CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Approximability of Optimization Problems within an Approach of Quantum Computing

A dissertation submitted by

William de la Cruz de los Santos

For the degree of

Doctor of Computer Science

Supervisor

Guillermo Morales Luna

México D.F. March 2013

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Aproximación a problemas de optimización con un enfoque de cómputo cuántico

Tesis que presenta

William de la Cruz de los Santos

Para obtener el grado de

Doctor en Ciencias en Computación

Director de tesis:

Guillermo Morales Luna

México, D.F. Marzo de 2013

Abstract

Quantum Computing appeared about 30 years ago motivated by the ideas of Richard Feynman. He wondered whether it is possible to simulate a quantum system by means of a universal quantum machine or *quantum computer*. Although, there is no yet a practical implementation of a quantum computer, researches have done impressive theoretical results in the design of *quantum algorithms*, an important quantum algorithm is the Shor algorithm to factorize an integer number into its prime factors [76]. A quantum algorithm has as input an initial state, and then a series of unitary matrices are applied over the initial state in order to produce a final state, the output of the algorithm is obtained by performing a quantum measurement over the final state. This kind of quantum algorithms belong to the *Quantum Circuit Model* (QCM) [61].

Recently, it was proposed the *Adiabatic Quantum Computation* (AQC) [37, 35] that is based on the *Adiabatic Theorem* [58, 42] to approximate solutions of the Schrödinger equation. The design of an AQC algorithm involves the construction of a Hamiltonian that describes the behavior of the quantum system, this Hamiltonian is expressed as a linear interpolation of an *initial Hamiltonian* whose ground state is easy to compute, and a *final Hamiltonian* whose ground state corresponds to the solution of a given optimization problem. The Adiabatic Theorem asserts that if the time evolution of a quantum system described by a Hamiltonian is large enough, then the system remains close to its ground state. Thus, given an optimization problem, an AQC algorithm uses the Adiabatic Theorem to approximate the ground state of the final Hamiltonian that corresponds to the solution of the given optimization problem. The time complexity of an AQC algorithm is the minimum time that satisfies the Adiabatic Theorem.

AQC has been used to solve optimization problems, in [35] the authors claim that the optimization problem MAX-SAT can be solved in polynomial time complexity by an AQC algorithm. In [2] it was proved that QCM is equivalent to AQC. From the computational point of view it is important to compute the spectrum of the Hamiltonian in an AQC algorithm along the integration time in order to estimate its time complexity. In general, it is a hard problem to know the time complexity of an AQC algorithm.

We investigate the computational simulation of AQC algorithms for the MAX-SAT optimization problem, we propose a symbolic analysis of the AQC solution in order to understand the involved computational complexity of the AQC algorithms. This approach can be extended to others combinatorial optimization problems. The computational simulation of an AQC algorithm requires the construction of a matrix of dimension $2^n \times 2^n$ where n is the dimension of the quantum system, that in general corresponds to a sparse matrix, this matrix is constructed using matrix tensor products. We propose an efficient construction of the Hamiltonian for AQC algorithms that avoid the matrix tensor products.

The design of AQC algorithms has important consequences in its time complexity and also for its possible physical implementation. In Quantum Mechanics it is con-

venient to describe a Hamiltonian as an addition of local Hamiltonians i.e., Hamiltonians that only act on a subset of states in the quantum system. On the other hand, the *pseudo-Boolean optimization model* has been used to model combinatorial optimization problems into pseudo-Boolean maps. We propose a general scheme to design AQC algorithms based on pseudo-Boolean maps for combinatorial optimization problems, and we show that for a given optimization problem expressed in the pseudo-Boolean optimization model, then it is possible to construct an AQC algorithm with local Hamiltonians.

In [47] it was proved that NP-problems can be expressed in Second Order Logic (SOL). In [27] it was shown that all instances of graph problems expressed in Monadic Second Order Logic (MSOL) with bounded *treewidth* can be solved in polynomial time complexity. The algorithmic solution proposed in [27] is based on a *Dynamic Programming approach* over *tree-decompositions* of graphs [13, 18]. We show that every MSOL expression has associated pseudo-Boolean maps that can be obtained by expanding the given MSOL expression, and also can be reduced to quadratic forms. The equivalence between MSOL expressions and quadratic pseudo-Boolean maps can be considered as a general scheme to design AQC algorithms, since every quadratic pseudo-Boolean map can be optimized by an AQC algorithm. We also show a composition scheme for local Hamiltonians based on the dynamic programming approach over tree-decompositions of graphs.

Resumen

La *Computación Cuántica* apareció hace cerca de 30 años motivada por la ideas de Richard Feyman, quien se preguntaba si era posible simular un sistema cuántico por medio de una máquina cuántica universal o *computadora cuántica*. Aunque todavía no existe una implementación física de una computadora cuántica, se han hecho grandes progresos teóricos en el diseño de *algoritmos cuánticos*, por ejemplo el algoritmo cuántico de Shor para factorizar números enteros en sus factores primos [76]. Un algoritmo cuántico recibe como entrada un estado inicial, sobre el cual se aplican sucesivamente matrices unitarias, obteniendo así un estado final, la salida del algoritmo se obtiene llevando a cabo una *medición cuántica* sobre el estado final. Este tipo de algoritmos cuánticos pertenecen al Modelo de Circuitos Cuánticos (MCC) [61].

Recientemente, se propuso la *Computación Cuántica Adiabática* (CCA) [37, 35] que se basa en el *Teorema Adiabático* [58, 42] para aproximar soluciones de la ecuación de Schrödinger. El diseño de un algoritmo en CCA involucra la construcción de un hamiltoniano que describe el comportamiento del sistema cuántico, este hamiltoniano se expresa como una interpolación lineal de un *hamiltoniano inicial* cuyo estado firme sea fácil de calcular, y un *hamiltoniano final* cuyo estado firme corresponde a la solución de un problema de optimización dado. El Teorema Adiabático establece que si el tiempo de evolución de un sistema cuántico, descrito por un hamiltoniano, es lo suficientemente grande, entonces el sistema se mantiene cerca de su estado firme. Así, dado un problema de optimización, un algoritmo en CCA emplea el Teorema Adiabático para aproximar el estado firme del hamiltoniano final, que corresponde a la solución del problema de optimización. Se considera a la complejidad en tiempo de un algoritmo en CCA como al mínimo tiempo tal que se satisfaga el Teorema Adiabático.

La CCA se ha empleado para resolver problemas de optimización, en [35] los autores creen que el problema MAX-SAT puede ser resuelto en complejidad de tiempo polinomial por un algoritmo en CCA. En [2] se probó que el MCC es equivalente a la CCA. Desde el punto de vista computacional, es importante calcular el espectro del hamiltoniano a lo largo del tiempo de evolución de un algoritmo en CCA, esto ayudaría a conocer su complejidad en tiempo. En general, conocer la complejidad en tiempo de un algoritmo en CCA es un problema difícil [37].

Investigamos la simulación computacional de los algoritmos en CCA para el problema de optimización MAX-SAT, proponemos un análisis simbólico de la solución en CCA con el fin de entender la complejidad computacional de los algoritmos en CCA. Este enfoque se puede extender a otros problemas de optimización combinatorios. En la practica, la simulación computacional de un algoritmo en CCA requiere la construcción de una matriz de dimensión $2^n \times 2^n$ donde n es la dimensión del sistema cuántico, en general esta matriz corresponde a una matriz dispersa, y se construye usando el producto tensorial de matrices. Proponemos una construcción eficiente de los hamiltonianos en CCA para el problema MAX-SAT que evita el uso de productos tensoriales.

El diseño de algoritmos en CCA tiene consecuencias en su complejidad en tiempo y también en su posible implementación física. En la *Mecánica Cuántica* es conveniente describir un hamiltoniano como una suma de hamiltonianos locales i.e., hamiltonianos que actúan sobre un subconjunto de estados en el sistema cuántico. Por otro lado, el modelo de optimización de funciones pseudo-booleanas ha sido usado para modelar problemas de optimización combinatorios por medio de funciones pseudo-booleanas. Proponemos un esquema general para el diseño de algoritmos en CCA por medio de funciones pseudo-booleanas para problemas de optimización combinatorios. Probamos que para cada problema de optimización que se exprese en el modelo de optimización de funciones pseudo-booleanas, se puede diseñar un algoritmo en CCA con hamiltonianos locales.

En *Complejidad Descriptiva* la clase de problemas NP se puede describir como expresiones en la Lógica de Segundo Orden (LSO) [47]. En [27] se demuestra que todas las instancias de problemas sobre gráficas que se expresan en la Lógica Monádica de Segundo Orden (LMSO) con *ancho de árbol* acotado, se pueden resolver en complejidad de tiempo polinomial. La solución algorítmica propuesta en [27] se basa en un esquema de *programación dinámica* sobre descomposiciones en árbol de gráficas [13, 18]. Demostramos que cada expresión en LMSO tiene asociada funciones pseudo-booleanas que se obtienen expandiendo la expresión en LMSO, y que se pueden reducir a formas cuadráticas. Esta equivalencia entre expresiones en LMSO y funciones cuadráticas pseudo-booleanas se puede considerar como un esquema general para diseñar algoritmos en CCA, ya que cada función cuadrática pseudo-booleana puede ser optimizada por un algoritmo en CCA. Mostramos también un esquema de composición de hamiltonianos locales que se basa en el enfoque de programación dinámica sobre descomposiciones en árbol de gráficas.

Agradecimientos

Deseo agradecer sinceramente a mi asesor, al Dr. Guillermo Morales Luna por aceptar dirigir el presente trabajo y compartir sus conocimientos, así como su gran motivación en la investigación. Agradezco al Dr. Alán Aspuru-Guzik de la Universidad de Harvard, al Dr. Micho Durdevich Lucich del Instituto de Matemáticas en la Universidad Nacional Autónoma de México, y a los doctores Debrup Chakraborty y Sergio Víctor Chapa Vergara, ambos del CINVESTAV-IPN, por aceptar ser revisores de este trabajo, y por sus valiosos comentarios y observaciones.

A mis padres, les agradezco su apoyo y motivación por inculcarme los ideales y principios que rigen mi vida.

Sin dejar de lado a mis amigos y compañeros, les quiero dar las gracias por sus consejos y porque aún y cuando las distancias son largas siempre están ahí para dar consejos y nuevas ideas.

También agradezco al personal secretarial del departamento de Computación, Sofia Reza, Felipa Rosas y Erika Ríos por su valioso e incondicional apoyo en diversos trámites administrativos.

