \left(\overline{x}_{i,j,a_1} \vee \cdots \vee \overline{x}_{i+1,j+2,a_6} \right) \right]\end{aligned}$$

Having define all our constraints, all we need to do now is verify that the formula

$$\phi = \phi_{in} \wedge \phi_{out} \wedge \phi_{mutex} \wedge \phi_{prop}$$

can be transformed into a 3CNF formula with a polynomial number of clauses. Let's go over them one by one.

1. ϕ_{in} is already a 3CNF formula with $O(n^c)$ clauses.
2. ϕ_{out} can be easily transformed into 3CNF with additional variables using the following schema

   $$x_1 \vee x_2 \vee \cdots \vee x_r = (x_1 \vee x_2 \vee y_1) \wedge (\overline{y}_1 \vee x_3 \vee y_2) \wedge \cdots \wedge (\overline{y}_{r-2} \vee x_{r-1} \vee x_{r-2})$$

   Using this, the number of additional variable in ϕ_{out} is asymptotically equal to the number of clauses, $O(n^c)$.
3. ϕ_{mutex} contains $O(n^{2c})$ times the outer bracket, which can be reduced to a constant number of constraints using the same method as for ϕ_{out}. Thus $O(n^{2c})$ in total.
4. ϕ_{prop} is a little bit harder to see, but since the windows are of constant size, we can conclude that the inner AND contains a constant number of terms. Since the outer AND iterates on every possible window, we can see that once again the number of constraints is $O(n^{2c})$

Thus the total number of clauses in ϕ is $O(n^{2c}) = poly(n)$ which completes the proof. □

2.5 Variant : MAXSAT

2.5.1 Definition

Before moving on to the next chapter, I want to quickly discuss a variant of SAT that will be important in the upcoming matter. The principle here stays the same, but more than simply looking for a solution that satisfies all the constraints, we want to find one that *minimizes* the number of non-respected ones.

Definition 19 (MAXSAT)**.**
In terms of input/output, MAXSAT is defined as
Input : *a CNF formula ϕ*
Output : *an asignment f that satisfies the largest number of clauses in ϕ*

2.5.2 Complexity

What is interesting is that such a little tweak in the definition lead to a different complexity classification. In fact, while we have seen that SAT was NP-complete for $k = 3$, we have here the following result

Theorem 10. *MAX2SAT is NP-complete*

I will not provide the proof because it is not of interest here, but the reason for such a distinction between SAT and MAXSAT lies in the fact that MAXSAT is an optimization problem. In fact, we cannot apply a reduction algorithm anymore (like it can be done with 2SAT) simply because we cannot ensure the fact that the outcoming assignment will be the one that minimizes the number of unsatisfied clauses.

3 A Quantum Analogue for SAT

Having set the frame of theoretical computer science and quantum computing and analysed classical satisfiability, it is now time to turn our eyes on the main topic of this work, which is the study of some quantum equivalent for the satisfiability problem. As we will see, this problem has already captured the interest of a lot of computer scientists and physicists, and has the same property than its classical counterpart when it comes to be closely related to a lot of various areas of research. However, the problem is has not yet revealed all his secrets.

In this chapter, I will first explain the motivation behind such a study and what a quantum constraint looks like. Then, I will explore different definitions that were made by different physicists, and I will end by exploring and summarizing several articles about the place of such a problem among the complexity hierarchy.

3.1 Quantum Satisfiability in Nature

Quantum satisfiability, in its most natural form, is a phenomenon that appears in multiple fields of physics. It is therefore a nice place to start when trying to grasp the computational power of quantum interactions. Upon the most famous applications, one can mention the study of the states of matter, and the properties of magnets.

The magnetic or non-magnetic property of a given piece of matter comes from the alignment or disorder of huge quantities of *tiny magnets*. In fact, every particle possess a tiny magnetic moment[16], which is called the *spin*. The spin is typically a quantum property, as we can think of it being a superposition of two states[17], *up* $|\uparrow\rangle$ and *down* $|\downarrow\rangle$.

Having said that, the spin of each particle will be influenced by the environment and the neighbouring particles. In order to try and visualize this, try putting two magnets next to each other. The resulting system has two states

1. Parallel : both look in the same direction $|\uparrow\uparrow\rangle$ or $|\downarrow\downarrow\rangle$
2. Antiparallel : the south pole of the first touches the north pole of the second, and vice-versa $|\uparrow\downarrow\rangle$ or $|\downarrow\uparrow\rangle$

Upon doing the experiment with no external force, the opposite magnetic poles will attract each other and the system will usually end up in the antiparallel state. A way to think about this is to attribute each of these state an *energy level*, calculated with the magnetic potential energy of the two magnets. Nature then, will always try to be in the state of lowest energy.

On the other hand, if the magnetic field surrounding these two magnets is strong enough (or if you hold them with your hands), the first state may become of lower energy, which would mean that the magnets would then align in the parallel state.

The relation between this example and quantum satisfiability is that the spin of each particle can be seen as a variable, and the local spin interactions corresponds to constraints on the total system.

[16] Think of it as the orientation of the tiny magnet

[17] Note that there are different spin properties and the one mentioned here is the spin $\frac{1}{2}$. As it possess two base states, it can be used as a physical representation of a qubit.

Therefore, a constraint in our case must be understood as something that will *cost energy* to the system, and a solution of the underlying satisfiability problem will be a quantum state that minimizes the total energy of the system.

What is interesting about this, is that the behaviour of the solution space of large satisfiability problems will therefore exhibit very similar properties to the atomic structures. For instance, certain materials, such as *spin glass*[18], are stuck in states that do not minimize their energy. Physicists thus believe that it has to do with the fact that in this case, solutions are very clustered and nature does not have the time to explore all possible states in order to find the best one.

3.2 Projectors & Hamiltonians

Let us go back to quantum computing and define what would be the mathematical definition of a constraint between multiple qubits. Because qubits can take infinitely many values, it won't be possible to have a constraint that is as straight-forward as excluding one possible state, like in the classical case.

This idea though must be kept, but since the set of possible states is no more discrete but spans linearly in an Hilbert space, we should rather speak of excluding a specific *subspace* of the states. An important distinction as well is that our given state $|\psi\rangle$ is now able to be in a superposition of multiple states, where some are perpendicular to the *forbidden subspace*, and some aren't. We will thus affect a *cost* for each state that lies in the forbidden subspace, according to the probability of being in this subspace when we project our state into it. For example, if we want to forbid the subspace spanned by the state $|\phi\rangle$, we will add the cost defined by the probability of our space to be in the forbidden subspace, which is given by

$$|\langle\phi|\psi\rangle|^2$$

Such a notion is generalized in quantum physics by the notion of *Hamiltonian*, which is an observable associated to the total energy of a system. Its eigenvalues represent all the possible energy states that the projected state can take, and the corresponding eigenvector gives the outgoing state after projection.

Definition 20 (Hamiltonian)**.**
An Hamiltonian H is an observable on the total energy of a n qubits system in state $|\psi\rangle \in \mathcal{H}$, represented by a $2^n \times 2^n$ Hermitian operator

$$H = \sum_{i=1}^{d} \lambda_i |\phi_i\rangle\langle\phi_i|$$

Remark 20. *The lowest energy state, which corresponds to the lowest eigenvalue λ_1, is called the "ground state".*

Remark 21. *An Hamiltonian is said to be "k-local" if it acts non-trivially on a k-qubit subset of the total system.*

Remark 22. *An Hamiltonian of rank 1 projecting onto an unique state $|\phi\rangle$ with energy 1 will be called a "projector"*

$$\Pi^\phi = |\phi\rangle\langle\phi|$$

and we define in the same way a k-local projector Π_S^ϕ with respect to a k-qubits state $S \in \mathcal{H}$.

[18] https://en.wikipedia.org/wiki/Spin_glass

3.3 Quantum SAT

3.3.1 Definition

Let us now use the above notions in order to define the Quantum equivalent of 3SAT. Recall that in the classical case, we only exclude one possible state for each subset of k bits. Therefore, it would seem rather natural to define QSAT in terms of projectors.

Definition 21 (Quantum SAT)**.**
Input : *An Hamiltonian $H = \sum_{j=1}^{m} \Pi_j$ that is a sum of projectors acting on a n-qubit Hilbert space $\mathcal{H}$.*
Promise : *Either*

- *There exists an n-qubit state $|\psi\rangle \in \mathcal{H}$ such that $\Pi_j|\psi\rangle = 0$ for all $j = 1, \ldots, m \rightarrow H \in L_{yes}$.*
- *$\sum_{i=1}^{m} \langle\psi|\Pi_i|\psi\rangle \geq \delta$ for all $|\psi\rangle \in \mathcal{H}$ where $\delta = \frac{1}{poly(n)}$, $\rightarrow H \in L_{no}$.*

Problem : *Decide which one is the case*

Remark 23. *The "promise gap" δ is required in order to be able to distinguish a non-zero result from a null one in polynomial time.*

Remark 24. *If all the projectors are k-local, then we denote the problem as kQSAT.*

Now this definition is quite convenient, as it allows us to formulate the problem in terms of linear algebra in the following way

Claim 1. *An instance H of kQSAT is satisfiable if its kernel* $\ker H$ *has non-zero dimension.*

3.3.2 Product and entangled states

An interesting question that we can ask that has no counterpart in the classical case is to know whether the complexity of the problem changes if we require the satisfying state $|\psi\rangle$ to be a product state. It is not clear at this point if it actually makes a different or not. We will denote the restriction of QSAT to product state solutions as PRODSAT.

Definition 22 (PRODSAT)**.**
Input : *An Hamiltonian $H = \sum_{j=1}^{m} \Pi_j$ that is a sum of projectors acting on a n-qubit Hilbert space $\mathcal{H}$.*
Promise : *Either*

- *There exists an n-qubit product state $|\psi\rangle \in \mathcal{H}$ such that $\Pi_j|\psi\rangle = 0$ for all $j = 1, \ldots, m \rightarrow \Pi \in L_{yes}$.*
- *$\sum_{i=1}^{m} \langle\psi|\Pi_i|\psi\rangle \geq 1$ for all $|\psi\rangle \in \mathcal{H} \rightarrow \Pi \in L_{no}$.*

Problem : *Decide which one is the case*

3.4 The Local Hamiltonian Problem

This first definition of quantum satisfiability is very close to the classical definition, but it might be interesting to take a step back and look at a broader one. It is indeed hard to talk about QSAT without mentioning *The Local Hamiltonian Problem*, whose study has began earlier and seem of an even bigger importance in quantum complexity theory.

In fact, we will that in the end, QSAT is nothing but a special case of LH, though they come with different complexities.

3.4.1 Definition

The main difference here is that we allow our Hamiltonian to have ranks ≥ 1, within the terms we defined above, it means that we will use k-local Hamiltonians instead of k-local projectors. One of the direct implications of this is that we won't be able to guarantee that the ground state of our satisfying state is 0. Thus, the goal becomes to find the state that minimizes the ground energy and know if it is bigger than a value defined in the promise.As a result, we can say that the Local Hamiltonian problem is in fact closer to the classical MAXSAT.

Definition 23 (LH)**.**
Input : *A k-local Hamiltonian H acting on a n-qubits Hilbert space $\mathcal{H}$.*

$$H = \sum_{j=1}^{m} H_j$$

Promise : *There are real values a and b with $a < b$ such that*

- *There exists a n-qubits quantum state $|\psi\rangle \in \mathcal{H}$ such that $\langle\psi|H|\psi\rangle \leq a \rightarrow H \in L_{yes}$*
- *For all n-qubits quantum states $|\psi\rangle \in \mathcal{H}$, $\langle\psi|H|\psi\rangle \geq b \rightarrow H \in L_{no}$*

Problem : *Decide which one is the case.*

Remark 25. *Because of the fact that the energy of an Hamiltonian is minimized in the eigenspace, the problem can also be understood as either there exists an eigenvalue of H that is less than a, or all of them are greater than b.*

Remark 26. *3-SAT is an instance of 3-LH$_{0,1}$. For every clause*

$$c_j = x_a \vee \overline{x_b} \vee x_c$$

we define the corresponding Hamiltonian

$$\mathcal{H}_j = |0\rangle\langle 0|_a \otimes |0\rangle\langle 0|_b \otimes |0\rangle\langle 0|_c$$

Remark 27. *Using the same argument, 3-QSAT is equivalent to 3-LH$_{0,b}$*

3.4.2 Feynmann's Hamiltonian Computer

I discussed in the introduction some of the avant-garde thoughts of Feynmann on quantum computing. As a matter of fact, there is also another idea of his which I would like to discuss here.