Agradezco al CONACyT por la beca otorgada durante la realización de estos estudios de doctorado, y muy especialmente al CINVESTAV por ofrecerme un ambiénte académico de calidad y excelencia.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first idea to perform computations using a quantum computer was proposed by Richard Feynman in 1982. He wondered whether it is possible to simulate a quantum system by means of a universal quantum simulator [38]. The Feynman's proposal was the initial motivation for many future experimental and theoretical results in the field of *Quantum Computation* (QC).

The first important theoretical result was given in [76], it is proposed a polynomial time quantum algorithm to the problem of factoring an integer number into its prime factors, that is believed to be a hard problem.

Since then, many quantum algorithms were proposed, for instance in [43] a sub-linear time quantum algorithm is given to solve the problem of finding an element in a non-structured database.

Quantum algorithms (QA) are based on the application of unitary operators that act in a finite dimensional Hilbert space. Thus, a QA consists of consecutive applications of unitary operators over an initial quantum state, and the output of the algorithm is obtained by performing a *quantum measurement* over the final state. This approach is known as the *quantum circuit model* (QCM) (see [61]).

On other hand, *Adiabatic Quantum Computation* (AQC) was introduced in [37] and it has been applied to solve optimization problems. It is based in the construction of a time-dependent Hamiltonian which codify the optimal solution of the given optimization problem into its ground state (see chapter 3). AQC makes use of the *Adiabatic Theorem* (see [58, 42]) to approximate solutions of the Schrödinger equation in which a slow evolution occurs.

Although, AQC approach is defined by the solutions of the continuous Schrödinger equation, it has been proved that AQC is equivalent to QCM (see [2]), and therefore AQC is a universal model of computation.

A very active area in AQC deals with the problem to determine a time lower-bound that an AQC algorithm requires in order to obtain an optimal solution. In [37] an AQC algorithm was proposed for the 3-SAT problem (see chapter 4), and it is claimed by means of simulations that the time required for the AQC algorithm scale polynomially with the input size.

The Hamiltonian operators used in AQC should be local for convenience. *Local*

Hamiltonian operators are expressed as a polynomial sum i.e. the addition of a polynomial number of Hamiltonians each acting over a reduced number of states. With the use of local Hamiltonians, it is possible to perform computations in a local way, affecting only a neighborhood of states in the quantum system. A related important problem is the *Local Hamiltonian Problem* (LHP) that consists in deciding if a given Hamiltonian acting in a Hilbert space of dimension n has an eigenvalue below a, or if all its eigenvalues are at less b, where a and b are real numbers such that $b - a \geq n^{-O(1)}$.

The LHP is known to be QMA-complete (see [52, 51, 50, 22]) where QMA is the class of problems that can be solved in polynomial time by QA's. From the point of view of complexity theory, the LHP can be seen as the quantum analog of the SAT problem for the class of problems NP and restricted versions of the LHP coincide with NP-complete problems [83].

In [1] a general technique was proposed to decompose a Hamiltonian into a polynomial sum of local Hamiltonians, it is based on the assumption that the given Hamiltonian is d-sparse and row-computable. The Hamiltonian decomposition it is important to the *Hamiltonian simulation* which consists in computing the matrix exponentiation of a given Hermitian matrix that results into a unitary matrix (see [23, 12, 65]).

Formally, the Adiabatic Theorem guaranties that the final ground state of a given Hamiltonian can be approximated with arbitrary uncertainty, and assuming that QC cannot solve NP-complete problems it follows that AQC is a means to obtain an approximation to the ground state and the ground state energy. The ratio of this approximation and its dependence on the hardness of the problem are not well understood yet. On the other hand, classical algorithms have been proposed, for instance, in [9] it was shown a classical approximation algorithm for evaluating the ground-state energy of the classical *Ising Hamiltonian* with linear terms on an arbitrary planar graph. Also, a classical approximation algorithm is proposed to the LHP (also see [62]).

The current construction of local Hamiltonians does not use the structure of the given problem. For instance, in [35] an adiabatic quantum algorithm is given for the MAX-SAT problem. It is based on a natural equivalence between clauses and Hamiltonians defined for every literal in the given instance. A similar construction is given in [66] for the *protein folding problem*, it is based on the *Ising model* to describe the local interactions in a lattice.

The design and construction of Hamiltonians have important consequences in the running time and convergence for AQC algorithms (see [36, 79, 3, 25]). In this thesis we deal with the problem of local Hamiltonian construction for combinatorial optimization problems. Also, we investigate the classical simulation of the AQC for the MAX-SAT problem. An important challenge in AQC is to propose new techniques to codify a given problem into the Hamiltonian approach, for instance, the Dynamic Programming approach is a well known technique to solve NP-hard problems and has been the basis for many polynomial time algorithms applied to graph problems (see [19, 13, 18]).

The main contributions of this thesis are the following:

- The first contribution is a complete analysis of the AQC applied to the 3-SAT problem, we analyze the syntactical construction of the initial and final Hamiltonians involved in the AQC algorithm, such as analysis is useful in order to perform numerical simulations of the AQC algorithm i.e. to avoid a direct construction of the Hamiltonians by means of tensor products. Also, we provide a precise description of the computational complexity of the AQC algorithm.

- The second contribution is a general model in terms of *pseudo-Boolean functions* to solve optimization problems. The main idea is to model any combinatorial optimization as a *quadratic pseudo-Boolean function* [21], then a Hamiltonian operator is constructed such that its ground state correspond to the point that minimizes the quadratic pseudo-Boolean function. We also, show that any problem expressed in monadic second-order logic has an associated pseudo-Boolean function with a bounded number of variables.

- The last contribution of this thesis is a Dynamic Programming approach to solve NP-hard problems. It is based on the Tree Decompositions of graphs introduced in [69, 70, 71, 68] and a dynamic programming technique in which a graph problem can be decomposed into smallest subproblems and then composed in order to construct a global solution of the problem (see [13, 18]). Based on this decomposition, we propose the Hamiltonian construction for AQC on each subproblem of the tree decomposition and composed in a global Hamiltonian whose ground state is the point at with the pseudo-Boolean function has its minimum.

The organization of this thesis is as follows: In chapter 2 a succinct introduction to computational complexity is given, the basic definitions of complexity classes and optimization problems are introduced. In chapter 3 we give an introduction to adiabatic quantum computing. It is intended to be self-contained. In chapter 4 we explore the classical simulation of the AQC for the MAX-SAT problem, we propose a symbolic analysis of the construction of Hamiltonias for AQC. In chapter 5 we give a general Hamiltonian construction for the pseudo-Boolean optimization problem. In chapter 6 we show an alternative construction of local Hamiltonians based on a study of graph decompositions and a dynamic programming approach. Finally, in chapter 7 we have the conclusions and further research.

Chapter 2

Approximability of NP-hard problems

The purpose of this chapter is to give a succinct introduction to computational complexity and its principal problems, the scope of this review ranges from complexity classes to approximating optimization problems. In the background, the knowledge of *Turing machines* (TM) is assumed (see [8, 40, 6]).

We emphasize the relationship between decision and optimization problems. In order to do this, we recall the class P in terms of *Deterministic Turing Machines* (DTM) and the class NP in terms of DTM's that test membership in languages.

The class NP has a probabilistic characterization in terms of *Probabilistic Turing Machines* (PTM), captured in the PCP theorem. The PCP theorem has important applications in approximating solutions to NP-hard problems using a standard methodology.

We also introduce the randomized complexity classes and their connections with optimization problems. Randomized computation is related to quantum computation by its probabilistic nature, that is, the quantum complexity class BQP contains the classical complexity class BPP. Here we make a survey of these notions.

2.1 Basic definitions

Let L be a language. There exists a DTM M that recognizes L whenever L is decidable. Given a DTM M, let L_M be the language recognized by M. M is said to be *polynomial-time*, if for each input string x, with $|x| = n$, M performs at most $O(n^k)$ computing steps for a fixed non-negative exponent k, where $|x|$ denote the length of the string x.

Definition 1. *The class* P *consists of all languages recognized by polynomial time DTM's.*

Let L be a language, a *verification procedure* for L is a DTM V that satisfies the following conditions:

1. *Completeness*: For each $x \in L$ there exists a string y such that $V(x, y) = 1$. (V accepts y as a valid proof for the membership of x in L)

2. *Soundness*: For each $x \notin L$ and every string y it holds that $V(x, y) = 0$. (V rejects y as proof for the membership of x in L)

A language L has an *efficiently verifiable proof system* if there exists a polynomial p and a polynomial time verification procedure V such that, for each $x \in L : (\exists y : |y| \leq p(|x|) \wedge V(x, y) = 1)$ and for every $x \notin L$ and every y, the equation $V(x, y) = 0$ holds.

Definition 2. *The class* NP *consists of all languages that have efficient verifiable proof systems.*

Hence, P $\subseteq$ NP.

Although the definition of the classes P and NP have been expressed in terms of decision problems, there exist equivalent definitions for search problems.

A *search problem* is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. For an instance x of R, let $R(x) := \{y : (x, y) \in R\}$ be the set of solutions of x. A function $f : \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ solves the search problem R if for every x, whenever $R(x) \neq \emptyset$ we have $f(x) \in R(x)$, otherwise $f(x) = \perp$.

Let L_1, L_2 be two languages, L_1 is *reducible* to L_2 if there exists a function $\Phi : L_1 \to L_2$ such that, $x \in L_1$ if and only if $\Phi(x) \in L_2$. The language L_1 is said to be *polynomially reducible* to L_2 if the reduction map Φ can be computed in polynomial time. In this case, it is written $L_1 \leq_p L_2$.

A language $L \in$ NP is NP-*complete* if for each $L' \in$ NP, $L' \leq_p L$. The NP-complete problems are thus the most difficult problems in the class NP.

2.2 Probabilistic proof systems

A verification procedure given by a DTM M can be extended by changing M with a PTM that uses a source of random bits for its computation. From now on, a verification procedure will be called a *verifier* for short.

Definition 3. *A verifier V is a PTM having an input tape, a work tape, a source of random bits and a read-only tape called proof string π. V has random access to π and the operation of reading a bit in π is called a query.*

The source of random bits of a verifier can be viewed as an input random string ρ. The figure 2.1 shows a verifier and its components: It can be seen that a verifier is equivalent to a verification procedure when the verifier does not use the random string for its computation.

Let L be a language and $q, r : \mathbb{N} \to \mathbb{N}$ be two functions. L has a $(r(n), q(n))$-*restricted verifier* if there is a verifier V such that satisfies the following conditions:

1. *Efficiency*: For each input x with $|x| = n$ and given a proof string π of length at most $q(n)2^{r(n)}$, V uses at most $r(n)$ random bits and queries at most $q(n)$ positions in π. Then V outputs 1 for "accept" or outputs 0 for "reject".

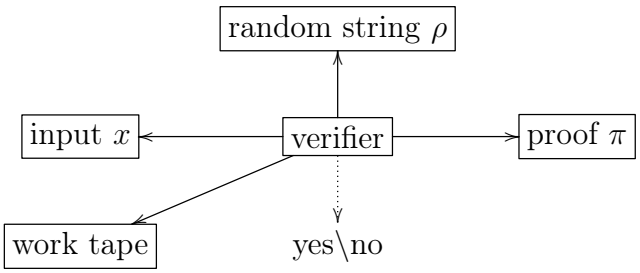

Figure 2.1: Probabilistic Turing machine (verifier).

2. *Completeness*: For each $x \in L$ there exists a proof π_x such that for every random string ρ, $\Pr[V(x, \pi_x, \rho) = 1] = 1$.

3. *Soundness*: For each $x \notin L$ and every proof π and random string ρ, $\Pr[V(x, \pi, \rho) = 1] \leq \frac{1}{2}$.

Definition 4. *The class* $\mathrm{PCP}[r(n), q(n)]$ *consists of all languages that have* $(r(n), q(n))$*-restricted verifiers.*

Note that $\mathrm{NP} = \mathrm{PCP}[0, \mathrm{poly}(n)]$ where $\mathrm{poly}(n) = \bigcup_{k \in \mathbb{N}} n^k$.

Theorem 1 (Arora & Safra, [7])**.** $\mathrm{NP} = \mathrm{PCP}[\log n, 1]$.

Theorem 1 asserts that in order to check membership, just a constant number of accesses to the proof and a logarithmic number of random bits are required. Also, theorem 1 has important applications to prove the hardness in approximating NP-hard problems [45, 82, 78], as we review in the next section.

2.3 Optimization problems

Definition 5. *An optimization problem* Π *is a tuple* $(I_\Pi, sol_\Pi, m_\Pi, goal_\Pi)$ *where* I_Π *is the set of instances of* Π*,* $sol_\Pi : I_\Pi \to \Omega(x)$ *is a function that associates to any instance* $x \in I_\Pi$ *the set of feasible solutions of* x*, and* $m_\Pi : I_\Pi \times sol_\Pi \to \mathbb{Z}^+$ *is the measure function and* $goal_\Pi \in \{\min, \max\}$*.*

The optimal solution to an instance $x \in I_\Pi$ is denoted as $y^*(x) \in \mathrm{sol}_\Pi(x)$ according to goal_Π and its measure as $m^*_\Pi(x)$.

Let $\Pi = (I_\Pi, \mathrm{sol}_\Pi, m_\Pi, \mathrm{goal}_\Pi)$ be an optimization problem, Π is in the class NPO if the set of instances I_Π can be recognized in polynomial time, namely for each $x \in I_\Pi$ there exists a polynomial p and for any $y \in \mathrm{sol}_\Pi(x)$ with $|y| \leq p(|x|)$, the membership of y in $\mathrm{sol}_\Pi(x)$ can be decided in polynomial time and the measure function can be computed in polynomial time.

The class NPO is the optimization version of NP, in the sense that every optimization problem in NPO has its corresponding decision problem in NP.

A problem Π is NP-hard if there exists an NP-complete problem Π' such that $\Pi' \leq_{T,p} \Pi$ where $\leq_{T,p}$ is a polynomial Turing reduction (See [8]).

The NP-hard problems are the most difficult problems in NPO.

2.3.1 Approximation algorithms

Let Π be an optimization problem, for any $x \in I_\Pi$ and for any value $y \in \mathrm{sol}_\Pi(x)$, the *performance ratio* of y with respect to x is defined as:

$$R(x,y) = \max\left\{\frac{m_\Pi(x,y)}{m^*_\Pi(x)}, \frac{m^*_\Pi(x)}{m_\Pi(x,y)}\right\},$$

the performance ratio is always a number greater than or equal to 1 and is closer to 1 as y is closer to the optimum solution.

An algorithm T is an *ε-approximation algorithm* for Π if, given any $x \in I_\Pi$, $R(x,T(x)) \leq \varepsilon$, and it is said that Π is ε-approximated.

APX is the class of all NPO problems Π such that, for some $\varepsilon > 1$, there exists a polynomial time ε-approximation algorithm for Π.

Let Π be an NPO problem. An algorithm T is said to be an *approximation scheme* for Π if, for any $x \in I_\Pi$ and for any rational $\varepsilon > 1$, $T(x,\varepsilon)$ returns a feasible solution of x whose performance ratio is at most ε.

Definition 6. *An* NPO *problem* Π *belongs to the class* PTAS *if it admits a polynomial-time approximation scheme.*

Note that the time complexity of an approximation scheme may be of the type $2^{1/(\varepsilon-1)}p(|x|)$ or $|x|^{1/(\varepsilon-1)}$ where p is a polynomial.

An NPO problem Π belongs to the class FPTAS if it admits a fully polynomial-time approximation scheme, that is, an approximation scheme whose time complexity is bounded by $q(|x|, 1/(\varepsilon-1))$ where q is a polynomial.

Clearly, FPTAS $\subseteq$ PTAS $\subseteq$ Apx $\subseteq$ NPO.

2.4 Randomized classes

In previous sections a PTM was introduced and considered as a verifier, here a PTM is used to compute functions in a general setting.

Let L be a language, L has a polynomial-time PTM M if there exists a polynomial p such that, for each $x \in L$, $M(x)$ can be computed within at most $p(|x|)$ steps. For any $x \in L$, $M(x)$ is a random variable over the output distribution of M with input x. Note that, if $x \in L$ then M may fail to give the answer $M(x) = 1$ with input x.

The types of failures of a PTM M for a language L can be characterized as follows:

1. *Two-sided error*: M can fail in both directions, i.e., if $x \in L$, M may rule that $M(x) = 0$, and conversely.

Type of error	Class	$x \in L$	$x \notin L$	$\perp$
Two-sided	BPP	$\Pr[M(x)=1] \geq \frac{2}{3}$	$\Pr[M(x)=0] \geq \frac{2}{3}$	
One-sided	RP	$\Pr[M(x)=1] \geq \frac{1}{2}$	$\Pr[M(x)=0] = 1$	
Zero-sided	ZPP	$\Pr[M(x)=1] = 1$, $\Pr[M(x)=1] \geq \frac{1}{2}$	$\Pr[M(x)=0] = 1$, $\Pr[M(x)=0] \geq \frac{1}{2}$	$\Pr[M(x)=\perp] = 1$

Table 2.1: Randomized class of languages.

2. *One-sided error*: M can fail in one direction, i.e., if $x \notin L$, M may rule that $M(x) = 1$, but if $x \in L$, $M(x) = 1$.

3. *Zero-sided error*: M does not fail to recognize an element in L and is able to indicate its failure to find an answer.

The table 2.1 shows the classes of languages L that can be recognized by polynomial-time PTM's M with respect to the type of failure defined before:

The symbol "$\perp$" is the output of a PTM when fails to give an answer.

Formally the class ZPP is defined as follows: A language L is in ZPP if there exists a PTM M such that $\Pr[M(x) \in \{\chi_L(x), \perp\}] = 1$ and $\Pr[M(x) \in \chi_L(x)] \geq \frac{1}{2}$ where $\chi_L(x) = 1$ if $x \in L$ and $\chi(x) = 0$ if $x \notin L$.

It is easy to prove that RP $\subseteq$ NP and RP $\subseteq$ BPP.

A PTM can approximate solutions of a NP-hard problem considering their corresponding decision problem: Let $\Pi = (I_\Pi, \text{sol}_\Pi, m_\Pi, \text{goal}_\Pi)$ be an optimization problem. Given $x \in I_\Pi$ and an integer $k \in \mathbb{Z}^+$, the decision problem Π_D with respect to Π is the following: decide whether $m^*_\Pi(x) \geq k$ if $\text{goal}_\Pi = MAX$ or whether $m^*_\Pi(x) \leq k$ if $\text{goal}_\Pi = \min$. Finally, the language with respect to Π_D is $L_\Pi = \{(x,k) | x \in I_\Pi \wedge m^*_\Pi(x) \geq k\}$ if $\text{goal}_\Pi = \max$.

2.4.1 Quantum complexity

Quantum computation is realized in finite dimensional Hilbert spaces, the operations are realized as unitary operators over unit vectors represented as linear combinations of vectors in an orthonormal basis. Quantum measurements are the operations of reading the results. The notion of Quantum algorithms were first introduced in [11] using *Quantum Turing Machines* (QTM), which extend the classical TM.

Formally, a QTM is a triplet (Σ, Q, δ) where Σ is a finite alphabet, Q is a finite set of states with an distinguished initial state q_0, a final state q_f, and δ a quantum transition function

$$\delta : Q \times \Sigma \to \tilde{\mathbb{C}}^{Q \times \Sigma \times \{L,R\}}$$

where L, R is a left or right displacement over the tape machine and $\tilde{\mathbb{C}}$ is a set of computable complex numbers within some precision.

The QTM has a two-way infinite tape of cells indexed by $\mathbb{Z}$ and a single red/write tape head that moves along the tape. Given a pair $(q, s) \in Q \times \Sigma$, δ associates a

complex number $\alpha \in \tilde{\mathbb{C}}$ to (q, s), such that the absolute value of α is the probability that the QTM will be in the new configuration (q', s') performing a left or right displacement over the tape machine.

A QTM halts if reaches the final configuration with state q_f, and it is said to be polynomial time if it performs a polynomial number of states transitions. It is possible to define an inner-product $\mathbb{S}$ space over $\tilde{\mathbb{C}}$ as the space of configurations of a QTM with the Euclidean norm, in this way, a linear combination of states in $\mathbb{S}$ is a superposition of states of a QTM.

A QTM M recognizes exactly the language L, if for each $x \in L$, M accepts x with probability 1 and for each $x \notin L$, M rejects x with probability 1.

The class EQP consists of all languages recognized exactly by polynomial time QTM's.

A QTM M recognizes the language L with probability p if for each $x \in L$, M accepts x with probability p and for each $x \notin L$, M rejects x with probability $1 - p$.

Definition 7. *The class* BQP *consists of all languages that are recognized by polynomial time QTM's with probability* $\frac{2}{3}$.

The zero-sided error version of BQP is the class ZQP defined as follows: A language L is in ZQP if for every $x \in L$, x is accepted by some polynomial time QTM with probability $\frac{2}{3}$ and rejected with probability 0, and for every $x \notin L$, x is rejected by some polinomial time QTM with probability $\frac{2}{3}$ and accepted with probability 0.

Hence, EQP $\subseteq$ ZQP $\subseteq$ BQP.

The known relations with classical complexity classes are: P $\subseteq$ BQP, BPP $\subseteq$ BQP and BQP $\subseteq$ PSPACE [61].

Chapter 3

Adiabatic quantum computing

This chapter is a self-contained introduction to *adiabatic quantum computing* (AQC) as a general approach to solve optimization problems.