Beyond the idea of quantum circuit or quantum Turing machine, Feynmann had another interpretation of what a *quantum computer* would look like. In order to understand it properly, we need to state Schrödinger's equation, which is of major importance in quantum physics as it describes the time evolution of any quantum system. It can be written in many ways, but since the matter of interest here is its meaning, I'll use the following.[19]

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = H|\Psi(t)\rangle$$

[19] $\hbar$ is Planck's constant and i is the imaginary unit

The resolution of this equation is a very hard problem which I will not discuss here. However, what interests us is that if we chose H so that it does not depend on the time, we can use a simpler version, which is often called the time-independent Schrödinger's equation.

$$E|\Psi\rangle = H|\Psi\rangle$$

This result explains why I presented the Hamiltonian as the observable of the energy of a system. Considering this, Feynmann's great idea was to use this time-independent equation to simulate a sequence of operations in a quantum mechanical system. In [6], he states that, if we can build an Hamiltonian that does that, then the only thing left to do is let nature run its course and measure the result, hence the idea of *Hamiltonian computer*.

Feynmann also provides a possible implementation of this kind of computer, which is very clearly explained by Daniel Nagaj in [14] and that I'll state here. He begins by describing the total space of the system as two registers. The first contains the n qubits that will be used for the actual computation (the *work* space), and the other serve as a counter to keep track of the order in which the operators must be applied (the *clock* space).

$$\mathcal{H} = \mathcal{H}_{work} \otimes \mathcal{H}_{clock}$$

In his original representation, the clock space contains L particles which I'll denote $\{c_0, \ldots, c_L\}$, and every step t is represented as all qubits in state $|0\rangle$ expect the t-th one.

$$|t\rangle_c = |0_{c_0}, \ldots, 0_{c_{t-1}}, 1_{c_t}, 0_{c_{t+1}}, \ldots, 0_{c_L}\rangle$$

Let us now focus on how to implement the actual operations on the work system. If the initial state is described by $|\psi_0\rangle_w \otimes |0\rangle_c$, where I use the simplified notation $|0\rangle_c = |1_{c_0}, 0_{c_1}, \ldots, 0_{c_L}\rangle$, then the Hamiltonian should enforce the following transition

$$|\psi_0\rangle_w \otimes |0\rangle_c \rightarrow U_1|\psi_0\rangle_w \otimes |1\rangle_c = |\psi_1\rangle_w \otimes |1\rangle_c$$

A simple way of achieving this is by using the following term

$$U_1 \otimes |1\rangle\langle 0|_c$$

Now since all Hamiltonian must be Hermitian, so that the computation can be reversible, we must add the conjugate term and thus we get the first term of Feynmann's Hamiltonian

$$H_{F,1} = U_1 \otimes |1\rangle\langle 0|_c + U_1^\dagger \otimes |0\rangle\langle 1|_c$$

Finally, in order to represent all the steps in the computation, Feynmann will simply use the following

$$H_F = \sum_{t=1}^{L} H_{F,t}$$

With such a construction and according to Schrödinger's equation, this Hamiltonian should bring the state of the computer into a superposition

$$|\psi\rangle = \frac{1}{\sqrt{L+1}} \sum_{t=0}^{L} |\psi_t\rangle_w \otimes |t\rangle_c$$

Considering this, if I measure such a state after enough time and find that the clock registers contains $|L\rangle$, then I can simply take the measure on the working qubits as the solution of my problem, $|\psi_L\rangle$. Moreover, I can always improve the probability of finding my system in this state by providing, for instance, $2L$ extra clock qubits and linking them to the operation $\mathbb{I}$ on the working qubits. With such a system, the probability of measuring $|L\rangle$ in the clock register is $\frac{2}{3}$, which is more than enough for practical purposes.

The idea of computing with Hamiltonians has evolved quite a lot since then, and in fact it has given birth to a new computing model which is called *Adiabatic Quantum Computation*, which has been proven to be equal to the standard quantum circuits computing model. But that is a story for another day.

3.5 Complexity

Let us now focus on trying to find a proper complexity class for all problems we talked about. Since the studies in this field are quite recent, I will go over several results in chronological order

3.5.1 Kitaev's Clock Construction

The first person to have ever studied the problem of quantum satisfiability in the context of complexity theory was the Russian-American physicist A. Yu. Kitaev in 1985. In his book [10], he reuses Feynmann's idea of Hamiltonian computation to build a proof that is based on similar principles as the one used for the Cook-Levin theorem.

His first attempt, he proved that the problem with Hamiltonians built on $O(\log n)$-local interactions was contained in QMA. I will not provide the proof here since a little tweak in the end allowed him to prove the following, stronger result.

Theorem 11 (Kitaev '02)**.** *5-LH is QMA-complete*

Proof. As always, we will separate this proof with respect to the two requirements of QMA-completeness.

5LH $\in$ QMA

Proving that 5LH is in QMA, as in the classical case, requires to provide an explicit quantum circuit U (Arthur) that can verify a given answer in polynomial time. Kitaev proved it by adding an extra qubit to the considered system (who will serve as a recipient for the answer), and constructing, for each local Hamiltonian $H_j = \sum_s \lambda_s |\psi_s\rangle\langle\psi_s|$, a measurement operator

$$W_j : |\psi_s, 0\rangle \to |\psi_s\rangle \otimes \left(\sqrt{\lambda_s}|0\rangle + \sqrt{1-\lambda_s}|1\rangle\right)$$

Now, given an arbitrary state $|\phi\rangle$ expanded in the eigenspace of H_j with the form $|\phi\rangle = \sum_s c_s|\psi_s\rangle$,

the probability of getting 1 in the answer qubit is given by

$$\begin{aligned}
Pr\big[W_j \textit{ accepts } |\phi,0\rangle\big] &= \langle\phi,0|W_j^\dagger\big(\mathbb{I}\otimes|1\rangle\langle 1|\big)W_j|\phi,0\rangle \\
&= \Big(\sum_s c_s^*\langle\psi_s,0|\Big)W_j^\dagger\big(\mathbb{I}^{\otimes n}\otimes|1\rangle\langle 1|\big)W_j\Big(\sum_t c_t|\psi_t,0\rangle\Big) \\
&= \sum_{s,t}\sqrt{1-\lambda_s}c_s^*\sqrt{1-\lambda_t}c_t\langle\psi_s|\psi_t\rangle \\
&= \sum_s(1-\lambda_s)c_s^*c_s \\
&= 1-\langle\phi|H_j|\phi\rangle
\end{aligned}$$

This is already a great result, since it will accept $|\phi\rangle$ with probability $p \geq 1-a$ if $|\phi\rangle$ is in the ground space of H_j, and in the opposite case, it will only get fooled with probability $p \leq 1-b$.

Now since we want to not only check one specific constraint, but all of them at the same time, Kitaev's idea was the chose an integer j uniformly at random, and to apply the corresponding operator W_j. An easy way to do this is to add an extra set of qubits to our initial state, which we'll denote by the term operator qubits and which are in the state

$$|\phi\rangle_{op} = \frac{1}{\sqrt{m}}\sum_j|\underline{j}\rangle$$

With these extra qubits, the operator we want to build can simply be represented as

$$W = \sum_j|\underline{j}\rangle\langle\underline{j}|\otimes W_j$$

Applying W to our state $\phi = |\phi_{op}\rangle\otimes|\phi_{in}\rangle\otimes|0\rangle_{ans}$ provides an accepting probability of

$$\begin{aligned}
Pr\big[W \textit{ accepts } |\phi\rangle\big] &= \sum_j\frac{1}{m}Pr[W_j \textit{ accepts } \phi] \\
&= \sum_j\frac{1}{m}\big(1-\langle\phi|H_j|\phi\rangle\big) \\
&= 1-\frac{\langle\phi|H|\phi\rangle}{m}
\end{aligned}$$

which again is enough for Arthur to efficiently decide if he should accept it or not.

5LH is QMA-hard

This part gets trickier, since the goal here is to provide a polynomial-time reduction from any quantum circuit to a 5-Local Hamiltonian problem. In his proof, Kitaev will reuse Feynmann's idea of time-independent Hamiltonian and, given a quantum circuit $U = U_L\ldots U_1$ acting on an input state that is composed of Merlin's p-qubits certificate $|\xi\rangle$ and $n-p$ ancillas qubits, use a Hilbert space

$$\mathcal{H}_{work}\otimes\mathcal{H}_{clock}$$

where the exact qubit implementation $\mathcal{H}_{clock}$ will be explicitely defined later (for the moment let us believe that it can be in states $|0\rangle,\ldots|L\rangle$). Moreover, in order to simplify the notation, I will denote

the work qubits as $\{w_1, \ldots, w_m\}$.

The idea here is to try to enforce the ground state of this Hamiltonian to be a superposition of the computational path of U, adding *penalties* (in the form of extra energy) whenever the state deviates from this path. He calls such a superposition the *history* state of U

$$|\phi\rangle_{history} = \frac{1}{\sqrt{L+1}} \sum_{t=0}^{L} |\psi_t\rangle_w \otimes |t\rangle_c$$

where $|\psi_t\rangle_w = U_t \ldots U_1 |\psi_0\rangle_w$ and $|\psi_0\rangle_w = |\xi\rangle \otimes |0, \ldots, 0\rangle$. Having said all that, Kitaev's Hamiltonian now consists of three terms

$$H = H_{in} + H_{out} + H_{prop}$$

which are defined in the following way.

1. $H_{in}-$ Its role is to verify the fact that all the ancillas qubits are in initial state $|0\rangle$ when $t = 0$. This translates nicely into

$$H_{in} = \bigotimes_{i=p+1}^{n} |1\rangle\langle 1|_{w_i} \otimes |0\rangle\langle 0|_c$$

2. $H_{out}-$ It has low ground state only when the quantum circuit U ends up accepting the input by providing a constraint on the output qubit of U (which in our definition is nothing but the first qubit of the input) at time $t = L$

$$\hat{H}_{out} = \Pi^1_{w_1} \otimes |L\rangle\langle L|_{clock}$$

3. $H_{prop}-$ In the same manner as in the classical case, the idea is now to verify the proper progression of the circuit U. The Hamiltonian constructed here is nothing but Feynmann's in his Hamiltonian computer. For each transition $(t-1) \to t$, we need to enforce the fact that the operator U_t is applied. This yields

$$\begin{aligned} H_{prop,t} &= \frac{1}{2}\mathbb{I}_w^{\otimes n} \otimes \big(|t-1\rangle\langle t-1|_c + |t\rangle\langle t|_c\big) \\ &= -\frac{1}{2}\big[(U_t)_w \otimes |t\rangle\langle t-1|_c + (U_t^\dagger)_w \otimes |t-1\rangle\langle t|_c\big] \end{aligned}$$

   Now it is straightforward to see that this operator will yield 0 for every term that does not involve step t or $t-1$. On the other hand, when applying it to the terms including $|\psi_{t-t}\rangle$ and $|\psi_t\rangle$, it creates a penalty if the two states are not related with respect to the corresponding unitary operator U_t. In order to show this, let us consider an arbitrary state

$$|\phi\rangle = |\phi_t\rangle_w \otimes |t\rangle_c + |\phi_{t+1}\rangle_w \otimes |t+1\rangle_c$$

   When measured with this Hamiltonian, we get the following result

$$\begin{aligned} \langle\phi|H_{prop,t}|\phi\rangle &= \frac{1}{2}\langle\phi_{t-1}|\mathbb{I}|\phi_{t-1}\rangle\langle t-1|t-1\rangle^2 + \frac{1}{2}\langle\phi_t|\mathbb{I}|\phi_t\rangle\langle t|t\rangle^2 \\ &\quad - \frac{1}{2}\langle\phi_t|U_t|\phi_{t-1}\rangle\langle t|t\rangle\langle t-1|t-1\rangle - \frac{1}{2}\langle\phi_{t-1}|U_t^\dagger|\phi_t\rangle\langle t-1|t-1\rangle\langle t|t\rangle \\ &= 1 - \parallel \langle\phi_t|U_t|\phi_{t-1}\rangle \parallel \end{aligned}$$

which is equal to 0 only when the $\phi_t = U_t|\phi_{t-1}\rangle$ as we wanted. After having defined such an $H_{prop,t}$ for every U_t, we combine them into the final term

$$H_{prop} = \sum_{t=1}^{L} H_{prop,t}$$

Now that we have our Hamiltonian, but we still need to prove that it has eigenvalues that correspond to the cases where U accepts Merlin's proof $|\psi_c\rangle$.