The first part is dedicated to recall the basic notions of *Hilbert spaces* and their metrics. The linear operators are introduced and their properties in the evolution of Quantum Systems using the Schrödinger picture.

Also, a brief terminology with *Quantum Mechanics* is given, in order to define *states*, *observables*, *measurements* and *dynamics* of a quantum system. An important part of this chapter is the exposition of the *Adiabatic Theorem*, which is the fundamental tool in AQC. We sketch the general algorithm for AQC to solve optimization problems (see [37]).

Finally, we analyze the conditions in which the Adiabatic Theorem is satisfied and the influence of the geometric Berry phases in the evolution of the adiabatic paths (see [42]).

3.1 Basic definitions

A *metric space* is a pair (M, d) where M is a non-empty set and $d : M \times M \to \mathbb{R}^+$ is a metric on M satisfying $\forall x, y, z \in M$:

1. $d(x, y) \geq 0$, and $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, y) \leq d(x, z) + d(z, y)$.

A metric space M is *complete* if every Cauchy sequence converges in M.

If V is any vector space then, the elements of V will be written using bold lowercase letters as $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Any complex vector space V with induced norm by the inner product is a metric space with metric defined as $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in V$.

Let $\langle \cdot | \cdot \rangle : V \times V \to \mathbb{C}$ be an inner product thus, for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V, a, b \in \mathbb{C}$:

1. $\langle \mathbf{x} | \mathbf{y} \rangle \geq 0$ and the following holds $\langle \mathbf{x} | \mathbf{y} \rangle = 0$ if and only if $\mathbf{x} = \mathbf{y}$,

2. $\langle \mathbf{x}|a\mathbf{y} + b\mathbf{z}\rangle = a\langle \mathbf{x}|\mathbf{y}\rangle + b\langle \mathbf{x}|\mathbf{z}\rangle$,

3. $\langle \mathbf{x}|\mathbf{y}\rangle = \langle \mathbf{y}|\mathbf{x}\rangle^*$.

The *induced norm* of a complex vector space V is defined as $\|\mathbf{x}\| = \langle \mathbf{x}|\mathbf{x}\rangle^{\frac{1}{2}}$ with $\mathbf{x} \in V$, and for any $\mathbf{x}, \mathbf{y} \in V, a \in \mathbb{C}$,

1. $\|\mathbf{x}\| \geq 0$, and $\|\mathbf{x}\| = 0$ if $\mathbf{x} = \mathbf{0}$,

2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$,

3. $\|a\mathbf{x}\| = |a|\|\mathbf{x}\|$.

Definition 8. *A Hilbert space is a complex vector space $\mathbb{H}$ with inner product which is complete with respect to the metric induced by the inner product.*

Let $\mathbb{H}_1 = \mathbb{C}^2$ be the complex Hilbert space of dimension 2. Let, for each $n > 1$, $\mathbb{H}_n = \mathbb{H}_{n-1} \otimes \mathbb{H}_1$ be the n-fold tensor product of $\mathbb{H}_1$. $\mathbb{H}_n$ is a Hilbert space of dimension $N = 2^n$, and for any $\mathbf{x} \in \mathbb{H}_n : \mathbf{x} = (x_0, \ldots, x_{N-1})$ is a vector with N complex entries.

For any two integers $i, j \in \mathbb{N}$, $i \leq j$, let $[\![i, j]\!]$ denote the collection of integers ranging from i to j, $[\![i, j]\!] = \{i, i+1, \ldots, j-1, j\}$.

Let $\langle \cdot|\cdot\rangle : \mathbb{H}_n \times \mathbb{H}_n \to \mathbb{C}$ be the inner product in $\mathbb{H}_n$ defined as:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{H}_n : \langle \mathbf{y}|\mathbf{x}\rangle = \sum_{i=0}^{N-1} y_i^* x_i = (\mathbf{y})^H \mathbf{x}$$

where $(\mathbf{y})^H = (\mathbf{y}^T)^*$ is the *Adjoint Hermitian* of $\mathbf{y}$.

A vector $\mathbf{x} \in \mathbb{H}_n$ is a *unit vector* if $\|\mathbf{x}\| = 1$. A basis for $\mathbb{H}_n$ is a linearly independent vector family $(\mathbf{x}_i)_{i=0}^{N-1}$ satisfying $\forall \mathbf{z} \in \mathbb{H}_n : \mathbf{z} = \sum_{i=0}^{N-1} \alpha_i \mathbf{x}_i$ for some complex numbers $(\alpha_i)_{i=0}^{N-1}$. A basis $(\mathbf{x}_i)_{i=0}^{N-1}$ is orthonormal if, for all $i, j \in [\![0, N-1]\!]$ with $i \neq j$, $\langle \mathbf{x}_i|\mathbf{x}_j\rangle = 0$.

3.1.1 Linear operators

Let $\mathbb{H}_n$ be a Hilbert space and let $T : \mathbb{H}_n \to \mathbb{H}_n$ be an operator, T is a linear operator if $T(\sum_{i=0}^{k-1} a_i \mathbf{x}_i) = \sum_{i=0}^{k-1} a_i T(\mathbf{x}_i)$ where $\mathbf{x}_i \in \mathbb{H}_n$ and $a_i \in \mathbb{C}$ for all $i \in [\![0, k-1]\!]$. Let $I : \mathbb{H}_n \to \mathbb{H}_n$ be the *identity operator* and $0 : \mathbb{H}_n \to \mathbb{H}_n$ be the *zero operator*: for any $\mathbf{x} \in \mathbb{H}_n$, $I\mathbf{x} = \mathbf{x}$ and $0\mathbf{x} = \mathbf{0}$.

The set of all linear operators from $\mathbb{H}_n$ to $\mathbb{H}_n$ is denoted as $\mathcal{L}(\mathbb{H}_n)$. A linear operator $T \in \mathcal{L}(\mathbb{H}_n)$ is *self-adjoint* or *Hermitian* if $\langle T\mathbf{x}|\mathbf{y}\rangle = \langle \mathbf{x}|T\mathbf{y}\rangle$ and is *unitary* if $\langle T\mathbf{x}|T\mathbf{y}\rangle = \langle \mathbf{x}|\mathbf{y}\rangle$ for every choice of $\mathbf{x}, \mathbf{y} \in \mathbb{H}_n$.

Let $\mathrm{GL}(\mathbb{H}_n) = \{T \in \mathcal{L}(\mathbb{H}_n) | \det T \neq 0\}$ be the set of all invertible linear operators in $\mathcal{L}(\mathbb{H}_n)$ and let $\mathrm{SU}(\mathbb{H}_n) = \{T \in \mathrm{GL}(\mathbb{H}_n) | \ |\det T| = 1\}$ be the set of all unitary operators in $\mathrm{GL}(\mathbb{H}_n)$.

Given an orthonormal basis $(\mathbf{x}_i)_{i=0}^{N-1}$ for $\mathbb{H}_n$ and $T \in \mathcal{L}(\mathbb{H}_n)$ then, the matrix representation of T is a matrix $T_{ij} \in \mathbb{C}^{N\times N}$ with entries t_{ij}:

$$\forall i \in [\![0, N-1]\!] : T\mathbf{x}_i = \sum_{j=0}^{N-1} t_{ij}\mathbf{x}_j,$$

we will use the matrix representation of an operator when it is clear from the context.

A linear operator $T \in \mathcal{L}(\mathbb{H}_n)$ is *Hermitian* if $T^H = T$ and is *unitary* if $T^H T = I$.

Definition 9. *Let $T, S \in \mathcal{L}(\mathbb{H}_n)$ be two Hermitian matrices, T and S commute if and only if $[S, T] \equiv ST - TS = \mathbf{0}$.*

For any T, S Hermitian operators and $a \in \mathbb{C}$, the following properties are satisfied:

1. $(aT)^H = a^* T^H$,
2. $(T + S)^H = T^H + S^H$,
3. $(TS)^H = S^H T^H$,

TS is Hermitian if and only if T and S commute.

A Hermitian operator $T \in \mathcal{L}(\mathbb{H}_n)$ is *positive definite* if $\mathbf{x}^H T\mathbf{x} > 0$ for any vector $\mathbf{x} \in \mathbb{H}_n$.

Let $T \in \mathcal{L}(\mathbb{H}_n)$ be a linear operator, a nonzero vector $\mathbf{x} \in \mathbb{H}_n$ is *invariant* under T if and only if there exists a constant $\lambda \in \mathbb{C}$ such that $T\mathbf{x} = \lambda\mathbf{x}$. The number λ is said to be an *eigenvalue* of T and the vector $\mathbf{x}$ is said to be an *eigenvector* of T.

Let $T \in \mathcal{L}(\mathbb{H}_n)$, the *null subspace* of T is defined as $\mathcal{N}(T) := \{\mathbf{x} \in \mathbb{H}_n | T\mathbf{x} = \mathbf{0}\}$. The *spectrum* of T is defined as $\Lambda(T) := \{\lambda \in \mathbb{C} | \mathcal{N}(T - \lambda I) \neq \{\mathbf{0}\}\}$, i.e., $\Lambda(T) = \{\lambda_1, \ldots, \lambda_m\}$ is the set of all distinct eigenvalues of T. For any $j \in [\![1, m]\!]$, let $\gamma_j \equiv \dim \mathcal{N}(T - \lambda_j I)$ be the dimension of the subspace spanned by the eigenvectors corresponding to the eigenvalue λ_j. An eigenvalue λ_j is called *non-degenerate* if $\gamma_j = 1$ and it is called *degenerate* if $\gamma_j > 1$.

An important measure on linear operators is the *spectral norm*, which is defined as follows: For any linear operator $T : \mathbb{H}_n \to \mathbb{H}_n$,

$$\|T\| = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|T\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|T\mathbf{x}\|.$$

The spectral norm satisfies the following properties: For any $T, S \in \mathcal{L}(\mathbb{H}_n)$

1. $\|TS\| \leq \|T\|\|S\|$,
2. $\|T^H\| = \|T\|$,
3. $\|T \otimes S\| = \|T\|\|S\|$,
4. $\|T\| = 1$ if T is unitary.

If T is a Hermitian operator then its spectral norm $\|T\| = \max\{|\lambda| \ | \lambda \in \Lambda(T)\}$ and $\|T\|^2$ is the largest eigenvalue of the operator $T^H T$.

3.2 Quantum states and evolution

In the following, a brief introduction to the concepts and terminology of *Quantum Mechanics* (QM) used in this thesis are given (see [58] for a complete treatment in QM).

The *Quantum Theory* is a mathematical model of the physical world. In order to specify this model it is necessary to define the following concepts: *states*, *observables*, *measurements* and *dynamics*.

1. *States*: A state is a complete description of a physical system. In QM a state is an unitary vector in a Hilbert space. Thus, the class of states of a quantum system coincides with the unit sphere on a Hilbert space. For instance, let $(\mathbf{x}_i)_{i=0}^{N-1}$ be an orthonormal basis for $\mathbb{H}_n$, for any $\mathbf{z} \in \mathbb{H}_n : \mathbf{z} = \sum_{i=0}^{N-1} \alpha_i \mathbf{x}_i$ where $\alpha_i \in \mathbb{C}$ with $i \in [\![0, N-1]\!]$, if $\mathbf{z}$ is unitary then $\sum_{i=0}^{N-1} |\alpha_i| = 1$. The squares of the absolute value of the scalars $(\alpha_i)_{i=0}^{N-1}$ correspond to a probability distribution and the value $|\alpha_i|^2$ is the probability of being in the state $\mathbf{x}_i$ for $i \in [\![0, N-1]\!]$.

2. *Observables and measurements*: An observable is a property of a physical system that in principle can be measured. In QM an observable is a Hermitian operator. Let us see how an observable M can be represented as a sum of projector matrices, also called the *spectral representation*. Let $M \in \mathcal{L}(\mathbb{H}_n)$ be an observable and let $(\mathbf{x}_i)_{i=0}^{N-1}$ be an orthonormal basis for $\mathbb{H}_n$, M can be represented as:

$$M = \sum_{i=0}^{N-1} \lambda_i P_i,$$

where $P_i = \mathbf{x}_i \mathbf{x}_i^H$ is the orthogonal projection onto the subspace spanned by the eigenvector $\mathbf{x}_i$ that corresponds to the eigenvalue $\lambda_i \in \Lambda(M)$. For all $i, j \in [\![0, N-1]\!] : P_i P_j = \delta_{ij} P_i, P_i^H = P_i$ and $\sum_{i=0}^{N-1} P_i^H P_i = \mathrm{Id}_n$.

An eigenstate of an observable is called an *energy state* and its corresponding eigenvalue is called the *energy*. The lowest energy of an observable is known as the *ground energy* and its corresponding energy state is known as the *ground state*. For any two observables $M_1, M_2 \in \mathcal{L}(\mathbb{H}_n)$, $M_1 + M_2$ is also an observable, but $M_1 M_2$ is an observable if and only if M_1 and M_2 commute.

The probability of finding a system in the energy λ_i of an observable M is given by:

$$\Pr(\lambda_i) = \|P_i \mathbf{x}\|^2 = \mathbf{x} P_i \mathbf{x}^H,$$

where $\mathbf{x}$ is the quantum state prior to the measurement and $\sum_{i=0}^{N-1} \Pr(\lambda_i) = 1$. If the outcome of a measurement is λ_i for an observable M, then the quantum state right after the measurement becomes:

$$\mathbf{y} = \frac{P_i \mathbf{x}}{(\mathbf{x} P_i \mathbf{x}^H)^{\frac{1}{2}}}.$$

3. *Dynamics*: The time evolution of a quantum state is described by a Hermitian operator also called a *Hamiltonian* of the system. In the *Schrödinger picture* of dynamics, the time evolution of a quantum system is governed by the Schrödinger equation. Let $H : \mathbb{R} \to \mathrm{GL}(\mathbb{H}_n)$ be a time dependent Hamiltonian and let $\mathbf{x} : \mathbb{R} \to \mathbb{H}_n$ be a differentiable transformation in the interval $I \subset \mathbb{R}$, then the Schrödinger equation is:

$$\forall t \in I : \frac{d}{dt}\mathbf{x}(t) = -iH(t)\mathbf{x}(t),$$

and can be rewritten as a first-order equation in the infinitesimal quantity dt as:

$$\mathbf{x}(t + dt) = U(dt)\mathbf{x}(t),$$

where $U(dt) := \mathrm{Id}_n - iH(t)dt$, hence $U^H U = \mathrm{Id}_n$. U is unitary if H is a time independent Hamiltonian.

3.3 The Adiabatic Theorem

The adiabatic approximation is a standard method of quantum mechanics used to derive approximate solutions of the Schrödinger equation in the case of a slowly varying Hamiltonian. The adiabatic approximation works as follows:

Put a quantum system in its ground state. If the Hamiltonian varies slowly enough, then the quantum system will stay in a state close to the instantaneous ground state of the Hamiltonian as the time goes on (see [58]).

3.3.1 Adiabatic evolution

Let $\mathbb{H}_n$ be a Hilbert space and let $H : \mathbb{R} \to \mathrm{GL}(\mathbb{H}_n)$ be a time dependent Hamiltonian. The differentiable transformation $\mathbf{x} : \mathbb{R} \to \mathbb{H}_n$ is a solution of the Schrödinger equation in the interval $I \subset \mathbb{R}$ if

$$\forall i \in I : i\frac{d}{dt}\mathbf{x}(t) = H(t)\mathbf{x}(t). \tag{3.1}$$

Let $J \subset \mathbb{R}$ be an interval and let $\tau : s \mapsto t = as + b$ be an affine transformation $J \to I$. Let $G : J \to \mathrm{GL}(\mathbb{H}_n)$ be such that $G(s) = aH(\tau(s))$.

Thus, if $\mathbf{x} : \mathbb{R} \to \mathbb{H}_n$ is a solution of (3.1) then

$$\forall s \in J : H(\tau(s))\mathbf{x}(\tau(s)) = i\frac{d}{dt}\mathbf{x}(\tau(s)) = i\frac{1}{a}\frac{d}{ds}\mathbf{x}(\tau(s))$$

thus,

$$\forall s \in J : i\frac{d}{dt}\mathbf{x}(\tau(s)) = G(s)\mathbf{x}(\tau(s))$$

and $\mathbf{x}\circ\tau$ is a solution of the Schrödinger equation in J for the Hamiltonian $G = aH\circ\tau$. G is a continuous path in the space of Hermitian operators on $\mathbb{H}_n$.

For instance, if $J_{t_0} = [0, t_0]$ and $I = [0, 1]$ the affine transformation is $s \mapsto as + b = \frac{s}{t_0}$ and the Hamiltonian on J_{t_0} is $H_{t_0}(s) = \frac{1}{t_0} H(\frac{s}{t_0})$.

Let $\mathbf{x}_{t_0} : J_{t_0} \to \mathbb{H}_n$ be a solution of the equation

$$\forall s \in J_{t_0} : i\frac{d}{dt}\mathbf{x}_{t_0}(s) = H_{t_0}(s)\mathbf{x}_{t_0}(s) \tag{3.2}$$

Let $\{\lambda_0, \ldots, \lambda_{N-1}\} \subset \mathbb{R}^I$ be the spectrum of the Hamiltonian H such that for all $j \in [\![0, N-1]\!]$ and for all $t \in I$, there exists $\mathbf{y}_j(t) \in \mathbb{H}_n$ (instantaneous eigenstate of the Hamiltonian $H(t)$ with corresponding energy λ_j):

$$H(t)\mathbf{y}_j(t) = \lambda_j \mathbf{y}_j(t) \text{ with } \|\mathbf{y}_j(t)\| = 1$$

and

$$\lambda_0(t) \leq \cdots \leq \lambda_{N-1}(t).$$

The instantaneous eigenvalues are considered non-degenerated.

The path defined by the eigenvectors $(\mathbf{y}_0(t))_{t\in[0,1]}$ have extreme points $\mathbf{y}_0(0), \mathbf{y}_0(1)$. Let $\mathbf{z} \mapsto \langle \mathbf{y}_0(1)|\mathbf{z}\rangle$ be a linear transformation from $\mathbb{H}_n \to \mathbb{C}$ with respect to the instantaneous eigenvector $\mathbf{y}_0(1)$. If $\lambda_1(t) - \lambda_0(t) > 0$ for all $t \in [0, 1]$ then, the *Adiabatic Theorem* asserts that:

$$\lim_{t_0 \to +\infty} | \langle \mathbf{y}_0(1)|\mathbf{x}_{t_0}(t_0)\rangle | = 1.$$

This is the case of an infinitely slow or adiabatic passage. In other words, if the system is initially in an eigenstate of $H(0)$ it will, at time $t = 1$, under certain conditions to be specified later, have passed into the eigenstate of $H(1)$, that derives it by continuity.

An upper-bound for the time needed to satisfy the Adiabatic Theorem is the following:

$$T \geq \frac{\Delta_{max}}{\epsilon \delta_{min}^2}$$

where $\delta_{min} = \min_{0\leq t\leq 1}(\lambda_1(t) - \lambda_0(t))$, $\Delta_{max} = \max \|\frac{d}{dt}H(t)\|$ and $\epsilon \in [0, 1]$ is the approximation ratio to the ground state of H.

3.3.2 Quantum computation by adiabatic evolution

The AQC was proposed in [37] as a general technique to solve optimization problems and was initially applied to the MAX-SAT problem. In [35] it was shown by means of computational experiments that AQC can approximate solutions in polynomial time complexity for small instances of the MAX-SAT problem.

In [2] shows that AQC is equivalent to the circuit model of quantum computation and viceversa.

The adiabatic evolution of a quantum system can be used to solve optimization problems going from ground states to ground states of a time dependent Hamiltonian.

Thus, given an optimization problem Π with its corresponding energy function or evaluation function and for a time dependent Hamiltonian $H(t)$ for $0 \leq t \leq 1$. The

ground state of H at time $t = 1$ will correspond to the solution of the optimization problem. If the Hamiltonian H at time $t = 0$ is initially in an easily computable ground state (possibly in an uniform superposition of all basis states), then by the Adiabatic Theorem, for an infinitely slowly passage from $t = 0$ to $t = 1$, the evolution of the quantum system goes from the ground states to the ground states of H.

The general steps of the AQC algorithm are the following:

1. Prepare the quantum system in the ground state (which is known and easy to prepare) of another Hamiltonian H_0.

2. Encode the solution of an optimization problem into the ground state of a Hamiltonian H_f.

3. Evolve the quantum system slowly enough satisfying the Adiabatic Theorem with the Hamiltonian $H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_f$ for a total time T. The final state $\mathbf{x}(t)$ at time $t = T$ will be (very close) the ground state of H_f (see equation (3.1)).

4. Perform a measurement of the state $\mathbf{x}(t)$ at time $t = T$. With high probability the optimal solution of the optimization problem is found.

An important problem in AQC is to bound the time evolution T in order to satisfy the Adiabatic Theorem. Thus, for a given NP-hard problem, it is convenient that T grows polynomially with respect to the size of the instance problem.