At this point, we could just left this proof in this state, and in fact that is what Kitaev did in his first attempt, counting in the clock state in binary with $\log L$ qubits. The problem however, is that such an Hamiltonian acts on the whole clock space and thus is only $O(\log L)$-local.

A way to go around that is to realize that the clock state can be embedded in a larger Hilbert space containing L qubits in total, and where the correspondence with the time is given by

$$|t\rangle \to |\underbrace{1,\ldots,1}_{\text{t}},\underbrace{0,\ldots,0}_{L-t}\rangle$$

With such a clock space, we can replace our previous clock projectors in the following fashion

$$\begin{aligned}
|0\rangle\langle 0|_c &\to \Pi_1^0 \\
|L\rangle\langle L|_c &\to \Pi_L^1 \\
|t\rangle\langle t|_c &\to \Pi_t^1\Pi_{t+1}^0 \\
|t-1\rangle\langle t|_c &\to \Pi_{j-1}^1\big(|0\rangle\langle 1|\big)_j\Pi_{j+1}^0 \\
|t\rangle\langle t-1|_c &\to \Pi_{j-1}^1\big(|1\rangle\langle 0|\big)_j\Pi_{j+1}^0
\end{aligned}$$

which now uses only 3-local interactions. But now a new problem arises, given such an enormous clock space, there are a lot of states that should not be possible. What if, for instance, the clock is in a state $|0101\ldots01\rangle$? We should forbid such spaces, as they do not correspond to a legal representation of the time.

Kitaev deals with this problem pretty simply. He adds a new term to his Hamiltonian with the given role of forbidding unwanted spaces.

$$H_{clock} = \mathbb{I}_w^{\otimes n} \otimes \sum_{t=1}^{L-1} \Pi_t^0\Pi_{t+1}^1$$

At the end, since all U_t's act on 2 qubits from the work space and the clock can be represented with maximum 3 qubits, the hamiltonian H is 5-local as expected. □

3.5.2 Kempe and Regev on LH

The story continues two decades later when Julia Kempe and Oded Regev release a proof that 3LH is QMA-complete as well.

Theorem 12 (Kempe-Regev, '03)**.** *3LH is QMA-complete*

I will not enter the details of the proof [9], but the idea is the following. The 5-locality of Kitaev's Hamiltonian comes from the fact that circuit operations acts on 2 qubits and he needs 3 to define the clock state. Here, Kempe and Regev find a way to reduce the locality of the propagation Hamiltonian by using only 1 qubit of the clock register, which yields a result for 3-local Hamiltonians only.

The story doesn't stop here. 1 year later, the same scientists, with the help of Kitaev, manage to reduce the locality even further and prove the following

Theorem 13 (Kempe-Kitaev-Regev, '04)**.** *2LH is QMA-complete*

Once again, I will not state the details of this proof [8], because it relies on concepts that are far more complicated than what I studied in the context of this project. The main difficulty though was to include the fact that unitary operations of a quantum circuit U already using 2 qubits, it would mean that no clock qubit can be used. Their construction allow to reduce the constraint on the unitary operations in the work register to only 1 qubit, which gives a resulting Hamiltonian that is only 2-local.

3.5.3 About 1LH

Now, the interesting question becomes : can we reduce it even further ? Apparently not. In fact, it is not hard to see that even with a classical computer, finding the ground state of a sum of Hamiltonian that only act on a single qubit can be done by iterating on each qubit, and finding the lowest eigenvalue for each. Thus, this ultimate result on LH.

Theorem 14. *1LH* $\in$ *P*

3.5.4 Bravyi on QSAT

Now that we have quite a clear vision on the difficulty of finding ground states of local Hamiltonians, let us turn our eyes on QSAT. As a remainder, we know that each Hamiltonian here is of rank 1 only, so we can expect the problem to be somewhat simpler.

As a matter of fact, there is also another important difference with LH. When considering QSAT, the fact that we only deal with rank 1 projectors allows us to require the ground state of the Hamiltonian to be *exactly* 0. As a direct consequence, we can expect to compute QSAT with one-sided error on a quantum computer, hence the corresponding complexity class QMA_1.

The first person to really focus on QSAT in the way we do here and find interesting results is Sergei Bravyi. In his article from 2006 [4], he uses the same clock construction as Kitaev, but then maps the other Hamiltonians onto a set of 4-local projectors, which yields the following result

Theorem 15 (Bravyi, '06)**.** *4QSAT is* QMA_1*-complete*

In the same article, Bravyi shows an even more interesting fact. He provides an explicit classical polynomial-time algorithm that, given an instance of 2QSAT H, outputs one of the following

1. H has no satisfying assignment
2. A product state $|\psi\rangle = |\psi_1\rangle \otimes \dots |\psi_n\rangle$ such that $H|\psi\rangle = 0$.
3. An instance $\tilde{H}$ of quantum 2QSAT that acts on $n-1$ or smaller number of qubits, and that is equivalent to H.

Having found this, it is easy to see that we can repeat this algorithm at most n times and get an answer in polynomial time. Thus,

Theorem 16 (Bravyi, '06)**.** *2QSAT* $\in$ *P*

Once again, I will not provide the proof [4] here. The main idea is to use a reduction method that propagates the assignment of a qubit to its neighbours (in the same way as in the classical case). Such a propagation is realized with the help of a *Tranfer Matrix* which I will explain more in section 4.4.1.

All these result lead to the very interesting possible conclusion that QSAT behaves on quantum computers in the same way as SAT on a classical one.

3.5.5 Gosset and Nagaj on QSAT

Bravyi will not find a way to reduce his proof of 4QSAT, and we must way 7 years to finally have an answer. It is the two physicists David Gosset and Daniel Nagaj that will finally bring our exploration of QSAT to an end, with a result that comfort what we already believed.

Theorem 17 (Gosset-Nagaj, '13)**.** *3QSAT is* QMA_1*-complete*

In this paper, Gosset and Nagaj provide a new kind of clock construction whose objective is to reduce the locality of the Hamiltonian by providing two clocks with different behaviours, one using only 1-local interactions and the other 2-locals, coupled together with extra hamiltonians that check the synchronization between both and the legality of each state. I won't provide more details here because it gets very hard, but zealous readers can admire it in [7].

4 Random QSAT

An other interesting approach on the satisfiability problem is the study of the probability of finding a satisfiable set of constraints when one picks them at random. In this chapter, I will first explain how this problem is defined, and then provide a study of random graphs in order to understand different properties that it exhibits.

Most of the results I will present here are those of C. Laumann, R. Moessner, A. Scardicchio and S.L. Sondhi in [11] and [12], as well as those of S. Bravyi, C. Moore and A. Russell in [3].

4.1 Introduction

The idea here (which is nothing but the QSAT version of Random SAT), is to consider a set of n qubits and randomly chose a collection of m k-tuples out of the $\binom{n}{k}$ possible combinations. When doing that, we will denote $\alpha = \frac{m}{n}$ as the *clause density* of the problem.

Now in a similar way as QSAT, we will affect to every k-tuple j ($1 \leq j \leq m$) a projector

$$\Pi_j^{\phi} = |\phi_j\rangle\langle\phi_j|$$

picked at random in the corresponding 2^k-dimensional Hilbert space $\mathcal{H}_j$. The goal now is to analyse the probability of having a satisfiable assignment as $m, n \to \infty$ with α constant

4.2 The Geometrization Theorem

One of the first important result that one comes accross when analysing Random SAT is that the satisfiability of formulas does not depend on the values of the constraints but only on the geometry of the graph. This intriguing result raises the question of its applicability in the quantum case. Fortunately, the physicists which I mentioned above were able to prove a similar statement, which is denoted as the *geometrization theorem* and is stated as follows.

Theorem 18 (Geometrization Theorem)**.**
Given an instance $\mathcal{H}$ of random kQSAT over an hypergraph G, the degeneracy (dimension) of zero-energy states $dim(ker(\mathcal{H}))$ takes a particular value D with respect top the choice of projectors on the edges of the hypergraph G.

Proof. Let us take a fixed hypergraph G with m hyperedges, then

$$H = \sum_{i=0}^{m} \Pi_i = \sum_{i=0}^{m} |\phi_i\rangle\langle\phi_i|$$

depends only on the values of the $|\phi_i\rangle$'s and thus H can be seen as a matrix-valued function of the matrix

$$\Phi = \big(|\phi_0\rangle, |\phi_1\rangle, ..., |\phi_m\rangle\big)$$

which is a $2^k \times m$ matrix containing all 2^k components of each one of the m vectors $|\phi_i\rangle$. Now what we need to show is that $dim(ker(\mathcal{H}(\Phi)))$ is independent of Φ with probability 1.

The idea is that if we chose an explicit Φ so that H has maximal rank r, then there exist an $r \times r$

submatrix of H (let's call it A) such that $\det A$ is non-zero [20]. The trick is that this submatrix determinant is also a polynomial in the components of $|\phi\rangle$ and therefore is only zero on a submanifold of $|\phi\rangle$ of codimension at least 1.

Therefore, with probability 1, H has rank r and the degeneracy of zero-energy states is

$$\dim \ker H = 2^n - r$$

□

This result constitutes one of the more important ones in this study since it allows us to analyse quantum satisfiability as a graph geometry property and without reference to quantum Hamiltonians, which simplifies a lot the analysis. A good place to start then would be to analyze how such random graphs behave, particularly in the case where the number of vertices goes to infinity.

4.3 Erdös-Rényi Graphs

Erdös and Rényi are two mathematicians that developed a theory of random graphs which I will presented. I will explain the major points concerning the study of random QSAT, with the help of [2].

4.3.1 Two Models

When talking about random graphs, two different yet closely related definitions can arise. In the first, we will fix the number of vertices and edges, respectively n and m, and then place these edges randomly on the graph. The original name of this model is $\mathcal{G}(n, m)$, though for convenience we will often refer to it as $\mathcal{G}_m^n$. Furthermore, we will denote a graph built on this model as

$$G_m^n \sim \mathcal{G}_m^n$$

In the other model we don't want a specific number of edges but instead, we fix a probability p such that every possible edge out of the $\binom{n}{2}$ possible appears with probability p. Here again, we will refer to graphs built on this model as

$$G_p^n \sim \mathcal{G}_p^n$$

Now if we try to compare the two models, we can see that the probability of having a graph with m vertices when going for the $\mathcal{G}_p^n$ model is given by

$$Pr[G_p^n \textit{ has m vertices}] = p^m(1-p)^{\binom{n}{2}-m}$$

and furthermore, any graph with m vertices has an equal probability of

$$\binom{\binom{n}{2}}{m}^{-1} p^m(1-p)^{\binom{n}{2}-m}$$

of being chosen. Finally, the expected number of edges in such a graph is given by

$$\mathbb{E}[|E_p^n|] = \binom{n}{2} p$$

[20] This theorem is provided in Appendix B

Using the Law of Large Numbers[21], we can say that

$$Pr\left[|E_p| = \binom{n}{2}p\right] \xrightarrow{n\to\infty} 1$$

and thus we can suspect the fact that, for large values of n, $\mathcal{G}(n,p)$ and $\mathcal{G}(n, m = \binom{n}{2}p)$ behave similarly, which is the case for monotone properties as we will see. In particular, since in our problem we consider a fixed clause $\alpha = \frac{m}{n}$, we will look at probabilities of order

$$p = \frac{m}{\binom{n}{2}} = \frac{\alpha n}{\binom{n}{2}} = \frac{2\alpha}{n-1} \sim \frac{2\alpha}{n}$$

4.3.2 Properties

Now that we have seen the two models, we may want to take a closer looks at what kind of properties we can expect from them. But first, we need to define what we refer to as a random graph property. In particular, we will here take a look at *monotone properties* of random graphs.