3.4 Adiabatic paths

Let $\mathbb{H}_n$ be a Hilbert space of dimension $N = 2^n$ and let S be the unit sphere on $\mathbb{H}_n$, i.e., the class of all unitary states in $\mathbb{H}_n$. Let $t \mapsto H(t)$ be a continuous parametrization from $\mathbb{R}^+ \to \mathrm{GL}(\mathbb{H}_n)$ (a time dependent Hamiltonian). We claim that, for slowly changes of $H(t)$ in a closed internal, if the eigenvalue curves of H do not cross, then the instantaneous ground states remain invariants.

Let $t \mapsto \mathbf{x}(t)$ be a differentiable transformation and solution of the Schrödinger equation:

$$i\hbar \frac{d}{dt}\mathbf{x}(t) = H(t)\mathbf{x}(t). \tag{3.3}$$

Let $\Lambda(t) = \{\lambda_0(t), \ldots, \lambda_{N-1}(t)\}$ be the set of eigenvalues of $H(t)$ with $t \in \mathbb{R}^+$, sorted in decreasing order with respect to the absolute values. For each $j < N$, let $\mathbf{x}_j(t)$ be an eigenvector with corresponding eigenvalue $\lambda_j(t)$. Then:

$$H(t)\,\mathbf{x}_j(t) = \lambda_j(t)\,\mathbf{x}_j(t). \tag{3.4}$$

Assuming that the curves $\lambda_j(t)$ do not cross, i.e., each curve $\mathbf{x}_j : \mathbb{R}^+ \to S$ evolve adiabatically, the ground state of $H(t)$ is $\mathbf{x}_{N-1}(t)$.

At each time $t \in \mathbb{R}^+$ the set of eigenvectors $E(t) = (\mathbf{x}_j(t))_{j=0}^{N-1}$ is orthonormal in $\mathbb{H}_n$:

$$\forall k, j \in [\![0, N-1]\!] : \left[k \neq j \implies \mathbf{x}_k(t)^H \mathbf{x}_j(t) = \delta_{jk}\right].$$

Expressing a solution $\mathbf{x}(t)$ of the equation (3.3) as a linear combination of the elements in $E(t)$ modified by a phase factor:

$$\forall t \in \mathbb{R}^+ : \quad \mathbf{x}(t) = \sum_{j=0}^{N-1} c_j(t)\, e^{i\theta_j(t)}\, \mathbf{x}_j(t), \tag{3.5}$$

where each phase θ_j is given by

$$\forall t \in \mathbb{R}^+ : \quad \theta_j(t) = -\frac{1}{\hbar} \int_0^t \lambda_j(s)\, ds. \tag{3.6}$$

Then, according to the equation (3.3) and using elementary rules of derivation:

$$i\hbar \sum_{j=0}^{N-1} e^{i\theta_j(t)} \left[c_j'(t)\, \mathbf{x}_j(t) + c_j(t)\, \mathbf{x}_j'(t) + i\theta_j(t) c_j(t)\, \mathbf{x}_j(t)\right] = \sum_{j=0}^{N-1} c_j(t)\, e^{i\theta_j(t)}\, H(t) \mathbf{x}_j(t). \tag{3.7}$$

and

$$c_k'(t) = -\sum_{j=0}^{N-1} c_j(t) e^{i(\theta_j(t) - \theta_k(t))}\, \mathbf{x}_k(t)^H \mathbf{x}_j'(t). \tag{3.8}$$

Now, deriving equation (3.4), it follows:

$$H'(t)\, \mathbf{x}_j(t) + H(t)\, \mathbf{x}_j'(t) = \lambda_j'(t)\, \mathbf{x}_j(t) + \lambda_j(t)\, \mathbf{x}_j'(t)$$

hence

$$\mathbf{x}_k(t)^H H'(t)\, \mathbf{x}_j(t) +, \mathbf{x}_k(t)^H H(t)\, \mathbf{x}_j'(t) = \lambda_j'(t)\, \delta_{kj} + \lambda_j(t)\, , \mathbf{x}_k(t)^H \mathbf{x}_j'(t).$$

Thus, since H is an adjoint operator,

$$k \neq j \implies \mathbf{x}_k(t)^H H'(t)\, \mathbf{x}_j(t) = (\lambda_j(t) - \lambda_k(t))\, , \mathbf{x}_k(t)^H \mathbf{x}_j'(t). \tag{3.9}$$

From (3.8) and (3.9), it follows:

$$c_k'(t) = -c_k(t)\, \mathbf{x}_k(t)^H \mathbf{x}_k'(t) - \sum_{j \in [\![0, N-1]\!] - \{k\}} c_j(t)\, \frac{e^{i(\theta_j(t) - \theta_k(t))}}{\lambda_j(t) - \lambda_k(t)}\, \mathbf{x}_k(t)^H H'(t)\, \mathbf{x}_j(t). \tag{3.10}$$

Now, if H change slowly enough with respect to t, then $\|H'(t)\|$ will be small, the terms in the right hand side of (3.10) are negligible, and the following approximation is found:

$$c_k'(t) = -c_k(t)\, \mathbf{x}_k(t)^H \mathbf{x}_k'(t),$$

whose solution is given by

$$\forall t \in \mathbb{R}^+ : \quad c_k(t) = c_k(0)e^{i\gamma_k(t)}. \tag{3.11}$$

where

$$\gamma_j(t) = i \int_0^t \mathbf{x}_k(s)^H \mathbf{x}_k'(s)\, ds. \tag{3.12}$$

The equation (3.5) can be written as

$$\forall t \in \mathbb{R}^+ : \quad \mathbf{x}(t) = \sum_{j=0}^{N-1} c_j(0)e^{i\gamma_j(t)}\, e^{i\theta_j(t)}\, \mathbf{x}_j(t). \tag{3.13}$$

If the system is initially in the ground state of $H(0)$, then $c_{N-1} = 1$ and $c_j = 0$ for any $j \neq N-1$, therefore, from equation (3.13),

$$\mathbf{x}(t) = e^{i\gamma_{N-1}(t)}\, e^{i\theta_{N-1}(t)}\, \mathbf{x}_{N-1}(t),$$

that is, the system remains in the same ground state up to a phase factor. The hypothesis concerning the not-crossing of the eigenvalue curves is used mainly in the relation (3.10).

3.4.1 Geometric Berry phases

In the following we consider that the continuous parametrization $t \mapsto H(t)$ from $\mathbb{R}^+ \to \mathrm{GL}(\mathbb{H}_n)$ follows a closed trajectory.

Let $P \subset \mathbb{C}^k$ be a set of parameters i.e., an open set in the topology of $\mathbb{C}^k$. Let $\mathbf{r} \mapsto H(\mathbf{r})$ be a continuous parametrization from $P \to \mathrm{GL}(\mathbb{H}_n)$. Let $t \mapsto \mathbf{r}(t)$ be a curve from $\mathbb{R}^+ \to P$ such that, for each t, k-parameters are selected. Let $t \mapsto \mathbf{x}(t)$ be a differentiable transformation and solution of the Schrödinger equation:

$$i\hbar \frac{d}{dt}\mathbf{x}(t) = H(\mathbf{r}(t))\mathbf{x}(t). \tag{3.14}$$

Let $\Lambda(\mathbf{r}(t)) = \{\lambda_0(\mathbf{r}(t)), \ldots, \lambda_{N-1}(\mathbf{r}(t))\}$ be the set of eigenvalues of $H(\mathbf{r}(t))$ with $t \in \mathbb{R}^+$, sorted in decreasing order with respect to the absolute values. For each, $j < N$, let $\mathbf{x}_j(\mathbf{r}(t))$ be an eigenvector with corresponding eigenvalue $\lambda_j(\mathbf{r}(t))$. Then:

$$H(\mathbf{r}(t))\, \mathbf{x}_j(\mathbf{r}(t)) = \lambda_j(\mathbf{r}(t))\, \mathbf{x}_j(\mathbf{r}(t)). \tag{3.15}$$

Assuming that the curves $\lambda_j(\mathbf{r}(t))$ do not cross, i.e., the curve $\mathbf{r} : \mathbb{R}^+ \to P$ evolve adiabatically, the ground state of $H(\mathbf{r}(t))$ is $\mathbf{x}_{N-1}(\mathbf{r}(t))$.

Assuming that the solution $\mathbf{x}(t)$ of (3.14) coincide with the ground state up to a phase-shift factor:

$$\forall t \in \mathbb{R}^+ : \quad \mathbf{x}(t) = e^{i\phi_{N-1}(t)}\, \mathbf{x}_{N-1}(\mathbf{r}(t)). \tag{3.16}$$

By physical considerations, the dynamic phase factor is defined as:

$$\forall t \in \mathbb{R}^+ : \quad \theta_{N-1}(t) = -\frac{1}{\hbar}\int_0^t \lambda_{N-1}(\mathbf{r}(s))\, ds. \tag{3.17}$$

The Berry phase is defined as the following difference:

$$\forall t \in \mathbb{R}^+ : \quad \gamma_{N-1}(t) = \phi_{N-1}(t) - \theta_{N-1}(t). \tag{3.18}$$

From equations (3.14), (3.15) and (3.16), it follows:

$$\forall t \in \mathbb{R}^+ : \quad 0 = \frac{d}{dt}\mathbf{x}_{N-1}(\mathbf{r}(t)) + i\,\gamma'_{N-1}(t)\mathbf{x}_{N-1}(\mathbf{r}(t)). \tag{3.19}$$

Hence, $\forall t \in \mathbb{R}^+$:

$$\begin{aligned} \gamma'_{N-1}(t) &= -i\, i\gamma'_{N-1}(t)\mathbf{x}^H_{N-1}(\mathbf{r}(t))\mathbf{x}_{N-1}(\mathbf{r}(t)) \\ &= i\,\mathbf{x}^H_{N-1}(\mathbf{r}(t))\frac{d}{dt}\mathbf{x}_{N-1}(\mathbf{r}(t)) \\ &= i\,\mathbf{x}^H_{N-1}(\mathbf{r}(t))D_{\mathbf{r}}\mathbf{x}_{N-1}(\mathbf{r}(t))\ \frac{d}{dt}\mathbf{r}(t) \end{aligned} \tag{3.20}$$

(in the first equality the fact of unitarity was used, in the second the relation (3.19), and in the last the chain rule of derivation was used). Integrating the equation (3.20),

$$\gamma_{N-1}(t) = i\int_0^t \left(\mathbf{x}^H_{N-1}(\mathbf{r}(s))D_{\mathbf{r}}\mathbf{x}_{N-1}(\mathbf{r}(s))\right)\ \mathbf{r}'(s)\ ds \tag{3.21}$$

Assuming that the curve $\mathbf{r}$ is a circuit C in P i.e., for a time T, $\mathbf{r}(T) = \mathbf{r}(0)$ and $C = \{\mathbf{r}(t) | t \in [0, T]\}$, then equation (3.21) becomes the so called *Geometric Berry phase*:

$$\gamma_{N-1}(C) = i\oint_C \left(\mathbf{x}^H_{N-1}(\mathbf{r})D_{\mathbf{r}}\mathbf{x}_{N-1}(\mathbf{r})\right)\ d\mathbf{r} \tag{3.22}$$

Since the states have constant length 1, it follows that:

$$\begin{aligned} 0 &= D_{\mathbf{r}}\left(\mathbf{x}^H_{N-1}(\mathbf{r})\mathbf{x}_{N-1}(\mathbf{r})\right) \\ &= \left(D_{\mathbf{r}}\mathbf{x}^H_{N-1}(\mathbf{r})\right)\mathbf{x}_{N-1}(\mathbf{r}) + \mathbf{x}^H_{N-1}(\mathbf{r})D_{\mathbf{r}}\left(\mathbf{x}_{N-1}(\mathbf{r})\right) \\ &= 2\,\Re\left(\mathbf{x}^H_{N-1}(\mathbf{r})D_{\mathbf{r}}\left(\mathbf{x}_{N-1}(\mathbf{r})\right)\right), \end{aligned}$$

thus, the integrand $\mathbf{x}^H_{N-1}(\mathbf{r})D_{\mathbf{r}}\mathbf{x}_{N-1}(\mathbf{r})$ in (3.22) is entirely imaginary and the geometric Berry phase $\gamma_{N-1}(C)$ is a real number. If $\gamma_{N-1}(C) = 0$, then the system is called *holonomic*. There are several types of non-holonomic systems and each one of them depends on the geometry where they belong.

Let us write (3.22) as:

$$\gamma_{N-1}(C) = \oint_C A_{N-1}(\mathbf{r})\, d\mathbf{r} \tag{3.23}$$

where

$$\mathbf{r} \mapsto A_{N-1}(\mathbf{r}) = i\mathbf{x}_{N-1}^H(\mathbf{r}) D_{\mathbf{r}} \mathbf{x}_{N-1}(\mathbf{r}) \tag{3.24}$$

is a *recalibration potential* (*gauge potential*). A_{N-1} is *invariant under change of phases*, that is, given $\xi_{N-1} : P \to \mathbb{R}$ continuous, making a change of phase $\mathbf{y}_{N-1}(\mathbf{r}) = e^{i\,\xi_{N-1}(\mathbf{r})}\mathbf{x}_{N-1}(\mathbf{r})$, it results in $A_{N-1}(\mathbf{r}) = i\mathbf{y}_{N-1}^H(\mathbf{r}) D_{\mathbf{r}} \mathbf{y}_{N-1}(\mathbf{r})$.

If the orthonormal basis $(\mathbf{x}_j(\mathbf{r}))_{j=0}^{N-1}$ for $H(\mathbf{r})$ is only changed by phases, then:

$$\left(\mathbf{y}_j(\mathbf{r}) = e^{i\,\xi_j(\mathbf{r})}\mathbf{x}_j(\mathbf{r})\right)_{j=0}^{N-1},$$

the Berry phase remains invariant. Thus, the Berry phase is invariant under certain transformations $U(1)$ of Hamiltonians.

The *Aharanov-Bohm effect* appears when the Hamiltonian of a magnetic field and the corresponding electric field are moving in a circuit. The geometric Berry phase is not negligible, that is, the system is no-holonomic, and this produce the following effect: an electron beam that passes perpendicularly through a solenoid, forks, surrounding the solenoid to compose later. The relationship between the paths and adiabatic geometric phases is evident by the similarity of the phases involved. The relation (3.12) determines the phase involved in the evolution of an adiabatic path, while relation (3.21) determines properly the geometric Berry phase (3.22), when the path followed by a Hamiltonian is a circuit.

There are other geometric phases, such as the *Aharonov-Anandan*, *Pancharatnam* or specific techniques such as NMR.

3.4.2 Geometric quantum computation

The Adiabatic Theorem provides an approximation to the instantaneous ground state of a Hamiltonian, but with the exception of a global phase, this phase can be divided into two parts: the dynamic phase and the geometric Berry phase, see equations (3.17) and (3.18), respectively. The Berry phase depends only on the path taken, not on how fast the path is traversed. Hence, if we design a cyclic path of Hamiltonians, the Berry phase is totally determined.

In *Geometric Quantum Computing* (GQC) [86, 56, 77] it is used the well known fact in differential geometry that arises when a vector is parallel transported around a loop on a smooth manifold (see figure 3.1). This vector may return rotated although there has no been local rotation along the loop. This global rotation is the *Holonomy* caused by the curvature of the underlaying space. In QM a state vector can be transported without locally rotating it around a loop in some quantum parameter space, and the resulting transformation has the same effect as applying a unitary matrix or phase factor that depends only on the global geometry of the loop.

In contrast to AQC that encodes the solution of an optimization problem into the ground state of a Hamiltonian, GQC encodes the solution of the problem into the Berry phase of the final state. In [87] shows an adiabatic algorithm for the *Counting Problem*, such that the solution is encoded in the Berry phase of the final state,

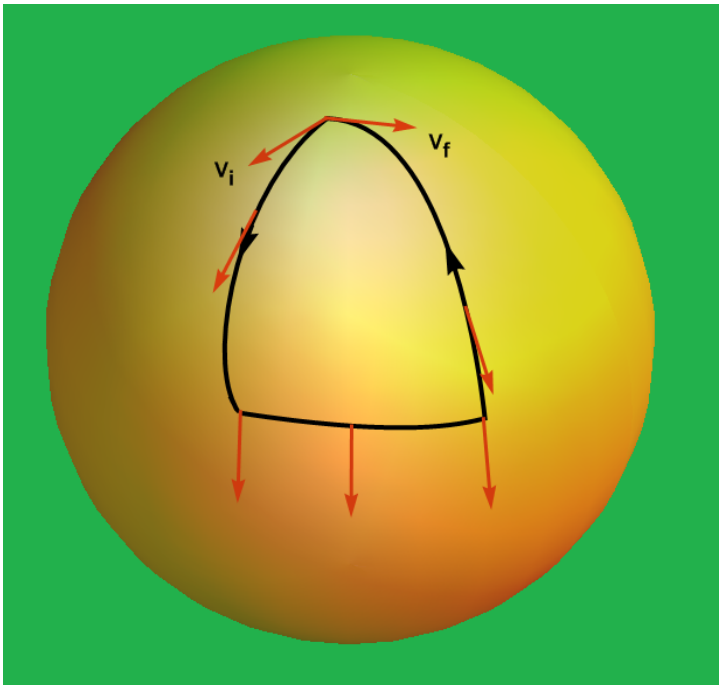

Figure 3.1: Parallel transport of a vector without local rotation on a curved surface (in a sphere). The final vector $\mathbf{v}_f$ has been rotated with respect to the initial vector $\mathbf{v}_i$, and the rotation angle being the solid angle enclosed by the loop.

rather than the ground state of the final state. The final information is obtained by estimating the relative phase, rather than the usual quantum measurement.

There are several methods that have been proposed to build quantum gates based on geometric Berry phases. In [34] it was shown a geometric quantum computation scheme based on laser manipulation of a set of trapped ions. In [49] a controlled phase shift gate was proposed by performing a nuclear magnetic resonance experiment in which a conditional Berry phase is implemented. Since, the geometric Berry phase depends only on the geometry of the path executed, this suggest the possibility of an intrinsically fault-tolerant way of performing quantum gate operations (see also [84, 63]).

The GQC provides a scheme to design new quantum algorithms that are fault-tolerant for some kind of source of errors, but it depends on specific physical implementation such as NMR techniques. There is not yet a general technique to codify the solution of hard problems into the geometric Berry phase. An important problem is to propose a quantum algorithm based on the geometric Berry phase for the search problem in a database with time complexity equal to the proposed in [43].

In this thesis we do not consider the influence of the Berry phase in the adiabatic evolution, and in general we assume that the adiabatic paths follow an arbitrary trajectory.

Chapter 4

Efficient Hamiltonian construction

In this chapter we propose a procedural construction of the Hamiltonian operators for AQC to avoid the direct tensor product construction. A complete treatment of AQC applied to the MAX-SAT problem is given, the initial and final Hamiltonian are constructed in order to simulate AQC in a computationally efficient way (see [32]).

The procedural construction of the Hamiltonian operators for AQC can be generalized for other optimization problems with a similar structure and it can be used in other applications such as Hamiltonian simulations and numerical analysis of the eigenvalue paths in the evolution of AQC.

This computational analysis can help to track the involved complexity of the AQC which is the topic of this thesis.

4.1 AQC applied to the MAX-SAT problem

In AQC, given a problem Π and any instance $\mathbf{x}$ of size n, a pair of Hamiltonians in a n-dimensional Hilbert space is determined. Each Hamiltonian corresponds to a Hermitian matrix represented by a $(2^n \times 2^n)$-complex matrix and in general corresponds to a sparse matrix. Two problems arise with the computational simulation of AQC: the first one is that the amount of memory to store the two Hamiltonians becomes impractical for most of the actual computers, and the number of operations grows exponentially.

In order to deal with these two problems, we describe and characterize in a procedural way every entry of the initial and final Hamiltonians. Such a characterization of the Hamiltonians can reduce the amount of used memory to half.

Here we follow a SAT coding similar to the already standard codings [37, 46] into AQC. We will consider 3-SAT: The satisfiability decision problem for 3-clauses. And we provide a procedural construction of the initial and final Hamiltonian for the given instances.

4.1.1 Satisfiability Problem

Let $\mathcal{X} = (X_j)_{j=0}^{n-1}$ be a set of n Boolean variables. A *literal* has the form X^δ, with $X \in \mathcal{X}$, and $\delta \in \{0,1\}$: $X^1 = X$ and $X^0 = \neg X$. A *clause* is a disjunction of literals, and a *conjunctive form* (CF) is a conjunction of clauses. An assignment is a point $\varepsilon = (\varepsilon_j)_{j=1}^n \in \{0,1\}^n$ in the n-dimensional hypercube. Such an assignment *satisfies* the literal X_j^δ if and only if $\varepsilon_j = \delta$; it *satisfies* a clause whenever it satisfies a literal in the clause; and it *satisfies* a CF whenever it satisfies all clauses in the CF. An *m-clause* is a clause consisting of exactly m literals, and an *m-CF* is a CF consisting just of m-clauses.

The *satisfiability problem* SAT consists of deciding whether a given CF has a satisfying assignment. SAT is NP-complete and 3-SAT (the restriction of SAT to 3-CF's) is also NP-complete.