Definition 24 (Monotone Property)**.**
A monotone property is a property Q such that for any graphs G, H,

$$G \in Q, G \subset H \implies H \in Q$$

The use of monotone properties makes the study of random graph much easier, since the probability for a model to verify a given property Q, which I'll denote respectively as $P_m(Q)$ and $P_p(Q)$ depending on the model, increases with m or p. In addition to that, and in order to simplify the notation, I will use the following notation used by Bollobàs

Definition 25 (Almost Every)**.**
We say that almost every (a.e.) graph G_m (resp. G_p) satisfy a property Q if

$$\lim_{n\to\infty} P_m(Q) \text{ (resp. } P_p(Q)) = 1$$

Moreover, since it is not very convenient to always have to switch between the models, I'll use the $\mathcal{G}_n^p$ one in this study. One should keep in mind that under several technical assumptions (which I'll not further discuss since it doesn't concern our matter), those two models are equivalent with respect to the expected values of monotone properties.

Finally, an important point that Erdös and Rényi discovered was that the behaviour monotone graph properties in random graphs is often dictated by *threshold* probabilities, under which the probability of having the property is almost 0 and which quickly becomes 1 as it slightly increases above the threshold value.

4.3.3 Apparition of Subgraphs

The context being set up, the specific property around which random SAT is built is the property of containing a specific subgraph. It is easy to see that it is indeed a monotone property, and as such it does exhibit a particular threshold probability above which the probability of apparition quickly

[21] Appendix C

becomes 1. At this point, the number of subgraphs is defined as a Poisson distribution[22].

Finally, the subgraphs which we are going to look at are called *strictly balanced*[23], and they appear with the following probability.

Theorem 19 (Stricly Balanced Subgraph)**.**
Let H be a fixed strictly balanced graph of order k and size $l \geq 2$, and denote by $|Aut(H)|$ the size of its automorphism group. Moreover, set $p = cn^{-k/l}$ with $c \geq 0$ constant. Finally, denote by X_H the number of distinct[24] *H-subgraphs of G_p^n. Then,*

$$X_H \xrightarrow{D} P_\lambda \text{ with } \lambda = \frac{c^l}{|Aut(H)|}$$

In particular, it means that

$$\lim_{n\to\infty} \mathbb{P}[X_H = r] = \frac{e^{-\lambda}\lambda^r}{r!}$$

Proof. Let's count all the possibilities of including a H subgraph in G_p^n. First of all, there is

$$\binom{n}{k}$$

ways of choosing k vertices out of the n in V. Now, there are

$$\frac{k!}{|Aut(H)|}$$

ways of choosing the edges so that the graph is similar to H. Finally, each of the chosen edges appears with probability p, so the total probability of having all edges chosen is given by

$$p^l$$

Putting it all together and using Stirling's formula we get,

$$\mathbb{E}[X_H] = \binom{n}{k}\frac{k!}{|Aut(H)|}p^l \sim \frac{n^k}{|Aut(H)|}p^l = \frac{c^l}{|Aut(H)|} = \lambda$$

Now, we only need to show that for every fixed $r = 0, 1, \ldots$, the expected number of ordered r-tuples of H graphs which is represented by the rth factorial moment $\mathbb{E}_r[X_H]$, is asymptotically equals to λ^r. This time, there is

$$\prod_{i=0}^{r-1}\binom{n-ik}{k}$$

ways of choosing the vertices for every graph, and

$$\left(\frac{k!}{|Aut(H)|}\right)^r p^{lr}$$

[22] Appendix C
[23] Appendix E
[24] Note that this result is the same without the distinct subgraphs property but the proof is harder and we do not need it in our context. More information about this can be found in [2], p.81

ways of choosing the edges, including the probability that each one appears. Altogether and with Stirling's formula again, we get

$$\begin{aligned}\mathbb{E}_r[X_H] &= \left[\prod_{i=0}^{r-1}\binom{n-ik}{k}\right]\left(\frac{k!}{|Aut(H)|}\right)^r p^{lr} \\ &\sim \left[\prod_{i=0}^{r-1}\frac{(n-ik)^k}{k!}\right]\left(\frac{k!}{|Aut(H)|}\right)^r p^{lr} \\ &\sim \frac{n^{rk}}{|Aut(H)|^r}p^{rl} \\ &= \lambda^r\end{aligned}$$

From there, the Poisson distribution follows. □

Having said that, it is easy to understand the following threshold probability.

Theorem 20 (Probability Threshold)**.**
Let H be an arbitrary fixed graph with maximal average degree $\frac{2l}{k} > 0$. Then

$$\lim_{n\to\infty} Pr[H \subset G_p] = \begin{cases} 0 & \text{if } pn^{k/l} \to 0 \\ 1 & \text{if } pn^{k/l} \to \infty \end{cases}$$

4.3.4 The Giant Component

In the light of the above theorem, Erdös and Rényi were able to prove that, under low clause density ($\alpha = \frac{1}{2}$ for a normal graph to be more precise), the only components that could appear were trees and finite loops. This is due to the fact that every subgraph that is more complicated than a cycle contains therefore $m \geq n+1$ edges, and we have seen that such subgraphs cannot appear before some $p = \frac{c}{n}$. Now the study of such behaviours is very technical and difficult, so I will only provide an useful overview and skip the theorems that lie behind.

When one goes over this threshold though, we can observe the apparition of a *Giant Component* at clause density α_{gc}. Such a component is defined by a size larger than $\frac{2}{3}$ of the graph and which quickly connects to every remaining isolated subgraph as one passes over the threshold probability.

Having said that, the giant component in a random k-regular hypergraph[25][26] appears after a specific clause density, which is defined by

$$\alpha_{gc} = \frac{1}{k(k-1)}$$

What we should expect from this behaviour when looking at 2QSAT is that such an component would allow the presence of a sufficient amount of unsatisfiable subgraphs (in our case figure eights) which will lead to a polynomial-time distinguishable lower bound for the ground state of H.

[25] a brief definition of hypergraphs is provided in Appendix F.
[26] a normal graph is just the case $k = 2$.

4.3.5 The Hypercore

An other interesting phenomenon is the emergence of an *Hypercore*. A k-Hypercore is defined as a component on which we cannot get an empty graph by successively removing each k-hyperedge that contain a vertex of degree 1.

Now it is easy to see that in the case $k = 2$, the giant component itself represent an 2-hypercore. Within higher values of k though, we can observe that the apparition of the Giant Component and the Hypercore do not coincide.

The threshold value for the apparition of a k-hypercore is not as straightforward as the one for the giant component, and requires a lot of prerequisites. I will therefore not enter details and you'll have to believe me that the value for $k = 3$ is

$$\alpha_{hc} = 0.81$$

4.4 Random 2QSAT

4.4.1 Bravyi's Transfer Matrix

The results in [11] and [12] are based on a construction called the *Transfer Matrix*, that was earlier discovered by Bravyi. This construction allows to propagate an assignment on a qubit onto its neighbours, in the same way that we can propagate boolean assignments in the classical 2SAT by looking at the implication graph. The matrix is defined in the following way.

Theorem 21 (Bravyi's Transfer Matrix)**.**
For every starting state $|\psi_i\rangle$ and projector $\Pi_{ij} = |\phi_{ij}\rangle\langle\phi_{ij}|$, there exist a matrix T_{ij} - called the Bravyi's Transfer Matrix - which yields a state $|\psi_j\rangle = T_{ij}|\psi_i\rangle$ such that

$$\langle\psi_{ij}|\Pi_{ij}|\psi_{ij}\rangle = 0$$

Where $|\psi_{ij}\rangle = |\psi_i\rangle \otimes |\psi_j\rangle$.

Proof. First of all, let us rewrite the constraint using our hypotheses

$$\begin{aligned}
\langle\psi_{ij}|\Pi_\phi|\psi_{ij}\rangle &= 0 \\
\big(\langle\psi_i| \otimes \langle\psi_j|\big)\Pi_{ij}\big(|\psi_i\rangle \otimes |\psi_j\rangle\big) &= 0 \\
\Big(\langle\psi_i| \otimes \langle\psi_i|(T_{ij})^\dagger\Big)\Pi_{ij}\Big(|\psi_i\rangle \otimes T_{ij}|\psi_i\rangle\Big) &= 0 \\
\Big(\langle\psi_i| \otimes \langle\psi_i|(T_{ij})^\dagger\Big)|\phi\rangle\langle\phi|\Big(|\psi_i\rangle \otimes T_{ij}|\psi_i\rangle\Big) &= 0
\end{aligned}$$

Now, given the properties of the scalar product, we have

$$\Big(\langle\psi_i| \otimes \langle\psi_i|(T_{ij})^\dagger\Big)|\phi\rangle = 0 \Leftrightarrow \langle\phi|\Big(|\psi_i\rangle \otimes T_{ij}|\psi_i\rangle\Big) = 0$$

which mean that we can only focus on finding T_{ij} such that

$$\langle\phi|\Big(|\psi_i\rangle \otimes T_{ij}|\psi_i\rangle\Big) = 0$$

Now, since $|\psi_i\rangle$ is a 1-qubit state, it can be written as

$$|\psi_i\rangle = \sum_{\alpha \in \mathbb{F}_2} (\psi_i)_\alpha |\alpha\rangle$$

In the same way, $|\phi_{ij}\rangle$ being a 2-qubit state, we write

$$|\phi_{ij}\rangle = \sum_{\beta,\gamma \in \mathbb{F}_2} (\phi_{ij})_{\beta\gamma} |\beta\gamma\rangle$$

where $\phi_{00}, \phi_{01}, \phi_{10}, \phi_{11} \in \mathbb{C}$. Finally, we can write T_{ij} in Dirac notation as

$$T_{ij} = \sum_{\delta,\eta \in \mathbb{F}_2} (T_{ij})_{\delta\eta} |\delta\rangle\langle\eta|$$

where again $a, b, c, d \in \mathbb{C}$. Now we can easily see that

$$\begin{aligned}
|\psi_i\rangle \otimes T_{ij}|\psi_i\rangle &= \Big(\sum_{\alpha \in \mathbb{F}_2} (\psi_i)_\alpha |\alpha\rangle \Big) \otimes \Big(\big(\sum_{\delta,\eta \in \mathbb{F}_2} (T_{ij})_{\delta\eta} |\delta\rangle\langle\eta| \big) \big(\sum_{\alpha \in \mathbb{F}_2} (\psi_i)_\alpha |\alpha\rangle \big) \Big) \\
&= \Big(\sum_{\alpha \in \mathbb{F}_2} (\psi_i)_\alpha |\alpha\rangle \Big) \otimes \Big(\sum_{\delta,\eta \in \mathbb{F}_2} (T_{ij})_{\delta\eta} (\psi_i)_\eta |\delta\rangle \Big) \\
&= \sum_{\alpha,\delta,\eta \in \mathbb{F}_2} (\psi_i)_\alpha (\psi_i)_\eta (T_{ij})_{\delta\eta} |\alpha\delta\rangle
\end{aligned}$$

And furthermore,

$$\begin{aligned}
\langle\phi| \Big(\psi_i\rangle \otimes T_{ij}|\psi_i\rangle \Big) &= \Big(\sum_{\beta,\gamma \in \mathbb{F}_2} (\phi_{ij}^*)_{\beta\gamma} \langle\beta\gamma| \Big) \Big(\sum_{\alpha,\delta,\eta \in \mathbb{F}_2} (\psi_i)_\alpha (\psi_i)_\eta (T_{ij})_{\delta\eta} |\alpha\delta\rangle \Big) \\
&= \sum_{\alpha,\gamma,\eta \in \mathbb{F}_2} (\phi_{ij}^*)_{\alpha\gamma} (\psi_i)_\alpha (\psi_i)_\eta (T_{ij})_{\gamma\eta}
\end{aligned}$$

Since we have no information on $(\psi)_{0,1}$, this gives us the following equations system

$$\begin{cases}
(T_{ij})_{00}\phi_{00}^* + (T_{ij})_{01}\phi_{01}^* = 0 \\
(T_{ij})_{00}\phi_{10}^* + (T_{ij})_{01}\phi_{11}^* + (T_{ij})_{10}\phi_{00}^* + (T_{ij})_{11}\phi_{01}^* = 0 \\
(T_{ij})_{10}\phi_{10}^* + (T_{ij})_{11}\phi_{11}^* = 0
\end{cases}$$

Which with a little bit of linear algebra, boils down to

$$\begin{cases}
(T_{ij})_{00} = -\phi_{01}^* \rho \\
(T_{ij})_{01} = -\phi_{00}^* \rho \\
(T_{ij})_{10} = \phi_{11}^* \rho \\
(T_{ij})_{11} = \phi_{10}^* \rho
\end{cases} \quad \rho \in \mathbb{C}$$

Now this ρ factor depends on ψ_i and since we will later on divide the result of the multiplication by its norm to fall back into a normalized quantum state, we can arbitrarily set $\rho = -1$, and thus get the matrix

$$T_{ij} = \begin{pmatrix} \phi_{01}^* & \phi_{11}^* \\ -\phi_{00}^* & -\phi_{10}^* \end{pmatrix}$$

□

Example 1. *Let's take two arbitrary states*

- $|\psi_i\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
- $|\phi_{ij}\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{i}{2}|01\rangle + \frac{1}{2}|11\rangle$

This yield the following Transfer Matrix

$$T_{ij} = \begin{pmatrix} -\frac{i}{2} & \frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

From there, we can compute

$$T_{ij}|\psi_i\rangle = \begin{pmatrix} -\frac{i}{2} & \frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1-i}{2\sqrt{2}} \\ -\frac{1}{2} \end{pmatrix} = \frac{1-i}{2\sqrt{2}}|0\rangle - \frac{1}{2}|1\rangle$$

Which allows us to find

$$|\psi_j\rangle = \frac{T_{ij}|\psi_i\rangle}{\| T_{ij}|\psi_i\rangle \|} = \frac{\frac{1-i}{2\sqrt{2}}|0\rangle - \frac{1}{2}|1\rangle}{\frac{1}{\sqrt{2}}} = \frac{1-i}{2}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Finally, the 2-qubit vector is

$$|\psi_{ij}\rangle = \frac{1-i}{2\sqrt{2}}|00\rangle - \frac{1}{2}|01\rangle + \frac{1-i}{2\sqrt{2}}|10\rangle - \frac{1}{2}|11\rangle$$

And we can easily find that

$$\langle\phi_{ij}|\psi_{ij}\rangle = \frac{1}{2\sqrt{2}}\frac{1-i}{\sqrt{2}} + \frac{i}{2}\frac{1}{2} - \frac{1}{2}\frac{1}{2} = \frac{1-i+i-1}{4} = 0$$

4.4.2 Trees are PRODSAT

In the classical case, it is easy to see that trees are satisfiable. Each constraint only require one bit to be set in order to be satisfied. Therefore, an usual way to solve satisfiability to a tree is by fixing the value of the *leaves*, and then iterating on every neighbour towards the root until all bits have been assigned. In the quantum case, we will see that the case is quite similar. In fact, we do not even require to have entangled states in order to satisfy constraints.

Theorem 22 (Trees)**.**
A tree comprising n qubits has a satisfying product state

$$|\psi\rangle = \bigotimes_{j=1}^{n} |\psi_j\rangle$$

Proof. The first idea here is to define a non-orthogonal product basis for the Hilbert space of the tree

$$\mathcal{H} = \bigotimes_{i=0}^{n-1} \mathcal{H}_i$$

In order to do so, we will proceed in the following way

1. Start by choosing any 2 linearly independent quantum states $|0^0\rangle$, $|1^0\rangle$ as a basis for $\mathcal{H}_0$.
2. For every neighbour i, use the Transfer Matrix to compute the corresponding basis

$$|0^i\rangle = \frac{T_{0i}|0^0\rangle}{\| T_{0i}|0^0\rangle \|}$$
$$|1^i\rangle = \frac{T_{0i}|1^0\rangle}{\| T_{0i}|1^0\rangle \|}$$

3. Proceed recursively in the same way and therefore get a basis $\{|0^i\rangle, |1^i\rangle\}$ for every $\mathcal{H}_i$

Now let us consider a generic state $|\psi\rangle \in \mathcal{H}$ and the action of any projector $\Pi_{ij} = |\phi_{ij}\rangle\langle\phi_{ij}|$ on it. In our new basis, we can factorize this state as

$$\begin{aligned}|\psi\rangle =& |0^i 0^j\rangle_{ij} \otimes |\rho_0\rangle_{-ij} \\ &+ |1^i 1^j\rangle_{ij} \otimes |\rho_1\rangle_{-ij} \\ &+ \big(|0^i 1^j\rangle_{ij} + |1^i 0^j\rangle_{ij}\big) \otimes |\rho_+\rangle_{-ij} \\ &+ \big(|0^i 1^j\rangle_{ij} - |1^i 0^j\rangle_{ij}\big) \otimes |\rho_-\rangle_{-ij}\end{aligned}$$

Where $|\rho_*\rangle_{-ij}$ represent some state state of $|\psi\rangle$ without its qubits i, j. For the first two superposed states, we know that

$$\Pi_{ij}|0^i 0^j\rangle = \Pi_{ij}|1^i 1^j\rangle = 0$$

because we explicitely defined our two bases to satisfy this. Then, the third state yields

$$\begin{aligned}\langle\phi_{ij}|\Big(|0^i 1^j\rangle + |1^i 0^j\rangle\Big) &= \langle\phi_{ij}|\Big(|0^i\rangle \otimes T_{ij}|1^i\rangle + |1^i\rangle \otimes T_{ij}|0^i\rangle\Big) \\ &= \langle\phi_{ij}|\Big(|0^i\rangle \otimes (\epsilon\phi_{ij}^\dagger)|1^i\rangle + |1^i\rangle \otimes (\epsilon\phi_{ij}^\dagger)|0^i\rangle\Big) \\ &= \Big(\sum_{\alpha,\beta\in\mathbb{F}_2} (\phi_{ij}^*)_{\alpha\beta}\langle\alpha^i\beta^i|\Big)\Big[|0^i\rangle \otimes \Big(\sum_{\gamma\in\mathbb{F}_2} (\epsilon\phi_{ij}^\dagger)_{\gamma 1}|\gamma^i\rangle\Big) \\ &\quad + |1^i\rangle \otimes \Big(\sum_{\gamma\in\mathbb{F}_2} (\epsilon\phi_{ij}^\dagger)_{\gamma 0}|\gamma^i\rangle\Big)\Big] \\ &= \Big(\sum_{\alpha,\beta\in\mathbb{F}_2} (\phi_{ij}^*)_{\alpha\beta}\langle\alpha^i\beta^i|\Big)\Big(\sum_{\gamma\in\mathbb{F}_2} (\epsilon\phi_{ij}^\dagger)_{\gamma 1}|0^i\gamma^i\rangle + (\epsilon\phi_{ij}^\dagger)_{\gamma 0}|1^i\gamma^i\rangle\Big) \\ &= \Big(\sum_{\alpha,\beta\in\mathbb{F}_2} (\phi_{ij}^*)_{\alpha\beta}\langle\alpha^i\beta^i|\Big)\Big(\sum_{\gamma,\delta\in\mathbb{F}_2} (\epsilon\phi_{ij}^\dagger)_{\gamma\delta}|\delta^i\gamma^i\rangle\Big) \\ &= \Big(\sum_{\alpha,\beta\in\mathbb{F}_2} (\phi_{ij}^*)_{\alpha\beta}\langle\alpha^i\beta^i|\Big)\Big(\sum_{\gamma,\delta\in\mathbb{F}_2} (\epsilon\phi_{ij}^\dagger)_{\gamma\delta}|\delta^i\gamma^i\rangle\Big) \\ &= \sum_{\alpha,\beta\in\mathbb{F}_2} (\phi_{ij}^*)_{\alpha\beta}(\epsilon\phi_{ij}^\dagger)_{\beta\alpha} \\ &= 0\end{aligned}$$

Therefore, for our state $|\psi\rangle$ to satisfy the constraint, we only need to satisfy the constraint

$$(\langle 0^i 1^j| - \langle 1^i 0^j|)|\psi\rangle = 0$$

Which is totally equivalent to

$$\langle 0^i 1^j|\psi\rangle = \langle 1^i 0^j|\psi\rangle$$

In particular, this means that every pair of neighbouring qubits lie in the symmetric subspace of the corresponding $(\mathbb{C}^2)^2$ Hilbert space. Supposing we have defined

$$|\psi_i\rangle = \cos\frac{\alpha}{2}|0^i\rangle + e^{i\beta}\sin\frac{\alpha}{2}|1^i\rangle$$

with α, β given and we need to define γ and δ such that

$$|\psi_j\rangle = \cos\frac{\gamma}{2}|0^i\rangle + e^{i\delta}\sin\frac{\gamma}{2}|1^i\rangle$$

This will yield a product state

$$\begin{aligned}|\psi\rangle_{ij} = &\cos\frac{\alpha}{2}\cos\frac{\gamma}{2}|^i 0^j 0\rangle_{ij} \\ &+ \cos\frac{\alpha}{2}e^{i\delta}\sin\frac{\gamma}{2}|^i 0^j 1\rangle_{ij} + e^{i\beta}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2}|^i 1^j 0\rangle_{ij} \\ &+ e^{i\beta}\sin\frac{\alpha}{2}e^{i\delta}\sin\frac{\gamma}{2}|^i 1^j 1\rangle_{ij}\end{aligned}$$

And thus the only constraint for this state to be in the symmetric subspace is

$$\cos\frac{\alpha}{2}e^{i\delta}\sin\frac{\gamma}{2} = e^{i\beta}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2}$$

Which gives us a 1-dimensional solution for $|\varphi\rangle_j$. As a result, since the degree of freedom of the first qubit is 2 and we have 1 extra for every other qubit in the tree, this leads to a solution space of dimension $n+1$. □

4.4.3 Loops are PRODSAT

Now that we have explored the small fraction of tree graphs, let's turn our eyes on loops. When considering a single loop, there is in general two satisfying states.

Theorem 23 (Loops)**.** *A loop containing n qubits has a satisfying product state*

$$|\psi\rangle \in \left\{|00...0\rangle,\ |11...1\rangle\right\} \subset \bigotimes_{i=0}^{n-1}\mathcal{H}_i$$

Proof. Start with a single line of $n+1$ qubits, with the set of projectors $\{\Pi_{i,i+1}|\ i \in [0, n-1]\}$. From what we saw in the previous section, any state

$$|\psi\rangle \in Sym^n\mathbb{C}^2$$

satisfy the constraint

$$\Pi_{i,i+1}|\psi\rangle = 0 \quad \forall i \in [0, n-1]$$

Now, a simple way to transform this line into a loop is to say that the first and the last qubits are the same. This yields two additional constraints. First, we need to ensure the fact that their basis states are the same, i.e. that after applying all T's along the path, we fall back to the basis $\{|0^0\rangle, |1^0\rangle\}$. This translates nicely into

- $\Big(\prod_{i\in[0,n-1]} T_{i,i+1}\Big)|0^0\rangle = \lambda_0|0^0\rangle$
- $\Big(\prod_{i\in[0,n-1]} T_{i,i+1}\Big)|1^0\rangle = \lambda_1|1^0\rangle$

Thus our base is now fixed as the eigenvectors of $\prod_{i\in[0,n-1]} T_{i,i+1}$. Now, as we add our projector Π_{0n} in order to "close the loop", we can again factorize our state $|\psi\rangle \in Sym(\mathbb{C}^2)$ as

$$\begin{aligned}|\psi\rangle =& c_0|0^0 0^n\rangle_{0n} \otimes |\rho_0\rangle_{-0n} \\ &+c_1|1^0 1^n\rangle_{0n} \otimes |\rho_1\rangle_{-0n} \\ &+c_+\big(|0^0 1^n\rangle_{0n} + |1^0 0^n\rangle_{0n}\big) \otimes |\rho_+\rangle_{-0n}\end{aligned}$$

Secondly, as 0 and n are seen as an unique qubit, we need to have

$$\big(\langle 0^0 1^n| + \langle 1^0 0^n|\big)|\phi\rangle_{0n} = 0$$

Which can be achieved only when

$$c_+ = 0$$

Finally, we can see that this last constraint is violated by all basis states for $Sym^{n+1}\mathbb{C}^2$ except $|00...0\rangle$ and $|11...1\rangle$, which is what we wanted (after merging qubits 0 and n). □

4.4.4 Figure Eights are not PRODSAT

Now, instead of having a single loop, let's consider two of them, with at least a qubit in common. We will refer to these graphs as *figure eights*, denoted E_k.

Definition 26 (Figure Eight)**.**
We call "figure eight" of order k a graph E_k defined by

(i) A cycle C_k

(ii) A "cross bar" between v_1 and $v_{k/2-1}$.

Moreover, we can see that the automorphism group of such a graph is just the graph and the permutation

$$\begin{cases} v_i \leftrightarrow v_{k/2-i} \\ v_{l/2+i} \leftrightarrow v_{k-i} \end{cases} \quad i = 0, \dots, \lfloor \frac{k}{4} \rfloor$$

Which yields

$$|Aut(E_k)| = 2$$

Now if Laumann and al. decided to study this particular subgraph, it is because of the following property.

Theorem 24 (Figure Eights)**.**
A figure eight comprising two loops with n and m qubits has no satisfying product state.

Proof. Let's refer to our two loops as O_1 and O_2, and denote the qubit they have in common as q_0. We can see that the above reasoning fails to provide a satisfying product state, since in order for our two loops to be consistent, we need two satisfy the two constraints

- $\Big(\prod_{e\in O_1} T_e\Big)|0^0\rangle = \lambda_0|0^0\rangle$

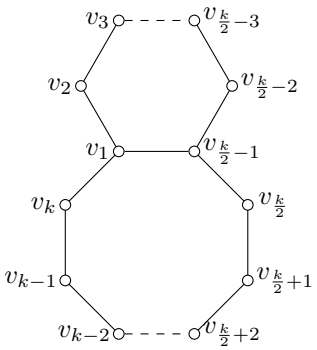

Figure 4: The figure eight E_k

- $\Big(\prod_{e\in O_2} T_e\Big)|0^0\rangle = \mu_0|0^0\rangle$

Now, since it is the same for $|1^0\rangle$, this would imply that

- $\mu_0\Big(\prod_{e\in O_1} T_e\Big)|0^0\rangle = \lambda_0\Big(\prod_{e\in O_2} T_e\Big)|0^0\rangle$
- $\mu_1\Big(\prod_{e\in O_1} T_e\Big)|1^0\rangle = \lambda_1\Big(\prod_{e\in O_2} T_e\Big)|1^0\rangle$

which is false with probability 1. □

Now, the interesting question here is actually to find a lower bound on the total energy for such a figure, since in order to call the problem UNSAT, we need to satisfy the promise

$$\langle\psi|\mathcal{H}|\psi\rangle \in O(n^{-b})$$

for some fixed b. In won't go too much into details but with the use of specific projectors, we can prove that the energy of E_k is lower bounded by an inverse polynomial in k.

4.4.5 Phase Transition for 2QSAT

The next step in the study of random satisfiability concerns the existence of what we call *phase transitions* in the space of the solutions. A phase should be understood as an interval in which all instances of the problem behave in a certain way. There are therefore a lot of different phases that occur when we change to value of α, but I will here only talk about the SAT/UNSAT phase.

Trees & Cycles

As we have seen in Section 4.3.4, before probability $\alpha_{gc} = \frac{1}{2}$, the structure of the graph is such that we can only find trees and single loops. Provided that we just saw that these two subgraphs were PRODSAT, we are able to draw the following conclusion

Theorem 25. *At clause density $\alpha \geq \frac{1}{2}$, $H \in$ 2QSAT.*

Figure Eights

Following our description in chapter 4.4.4, there is a major problem when trying to apply Theorem 19 : since a figure eight E_k contains k vertices and $k+1$ edges, we can only use this theorem provided that

$$p = cn^{-k/(k+1)} \rightarrow \alpha = \frac{c}{2} n^{1/(k+1)}$$

which is bad because our α scales with n and $n \rightarrow \infty$. Such a limitation can be overcome if we allow k and r to grow with n as long as $k \ll n$. At this regime, the threshold probability for the apparition of figure eights become $p = cn$ which corresponds to α constant. Another reason to allow k to grow with n is that it then transforms the lower bound of the energy of a figure eight into a value that is polynomial in n, which is required in our definition.

Having said this and according to Theorem 19, the distribution of X_{E_k} in a random is given by a Poisson distribution of parameter

$$\lambda = \frac{c^k}{2}$$

Since $p = \frac{c}{n} = \frac{2\alpha}{n}$, we can transform this into

$$\lambda = \frac{(2\alpha)^k}{2}$$

which, according to our regime $k \rightarrow \infty$, behaves in the following way.

$$\lim_{k\rightarrow\infty} \lambda = \begin{cases} 0 & \alpha < \frac{1}{2} \\ \frac{1}{2} & \alpha = \frac{1}{2} \\ +\infty & \alpha > \frac{1}{2} \end{cases}$$

Finally, we can use the first and second moment methods[27] to conclude that

(i) If $\alpha < \frac{1}{2}$, then

$$Pr[X_{E_k} = 0] = 1$$

(ii) If $\alpha > \frac{1}{2}$, then

$$Pr[X_{E_k} \geq 1] = 1$$

which provides the following result

Theorem 26. *At clause density $\alpha > \frac{1}{2}$, the probability of $H \in$ 2QSAT is 1.*

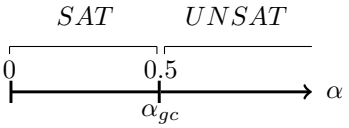

Figure 5: Phase Transition for 2QSAT

When comparing to the classical case, we find that the phase transition there is given by $\alpha_{2SAT} = 1$. The corresponding physical meaning is that the classical satisfiability problem is more *constrained* as its quantum equivalent.

[27] See Appendix C

4.5 Random 3QSAT

4.5.1 Upper Bounds on Satifiability

The phase transitions in 3QSAT are currently not understood as good as those in 2QSAT, so I will just try and recap some bounds that Laumann and al. were able to provide.

First of all, as in the $k = 2$ case, we can ensure the fact that no unsatisfiable subgraph appears before the giant component, therefore

Theorem 27. *At clause density $\alpha < \alpha_{gc} = 0.17$, $H \in$ 3QSAT.*

This result is not very interesting, since the apparition of the giant component quickly converges to 0. A first interesting result though can be provided using a generalization of Bravyi's Transfer Matrix for 3-local projectors, where one can propagate a clause on the neighbours while satisfying the constraint, which ensures the satisfiability of the problem until the hypercore emerges.

Theorem 28. *At clause density $\alpha < \alpha_{hc} = 0.81$, $H \in$ 3QSAT.*

A year later, they were able to provide an even sharper bound, still restricting the problem to product states only.

Theorem 29. *At clause density $\alpha < \alpha_{ps} = 0.92$, $H \in$ 3QSAT.*

4.5.2 Lower bounds on Unsatisfiability

When now approaching the problem from the other side and trying to provide a lower bound on the unsatisfiability, the results are the following.

Theorem 30. *At clause density $\alpha > \alpha_{wb} = 7.49$, $H \notin$ 3QSAT.*

Theorem 31. *At clause density $\alpha > \alpha_{cav}+ = 4.26$, $H \notin$ 3QSAT.*

Theorem 32. *At clause density $\alpha > \alpha_c^+ = 3.59$, $H \notin$ 3QSAT.*

Finally, the overall picture of our current knowledge in the matter is the following.

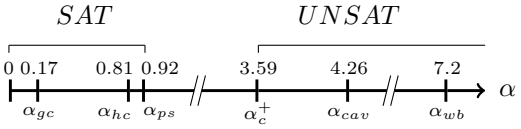

Figure 6: Phase Transition for 3QSAT

4.5.3 Product Versus Entangled

The analysis of this problem raises an interesting question : does the transition occurs at the same moment for QSAT and PRODSAT ? Since the space of entangled states is much bigger than that of product states, forbidding such states should make the problem harder. In fact, if the transition occurs later in normal QSAT it could mean that entangled states are more appropriate to lower the energy state of a system, which is an interesting point.

However, we can see that in 2QSAT it is not the case, since we are able to find product states up until $\alpha = \frac{1}{2}$. But in 3QSAT, the question remains open.

Conclusion

This exploration now coming to an end, I would like to first summarize the major points of what I learned throughout this project.

First of all, the idea of computation in quantum information theory is closely related with the notion of Hamiltonian which, through Schrödinger's equation, predicts the time evolution of a given system. This phenomenon, which has no classical counterpart, was used by Feynmann to build non-conventional computing models such as the Hamiltonian Computer, which questions our way of thinking about the true nature of computation.

These ideas led both physicists and computer scientists to try and use quantum mechanics and the principle of superposition in order to define computing models such as quantum circuits. These systems appear to provide exponential speed-up in several major problems of computer science, but it is still unknown at this point if such a improvement can be realized on any classical algorithm.

A good place to start then was the study of constraint satisfiability, since it is one of the problems that classical computers are striving to efficiently solve. What I've learned during this study was that this problem can be adapted to quantum computing, into two problems called the Local Hamiltonian and Quantum SAT.

When studying these two problems in particular, I found that they have similar properties when run on quantum computers than MAXSAT and SAT on a classical one. This leads to a hierarchy of quantum classes that looks a lot like the classical, probabilistic one. Moreover, all these problems are simply special cases of the Local Hamiltonian problem.

In the last chapter, Random Quantum SAT paved the way towards the study of Erdös-Rényi random graphs, which have the particular property of having multiple threshold points around which the behaviour completely changes in a very brief way. Satisfiability being in fact a graph property, it exhibits the same behaviour as random graphs with respect to the so-called phase transitions. This SAT/UNSAT phase transition is currently known for 2QSAT but still remains a mystery for higher values of k.

Finally, I would like to conclude by providing below some interesting questions that I acquired along the way, which I would have liked to study if I had more time but that I will here simply left open.

On the question of PRODSAT vs QSAT, if we ever find that cases exist where the only satisfying solution is an entangled state, and that this entangled state is global with respect to the system, how can we ever expect to use locally applied algorithms based on unit clause propagation such as DPLL in order to find a proper satisfying entangled state ?

Also, coming back to what I said in the introduction about the computation of nature, there is a possibility (in particular if QMA $\neq$ BQP) that constraint satisfaction could require super-polynomial time to be solved, even on a quantum computer. Therefore, I'm still wondering why and how nature manages to efficiently solve these problems while we can't.

Another interesting question to explore concerns the possibility of running 3SAT in polynomial-time

on a quantum computer. Now, since it is not proven that NP is in BQP, there exists no algorithm to this day, but it would be interesting to see what attempts have been made and what are the limitations when one tries to do so.

Last but not least, another path which I would have liked to explore is the question of MAXSAT versus SAT, and QSAT versus LH. Why are QSAT and SAT easy to solve up until 2, while the other two already becomes hard at $k = 2$? I briefly explained that it had to do with the fact that they were optimization problems, but this answer is not yet very satisfactory to me, since I haven't seen this concept in action.

Moreover, the fact that both classical and quantum interpretation of SAT require super-polynomial time to solve leads to the potential existence of an irreducible intrinsic difficulty, a *golden nugget* that one has yet to discover.

References

[1] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *Journal of Computer and System Sciences 36*, pages 254–276, 1985.

[2] Béla Bollobàs. *Random Graphs (2nd edition)*. Cambridge University Press, 2001.

[3] S. Bravyi, C. Moore, and A. Russell. Bounds on the quantum satisfiability threshold. 2009.

[4] Sergey Bravyi. Efficient Algorithm for a Quantum Analogue of 2-SAT. 2006.

[5] Mathilde Duclos. NP-completeness: Some Reductions.

[6] Richard P. Feynmann. Quantum Mechanical Computers. 1985.

[7] D. Gosset and D. Nagaj. Quantum 3-sat is QMA1-complete. *54TH annual symposium on Foundations of Computer Science*, pages 756–765, 2013.

[8] J. Kempe, A. Kitaev, and O. Regev. The complexity of the local hamiltonian problem. 2004.

[9] J. Kempe and O. Regev. 3-Local Hamiltonian is QMA-complete. *Quantum Computation and Information, Vol. 3*, pages 258–264, 2003.

[10] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, 2002.

[11] C. R. Laumann, A. M. Läuchli, R. Moessner, A. Scardicchio, and S. L. Sondhi. Phase Transition and Random Quantum Satisfiability. 2009.

[12] C. R. Laumann, A. M. Läuchli, R. Moessner, A. Scardicchio, and S. L. Sondhi. On product, generic and random generic quantum satisfiability. *Phys. Rev. A 81, 062345*, 2010.

[13] I. Mironov and L. Zhang. Applications of SAT Solvers to Cryptanalysis of Hash Functions. 2006.

[14] Daniel Nagaj. *Local Hamiltonians in Quantum Computation*. PhD thesis, Massachusetts Institute of Technology, 2008.

[15] H. Nishimura and M. Ozawa. Perfect Computational Equivalence between Quantum Turing Machines and Finitely Generated Uniform Quantum Circuit Families. *Quantum Information Processing 8*, pages 13–24, 2008.

Appendix

A Quantum Computing

The purpose of this short introduction is to provide some basic notions that might help you understand the content in this study. As a result, it will not be an exhaustive overview, and only related topics will be presented. In particular, I will not always mention all the physical phenomena and experiments involved, but rather focus on the mathematical point of view. The material presented here include

1. A little bit of background in linear algebra
2. The definition of a quantum bit and the extension to multi-qubit states.
3. An overview of some quantum operators
4. The measurement of quantum states

Finally, I would like to recommend [10] as a more precise and exhaustive introduction to this subject.

Introduction

When thinking about information theory problems, a common practice is to abstract physical phenomenons as much as possible. It is in fact much more convenient to work with abstract boolean circuits or Turing machines, as we are able to completely eliminate the problem of the physical realization of these machines, knowing that it is equivalent.

As we go further and further towards miniaturization though, the abstraction that we have nowadays may need to change, in the same way that when it comes to using particles instead of macroscopic objects, quantum physics takes over classical theories.

Having said that, it is important to first understand that classical computer science is based upon several implicit assumptions about the underlying physical phenomena used to create a computer. These include

1. Information is stable in time.
2. Information is infinitely reproducible.
3. Information can be read without being altered.

As you will see, these assumption hold only when we consider macroscopic systems, but not when it comes to microscopic entities upon which quantum computing is based. Dropping these assumptions may sound foolish in the first place, but it is the price to pay in order to unleash the power of quantum interactions.

Linear Algebra

Where the classical theory uses boolean variables and boolean functions, quantum computation is written in vectors and matrices. Therefore, it is important to first recall some basic notions of linear algebra, such as the definition of a *Hilbert Space*.

Definition 27 (Hilbert Space)**.**
An Hilbert space $\mathcal{H}$ is a real or complex vector space equipped with a norm function $\|\cdot\|$ defined in terms of the inner product $\langle\cdot,\cdot\rangle$. It has the following properties

(i) $\langle x, y\rangle = \overline{\langle y, x\rangle}$

(ii) $\langle x, x\rangle \geq 0$

(iii) $\langle x, x\rangle = 0 \Leftrightarrow x = 0$

(iv) $\|x\| = \sqrt{\langle x, x\rangle}$

Remark 28. *Properties (i), (ii) and (iii) follow from the definition of Banach spaces, whereas (iv) is an extra requirement.*

For our purpose, we will only use complex Hilbert spaces $\mathbb{C}^n$ where the inner product is defined as the usual *dot product*

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

Finally, an important notion that we will use is the *Tensor Product*. It operates on multiple Hilbert spaces in the following way

Definition 28 (Tensor Product)**.** *The tensor product of two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ is another Hilbert space denoted as*

$$\mathcal{H}_1 \otimes \mathcal{H}_2$$

The dimension of $\mathcal{H}_1 \otimes \mathcal{H}_2$ is $m \times n$, and all its components are built from $\vec{v} = (v_1, \ldots, v_n) \in \mathcal{H}_1$ and $\vec{w} = (w_1, \ldots, w_m) \in \mathcal{H}_2$ as

$$\vec{v} \otimes \vec{w} = (v_1 w_1, v_1 w_2, \ldots v_1 w_m, v_2 w_1, v_2 w_2, \ldots v_2 w_m, \ldots,$$

Example 2. *If we take $\vec{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\vec{w} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, we get*

$$\vec{v} \otimes \vec{w} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \times \begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ 2 \times \begin{pmatrix} 3 \\ 4 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

This tensor product operator is also applicable in the same way to matrices. This time, the dimension of the matri

Example 3. *If we have $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$, then the tensor product is*

$$A \otimes B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 2 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ 3 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 4 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{pmatrix}$$

Dirac Notation

When doing quantum physics, we often need to use complex vectors the usual notation for linear algebra that I used in the previous chapter is not very convenient. For this reasons, physicists have come to use a different one, called the *Dirac Notation*[28], which I will use in this study and briefly introduce here.

In this notation, a vector $\vec{v}$ will be denoted a $|v\rangle$, and its complex-conjugate $\vec{v}^\dagger$ will be denoted as $\langle v|$. If we use an Hilbert space $\mathcal{H} = \mathbb{C}^n$ equipped with the usual inner product and with canonical base vectors $\{\vec{b_1}, \vec{b_2}, \ldots, \vec{b_n}\}$, we have

$$\vec{v} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = a_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \cdots + a_n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

can now simply be written as

$$|v\rangle = \sum_{i=1}^{n} a_i |b_i\rangle$$

In the same way,

$$\vec{v}^\dagger = \big(\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_n\big) = \overline{a}_1 \cdot \vec{b_1}^\dagger + \cdots + \overline{a}_n \cdot \vec{b_n}^\dagger$$

becomes

$$\langle v| = \sum_{i=1}^{n} \overline{a}_i \langle b_i|$$

Finally, we are able to define the inner product $\langle x, y\rangle = \vec{x} \cdot \vec{y}$ as $\langle x|y\rangle$. If we express that in our example, it is easy to verify that

$$\begin{aligned} \langle x|y\rangle &= \Big(\sum_{i=1}^{n} \overline{x}_i \langle b_i|\Big)\Big(\sum_{j=1}^{n} y_j |b_j\rangle\Big) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{x}_i y_i \langle b_i|b_j\rangle \\ &= \sum_{i=1}^{n} \overline{x}_i y_i \end{aligned}$$

The second equality comes from the fact that, since $|b_i\rangle$'s are all orthogonal, $\langle b_i|b_j\rangle = 1$ if and only if $i = j$.

Quantum Bit

Definition 29 (Quantum Bit)**.**
A quantum bit (qubit) is a vector $|\psi\rangle$ that lives in an Hilbert space $\mathcal{H} = \mathbb{C}^2$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \in \mathcal{H}$$

with the additional requirements

[28] Also referred as "Bra-Ket" notation

(i) $|\alpha^2| + |\beta^2| = 1$

(ii) No global phase

Remark 29. *Since two complex numbers correspond to four degrees of freedom and we add two one-dimensional restrictions, every qubit has 2 degrees of freedom.*

Remark 30. *Following the previous remark, it is sometimes easier to visualize qubits as a tuple of angles* $\langle \theta, \phi \rangle$*,* $\theta \in [0, \frac{\pi}{2}], \phi \in [0, 2\pi]$ *where*

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

Example 4. $|\psi_1\rangle = 2|0\rangle + |1\rangle$ *is not a valid state because* $2^2 + 1^2 = 5 \neq 1$.

Example 5. $|\psi_2\rangle = \frac{i}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$ *is a valid state but it has a global phase of* i*. We can thus use the equality* $\frac{i}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle = i(\frac{1}{2}|0\rangle - \frac{\sqrt{3}i}{2}|1\rangle)$ *and write* $|\psi_2\rangle = \frac{1}{2}|0\rangle - \frac{\sqrt{3}i}{2}|1\rangle$ *instead.*

Example 6. $|\psi_3\rangle = |0\rangle$ *is a valid state*

Example 7. $|\psi_4\rangle = |1\rangle$ *is a valid state*

Example 8. $|\psi_5\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ *is a valid state*

Example 9. $|\psi_6\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ *is a valid state*

Example 10. $|\psi_7\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$ *is a valid state*

Composition

Using a qubit is great, but at some point we want to be able to build multi-qubit states. Knowing that a 1-qubit state lives in some Hilbert space $\mathcal{H} = \mathcal{B}$, we ask what is the state of a n-qubit state.

The vectorial operation that corresponds to this operation is called the *tensor product* and is defined as follows

Definition 30 (Tensor Product)**.** *Given two vector spaces* V *and* W*, the tensor product* $V \otimes W$

Example 11. *The tensor product of* $|\psi\rangle = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}$ *and* $|\phi\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$ *is*

$$\begin{aligned}|\psi\rangle \otimes |\phi\rangle &= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}}|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} \cdot \frac{-1}{\sqrt{2}}|0\rangle \otimes |1\rangle + \frac{-i}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}}|1\rangle \otimes |0\rangle + \frac{-i}{\sqrt{2}} \cdot \frac{-1}{\sqrt{2}}|1\rangle \otimes |1\rangle \\ &= \frac{1}{2}|0\rangle \otimes |0\rangle - \frac{1}{2}|0\rangle \otimes |1\rangle - \frac{i}{2}|1\rangle \otimes |0\rangle + \frac{i}{2}|1\rangle \otimes |1\rangle\end{aligned}$$

If we now simplify the notation, we finally have

$$|\psi\rangle \otimes |\phi\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{i}{2}|10\rangle + \frac{i}{2}|11\rangle$$

Example 12. *More generally, the tensor product of* $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ *and* $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$ *is*

$$|\psi\rangle \otimes |\phi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$$

Such a state is always valid since

$$|\alpha\gamma|^2 + |\alpha\delta|^2 + |\beta\gamma|^2 + |\beta\delta|^2 = (|\alpha|^2 + |\beta|^2)(|\gamma|^2 + |\delta|^2) = 1$$

Finally, an important question arises when using more than one qubit. What about the following state ?

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

You can search as long as you want, there is no way to find two 1-qubit states $|\psi_1\rangle \otimes |\psi_2\rangle = |\psi\rangle$. What should we do about it then ? Well, these states exist in nature and are called *entangled states*.

These states are quite difficult to apprehend at first glance but are in practice very useful in practice. In fact, we can show that, where the global state is completely defined, if we measure each one of the qubit, we cannot get any useful information about the global one.

Unitary Operators

Now that we know how to construct quantum states, let us focus on how to interact with them. In classical computation, we generally represent the evolution of boolean states as a series of *logic gates*, which are nothing but binary functions

$$f : \mathbb{B}^n \to \mathbb{B}^m$$

Among the most famous ones, we can find the following

- $NOT : x \to \overline{x} = \begin{cases} 1 & \textit{if } x = 0 \\ 0 & \textit{if } x = 1 \end{cases}$
- $ERASE : x \to 0$
- $AND : (x, y) \to x \wedge y = \begin{cases} 1 & \textit{if } x = 1 \textit{ and } y = 1 \\ 0 & \textit{otherwise} \end{cases}$
- $OR : (x, y) \to x \vee y = \begin{cases} 1 & \textit{if } x = 1 \textit{ or } y = 1 \\ 0 & \textit{otherwise} \end{cases}$
- $XOR : (x, y) \to x \oplus y = \begin{cases} 1 & \textit{if } y \neq x \\ 0 & \textit{otherwise} \end{cases}$

Having said that, the idea now is to use a similar model to represent quantum evolution of states. What Deutsch tried to do in 1985 was to define a series of *quantum gates* that could be applied in the same way that classical are. Due to the linear algebraic property of quantum computation, quantum gates will not be represented as boolean functions, but as linear applications.

$$U : \mathcal{B}^{\otimes n} \to \mathcal{B}^{\otimes m}$$

Since $\mathcal{B} = \mathbb{C}^2$, U can bee seen as a $2^n \times 2^m$ complex matrix. Moreover, for reasons that I will not provide here, the evolutions of any quantum state must be *unitary*.

Definition 31 (Unitary Matrix)**.** *A complex square matrix U is unitary if its conjugate transpose $U^\dagger$ is also its inverse. It means that*

$$U^\dagger U = UU^\dagger = \mathbb{I}$$

Remark 31. *Unitary matrices preserve the norm and are reversible. It preserves the inner product*

Remark 32. *The real analogue of a unitary matrix is an orthogonal matrix.*

The second point here is a bit of a problem. If we look back at our classical gates, a lot of them[29] are not reversible at all. This is due to *information loss* that happens when one erases or "forgets" a value. For example, if one knows that the output of an AND is 0, it is not possible to know if the input was $0 \wedge 0$, $0 \wedge 1$ or $1 \wedge 0$.

In practice, it means two things. First, all quantum operators must have the same number of input and output bits. Second, there are several operations that we will simply not be allowed to create. Here are some of the most basic unitary operators

- The Identity $\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1|$
  where $\mathbb{I}|b\rangle = |b\rangle$ and eigenvectors are $|0\rangle$ with $\lambda = 1$ and $|1\rangle$ with $\lambda = 1$.
- The 3 *Pauli Matrices*
  1. $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$
     where $X|b\rangle = |\bar{b}\rangle$ and eigenvectors are $\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)$ with $\lambda = 1$ and $\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$ with $\lambda = -1$.
  2. $iY = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = |0\rangle\langle 1| - |1\rangle\langle 0|$
     where $iY|b\rangle = (-1)^{b+1}|\bar{b}\rangle$ and eigenvectors are $\frac{1}{\sqrt{2}}\big(|0\rangle + i|1\rangle\big)$ with $\lambda = i$ and $\frac{1}{\sqrt{2}}\big(|0\rangle - i|1\rangle\big)$ with $\lambda = -i$.
  3. $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$
     where $iY|b\rangle = (-1)^{b}|b\rangle$ and eigenvectors are $|0\rangle$ with $\lambda = 1$ and $|1\rangle$ with $\lambda = -1$.
- The *Hadamard Gate* $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- The Rotation Matrices
  1. $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} = |0\rangle\langle 0| + e^{i\frac{\pi}{4}}|1\rangle\langle 1|$
  2. $S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix} = |0\rangle\langle 0| + e^{i\frac{\pi}{2}}|1\rangle\langle 1|$

Finally, before going to the next chapter, there is an important concept in quantum information theory that we are now able to understand, and which is called *The No-Cloning Principle*. Basically, it says that it is not possible to copy a n-qubit states $|\psi\rangle$ into another n extra qubits (initially in state $|0\rangle$). Mathematically speaking, this boils down to having a quantum operator

$$U_{copy} : \mathcal{B}^{\otimes 2n} \to \mathcal{B}^{\otimes 2n}$$
$$U_{copy}(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle$$

[29] In this case, ERASE, AND, OR and XOR

Measurements

One of the main difficulties in the building and use of a quantum computer comes from the *measurement problem*[30] in quantum mechanics. It depicts the physical uncertainty regarding the measurement of different properties of a quantum state. An early and very famous example is *Heisenberg's Uncertainty Principle*, which states that one cannot observe the *momentum* (p) and *position* (x) of a particle at the same time with infinite precision.

$$\Delta p \Delta x \geq \frac{\hbar}{2\pi}$$

This inequality means that if one wants to measure an exact position $\Delta x \to 0$, then the momentum goes more and more uncertain as $\Delta p \to \infty$, and vice versa.

Another intriguing phenomenon is that, before measuring the particle's position, it is in a superposition of states, which means that it is everywhere at the same time. However, once we have observe this particle, we force it to "choose" only one possible location *at random*[31] and thus we destroy the superposition.

For our purpose, this quantum mechanical property imply that it will not be possible to observe the whole superposition of a quantum bit. In fact, we can only observe states that are *orthogonal*[32], so we will first need to *project* the state we want to observe onto an orthogonal basis and then measure it. This also signify that any measurement will alter the quantum state, as we force it to take one of the possible states.

When observing a state $|\psi\rangle$ using a set of states $\{|\phi_1\rangle, \dots |\phi_n\rangle\}$, the probability of measuring the output state $|\phi_i\rangle$ is given by *Born's Law* and is equal to

$$|\langle\phi_i|\psi\rangle|^2$$

Having said this, let us see some example states and their measurement in different orthogonal basis.

Example 13. *The measurement of $|\psi_1\rangle = |0\rangle$ in the basis $\{|0\rangle, |1\rangle\}$ will yield the output*

$$\begin{cases} |0\rangle & \textit{with probability } 1 \\ |1\rangle & \textit{with probability } 0 \end{cases}$$

However, if we measure it using the base $\{\frac{|0\rangle+|1\rangle}{\sqrt{2}}, \frac{|0\rangle-|1\rangle}{\sqrt{2}}\}$, we will get

$$\begin{cases} \frac{|0\rangle+|1\rangle}{\sqrt{2}} & \textit{with probability } \frac{1}{2} \\ \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \textit{with probability } \frac{1}{2} \end{cases}$$

Finally, if we measure it in the base $\{\frac{|0\rangle+i|1\rangle}{\sqrt{2}}, \frac{|0\rangle-i|1\rangle}{\sqrt{2}}\}$, the outcome is also

$$\begin{cases} \frac{|0\rangle+i|1\rangle}{\sqrt{2}} & \textit{with probability } \frac{1}{2} \\ \frac{|0\rangle-i|1\rangle}{\sqrt{2}} & \textit{with probability } \frac{1}{2} \end{cases}$$

[30] Read `http://www.pitt.edu/~jdnorton/teaching/HPS_0410/chapters/quantum_theory_measurement/index.html` for more information

[31] According to a probability function defined by the particle state before the measurement

[32] Two states $|\psi_1\rangle$ and $|\psi_2\rangle$ are orthogonal if $\langle\psi_1|\psi_2\rangle = 0$.

Example 14. *The measurement of* $|\psi_2\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, $|\psi_3\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$ *and* $|\psi_5\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}}$ *in the basis* $\{|0\rangle, |1\rangle\}$ *will all yield the output*

$$\begin{cases} |0\rangle & \textit{with probability } \frac{1}{2} \\ |1\rangle & \textit{with probability } \frac{1}{2} \end{cases}$$

Example 15. *The measurement of* $|\psi_6\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$ *in the computational basis* $\{|00\rangle, |00\rangle, |00\rangle, |00\rangle\}$ *will give the outcome*

$$\begin{cases} |00\rangle & \textit{with probability } \frac{1}{4} \\ |01\rangle & \textit{with probability } \frac{1}{4} \\ |10\rangle & \textit{with probability } 0 \\ |11\rangle & \textit{with probability } \frac{1}{2} \end{cases}$$

Given this probabilistic nature of quantum measurement, the usual way of getting around it is to repeat the computation several times and approximate the initial superposition using the distribution of results.

B Rank and Submatrices theorem

Definition 32 (Submatrix)**.**
Let A *be a* $n \times n$ *matrix. A submatrix of* A *is any matrix that is obtained by removing* $n - k$ *rows and columns from* A*. We will denote the set of* $k \times k$ *submatrices of* A *as* $A_{|k\times k}$*.*

Theorem 33.
Let A *be a* $n \times n$ *non-zero matrix, and* $m = max\{\ k \mid \exists B \in A_{|k\times k}$ *such that* $det(B) \neq 0\ \}$ *(the biggest* k *such that there is a* $k \times k$ *submatrix with non-zero determinant). Then*

$$rank(A) = m$$

Conversely,

$$rank(A) = m \implies \exists B \in A_{|m\times m} \textit{ such that } det(B) \neq 0$$

C Probabilities

Definition 33 (Poisson)**.**
The poisson distribution with parameter λ *is a discrete probability distribution defined by*

$$\mathbb{P}[k] = \frac{\lambda^k}{k!} e^{-\lambda}$$

And the expectation of a random variable $X \sim P(\lambda)$ *is*

$$\mathbb{E}[X] = \lambda$$

Definition 34 (Convergence in Distribution)**.**
A sequence $(X_n)_{n\in\mathbb{N}}$ *of random variables is said to converge in distribution (or weakly) to a random variable* X *if*

$$\lim_{n\to\infty} F_n(x) = F(x)$$

for every number $x \in \mathbb{R}$ *at which* F *is continuous, and where* F_n *and* F *are the cumulative distribution functions of* X_n *and* X *respectively. It is denoted as* $X_n \xrightarrow{D} X$*.*

Definition 35 (Convergence in Probability)**.**
A sequence $(X_n)_{n\in\mathbb{N}}$ of random variables is said to converge in probability to a random variable X if $\forall \varepsilon > 0$

$$\lim_{n\to\infty} P(|X_n - X| \geq \varepsilon) = 0$$

It is denoted by $X_n \xrightarrow{P} X$.

Definition 36 (Almost Sure Convergence)**.**
A sequence $(X_n)_{n\in\mathbb{N}}$ of random variables is said to converge almost surely (or strongly) to a random variable X if

$$P(\lim_{n\to\infty} X_n = X) = 1$$

It is denoted by $X_n \xrightarrow{a.s.} X$

Theorem 34 (Relations between the 3)**.**
For a sequence $(X_n)_{n\in\mathbb{N}}$ of random variables, and another random variable X, we have

$$X_n \xrightarrow{a.s.} X \implies X_n \xrightarrow{P} X \implies X_n \xrightarrow{D} X$$

Which makes the Almost Sure Convergence the strongest of these ones.

Theorem 35 (Law of Large Numbers)**.**
For a probability distribution X with mean μ, let

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Then the Strong Law of Large Numbers says that

$$\overline{X} \xrightarrow{a.s.} \mu$$

Theorem 36 (Markov's Inequality)**.**
Let X a nonnegative and integrable random variable and a constant $a > 0$, then

$$\mathbb{P}[X \geq a] \geq \frac{\mathbb{E}[X]}{a}$$

Theorem 37 (Chebyshev's Inequality)**.**
Let X a random variable with expected value μ and finite variance σ^2, then for any $\alpha > 0$

$$Pr[|X - [X]| \geq \alpha] \leq \frac{Var[X]^2}{\alpha^2}$$

Theorem 38 (First Moment Method)**.**
Let X_n be a sequence of integer-valued random variables for $n \in \mathbb{N}$. If $\lim_{n\to\infty} \mathbb{E}[X_n] = 0$, then

$$\lim_{n\to\infty} Pr[X_n = 0] = 1$$

Proof. This formula is just a special case of Markov's Inequality with $a = 1$.

$$Pr[X_n = 0] = 1 - Pr[X_n \geq 1] = 1 - \mathbb{E}[X_n] \to 1$$

□

Theorem 39 (Second Moment Method)**.**
Let X_n be a sequence of integer-valued random variables for $n = 0, 1, \ldots$. If $\lim_{n\to\infty} Var[X_n]/\mathbb{E}[X_n]^2 = 0$, then

$$\lim_{n\to\infty} Pr[X_n \geq 1] = 1$$

Proof. This time, this is just a special case of Chebyshev's Inequality with $\alpha = \mathbb{E}[X_n]$.

$$Pr[X_n = 0] \leq Pr[|X_n - [X_n]| \geq [X_n]] \leq \frac{Var[X]^2}{\mathbb{E}[X_n]^2} \to 0$$

□

D Stirling's Formula

Theorem 40 (Stirling's Formula)**.**

$$\lim_{n\to+\infty} \frac{n!}{\sqrt{2\pi n}(n/e)^n}$$

In other words

$$n! \sim \sqrt{2\pi n}(n/e)^n$$

E Graph Properties

Definition 37 (Average Degree)**.**
The average degree of a graph $G = (V, E)$ ($n = |V|, m = |E|$), is given by

$$\tilde{d}(G) = \frac{2m}{n}$$

Definition 38 (Maximal Average Degree)**.**
In the same way as above, the maximal average degree of a graph G is given by

$$\hat{d}(G) = \max_{H\subset G}\{\tilde{d}(H)\}$$

Definition 39 (Strictly Balanced Graph)**.**
A Graph G is said to be balanced if

$$\tilde{d}(G) = \hat{d}(G)$$

Moreover, G is strictly balanced when no subgraph has an equal average degree, i.e.

$$\forall H \subset G \quad \tilde{d}(H) = \tilde{d}(G) \implies H = G$$

Definition 40 (Automorphism)**.**
An automorphism G' of a graph G is a permutation $\sigma : V \to V$ such that

$$(u, v) \in E \implies (\sigma(u), \sigma(v)) \in E$$

Definition 41 (Induced Subgraph)**.**
A subgraph $H \subset G$ is called an induced subgraph (on the vertex set $S \subset V$) if

$$\forall u, v \in S \quad (u, v) \in E_G \Leftrightarrow (u, v) \in E_H$$

We will often use the following notation to denote induced subgraphs

$$H = G[S]$$

Definition 42 (Spanning Subgraph)**.**
A subgraph $H \subset G$ is called a spanning subgraph if it has the same vertex set as G.

F Hypergraphs

Definition 43 (Hypergraph)**.**
A Hypergraph is a pair $H = (X, E)$, where X is the set of elements, or vertices, and $E \subset \mathcal{P}(X) \setminus \emptyset$ is the set of hyperedges. More formally, we will define these sets as

- $X = \{x_i \mid i \in I_v\}$
- $E = \{e_i \mid i \in I_e,\ e_i \subset X\}$

Where I_v and I_e are the index sets of the vertices and the edge respectively.

Definition 44 (Size of a Hyperedge)**.**
The size of hyperedge e is equal to the number of vertices it connects, i.e. $|e|$.

Definition 45 (k-Uniform Hypergraph)**.**
A k-Uniform hypergraph is one such that all hyperedges have size k.