For any clause C, let $h_C : \{0,1\}^n \to \mathbb{R}$, be the map such that

$$\begin{aligned} \varepsilon \text{ satisfies } C &\implies h_C(\varepsilon) = 0, \\ \varepsilon \text{ does not satisfy } C &\implies h_C(\varepsilon) = 1. \end{aligned}$$

And for any CF $\phi = (C_i)_{i=0}^{m-1}$ let $h_\phi : \{0,1\}^n \to \mathbb{R}$ be $h_\phi = \sum_{i=0}^{m-1} h_{C_i}$. Clearly:

$$\forall \varepsilon \in \{0,1\}^n : \ [h_\phi(\varepsilon) = 0 \iff \varepsilon \text{ satisfies } \phi],$$

thus deciding the satisfiability of ϕ is reduced to decide whether the global minimum of h_ϕ is 0.

4.1.2 AQC formulation of SAT

Let $|0\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ and $|1\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ be the vectors in the canonical basis of the Hilbert space $\mathbb{H}_1 = \mathbb{C}^2$. Let, for each $n > 1$, $\mathbb{H}_n = \mathbb{H}_{n-1} \otimes \mathbb{H}_1$ be the n-fold tensor power of $\mathbb{H}_1$. A basis of $\mathbb{H}_n$ is $(|\varepsilon\rangle)_{\varepsilon \in \{0,1\}^n}$ where

$$\varepsilon = (\varepsilon_j)_{j=1}^n \implies |\varepsilon\rangle = \bigotimes_{j=1}^n |\varepsilon_j\rangle .$$

Let $\sigma_z : \mathbb{H}_1 \to \mathbb{H}_1$ be the Pauli quantum gate with matrix $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. For any bit $\delta \in \{0,1\}$ let $\tau_{\delta z} = \frac{1}{2}(I_2 - (-1)^\delta \sigma_z)$. Independently of δ, the characteristic polynomial of $\tau_{\delta z}$ is $p_z(\lambda) = (\lambda - 1)\lambda$ and its eigenvalues are 0 and 1 with unit eigenvectors $|0\rangle$ and $|1\rangle$. The correspondence among eigenvalues and eigenvectors is determined by δ, namely:

$$\forall \varepsilon \in \{0,1\} : \quad \tau_{\delta z} |\varepsilon\rangle = (\delta \oplus \varepsilon) |\varepsilon\rangle , \tag{4.1}$$

in words: if $\delta = 0$ the index of each eigenvector coincides with the eigenvalue, otherwise, it is the complementary value. Thus, the zero eigenvalue of the map $\tau_{\delta z}$ corresponds to the eigenvector e_δ.

For any $\delta \in \{0,1\}$ and $j_1 \in [\![1,n]\!]$, let $R_{E\delta j_1 n} = \bigotimes_{j_2=1}^{n} \rho_{z\delta j_2} : \mathbb{H}_n \to \mathbb{H}_n$ where $\rho_{z\delta j_2} = \text{identity}_{\mathbb{H}_1}$ if $j_2 \neq j_1$ and $\rho_{z\delta j_1} = \tau_{\delta z}$, thus the effect of $R_{E\delta jn}$ in an n-quregister is to apply $\tau_{\delta z}$ to the j-th qubit. Consequently,

$$\forall \varepsilon \in \{0,1\}^n : \quad R_{E\delta jn}\left(|\varepsilon\rangle\right) = (\delta \oplus \varepsilon_j)\, |\varepsilon\rangle\,, \tag{4.2}$$

thus the zero eigenvalue corresponds to the basic vectors giving a satisfying assignment for the literal X_j^{δ}. Given a 3-clause $C = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$ let

$$H_{EC} = R_{E\delta_3 j_3 n} \circ R_{E\delta_2 j_2 n} \circ R_{E\delta_1 j_1 n} : \mathbb{H}_n \to \mathbb{H}_n.$$

Thus, for any $\varepsilon \in \{0,1\}^n$, $H_{EC}\left(|\varepsilon\rangle\right) = 0$ if and only if ε satisfies the clause C; and it coincides with the linear map that on the basis vectors acts as $|\varepsilon\rangle \mapsto h_C(\varepsilon)\,|\varepsilon\rangle$. Thus, if $x = \sum_{\varepsilon\in\{0,1\}^n} x_\varepsilon\,|\varepsilon\rangle$ then $H_{EC}(x) = \sum_{\varepsilon\in\{0,1\}^n} x_\varepsilon h_C(\varepsilon)\,|\varepsilon\rangle$ and

$$\langle x|H_{EC}(x)\rangle = \sum_{\varepsilon\in\{0,1\}^n} \overline{x_\varepsilon} x_\varepsilon h_C(\varepsilon) = \sum_{\varepsilon\in\{0,1\}^n} |x_\varepsilon|^2 h_C(\varepsilon) \geq 0. \tag{4.3}$$

Hence H_{EC} is a positive operator. Indeed, we have $\langle x|H_{EC}(x)\rangle = 0$ if and only if $H_{EC}(x) = 0 \in \mathbb{H}_n$, and this happens if and only if x is a linear combination of those basic vectors indexed by assignments satisfying the clause C.

For a given CF $\phi = (C_i)_{i=0}^{m-1}$ let $H_{E\phi} : \mathbb{H}_n \to \mathbb{H}_n$ be $H_{E\phi} = \sum_{i=0}^{m-1} H_{EC_i}$. Again, $H_{E\phi}$ is positive and $H_{E\phi}(x) = 0$ if and only if x is a linear combination of those basic vectors indexed by assignments satisfying the CF ϕ.

An unit n-quregister $x \in \mathbb{H}_n$ such that $H_{E\phi}(x) = 0$ is called a *ground state* for $H_{E\phi}$. Thus:

Remark 1. *In order to find a satisfying assignment for ϕ it is sufficient to find a ground state for $H_{E\phi}$.*

Let $\sigma_x : \mathbb{H}_1 \to \mathbb{H}_1$ be the Pauli quantum gate with matrix $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. The map $\tau_{\delta x} = \frac{1}{2}(I_2 - (-1)^{\delta}\sigma_x)$ also has, independently of δ, characteristic polynomial $p_x(\lambda) = (\lambda - 1)\lambda$ and its eigenvalues are 0 and 1, now with corresponding unit eigenvectors $c_0 = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$ and $c_1 = \frac{1}{\sqrt{2}}\left(-|0\rangle + |1\rangle\right)$, which form an orthonormal basis of $\mathbb{H}_1$. The correspondence among eigenvalues and eigenvectors is determined as in relation (4.1) by δ, namely:

$$\forall \varepsilon \in \{0,1\} : \quad \tau_{\delta x} c_\varepsilon = (\delta \oplus \varepsilon) c_\varepsilon. \tag{4.4}$$

Let us also make

$$\varepsilon = (\varepsilon_j)_{j=1}^{n} \implies c_\varepsilon = \bigotimes_{j=1}^{n} c_{\varepsilon_j}.$$

For any $j_1 \in [\![1,n]\!]$, let $R_{Z\delta j_1 n} = \bigotimes_{j_2=1}^{n} \mu_{\delta j_2} : \mathbb{H}_n \to \mathbb{H}_n$ where $\mu_{\delta j_2} = \text{identity}_{\mathbb{H}_1}$ if $j_2 \neq j_1$ and $\mu_{\delta j_1} = \tau_{\delta x}$, thus the effect of $R_{Z\delta jn}$ in an n-quregister is to apply $\tau_{\delta x}$ to the j-th qubit. Consequently, as in relation (4.2):

$$\forall \varepsilon \in \{0,1\}^n : \quad R_{Z\delta jn}\left(c_\varepsilon\right) = (\delta \oplus \varepsilon_j)\, c_\varepsilon. \tag{4.5}$$

Hence whenever $\varepsilon_j = \delta$, c_ε is a ground state of the operator $R_{Z\delta jn}$.

Let us consider $\delta = 0$ and let us write $R_{Zjn} = R_{Z0jn}$. Given a 3-clause $C = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$ let $H_{ZC} = R_{Zj_1n} + R_{Zj_2n} + R_{Zj_3n} : \mathbb{H}_n \to \mathbb{H}_n$. Then H_{ZC} does not depend on the "signs" $\delta_{j_1}, \delta_{j_2}, \delta_{j_3}$ of the literals, but just on the variables appearing in the clause. The following implication holds:

$$[\varepsilon_{j_1} = \varepsilon_{j_2} = \varepsilon_{j_3} = 0 \implies H_{ZC}(z_\varepsilon) = 0].$$

Given a CF $\phi = (C_i)_{i=0}^{m-1}$ let $H_{Z\phi} : \mathbb{H}_n \to \mathbb{H}_n$ be $H_{Z\phi} = \sum_{i=0}^{m-1} H_{ZC_i}$.

Remark 2. *From relation of equation (4.5),* $c_{00\cdots0} = \frac{1}{2^{\frac{n}{2}}} \sum_{\varepsilon\in\{0,1\}^n} |\varepsilon\rangle$ *is a ground state of* $H_{Z\phi}$.

Remark 3. *The following equation holds:*

$$H_{Z\phi} = \sum_{j=1}^{n} d_j R_{Zjn} \tag{4.6}$$

where, for each $j \in [\![1, n]\!]$, $d_j = card\{i \in [\![1, m]\!] |\ X_j$ *appears in* $C_i\}$.

From remark 2 we have that there is a "natural" ground state, $c_{00\cdots0}$, for the operator $H_{Z\phi}$, while, after remark 19, to solve the SAT instance given by ϕ it is necessary to find a ground state for the operator $H_{E\phi}$. In summary, $c_{00\cdots0}$ is a ground state for $H_{Z\phi}$ but our aim is to find a ground state for $H_{E\phi}$.

For any 3-clause C, let us consider the map $I \to \mathrm{GL}(\mathbb{H}_n)$, where $\mathrm{GL}(\mathbb{H}_n)$ is the group of invertible linear automorphisms of the space $\mathbb{H}_n$, and $I = [0, 1]$ is the unit real interval, given as $t \mapsto H_C(t) = (1 - t)H_{ZC} + tH_{EC}$.

For a CF $\phi = (C_i)_{i=0}^{m-1}$, let

$$H_\phi : t \mapsto H_\phi(t) = \sum_{i=0}^{m-1} H_{C_i}(t) = \sum_{i=0}^{m-1} [(1 - t)H_{ZC} + tH_{EC}].$$

Let

$$\forall t \in [0, 1] :\ i\frac{d}{dt}\psi(t) = H_\phi(t)\psi(t). \tag{4.7}$$

be the proper Schrödinger equation, with Hamiltonian H_ϕ.

Let $\{\eta_\nu\}_{\nu=0}^{2^n-1} \subset (\mathbb{R}^I)^{2^n}$ be the sequence of curves giving the eigenvalues of H_ϕ (indexed according to their absolute values at the initial points for $t = 0$). Then it is possible to see that η_0 and η_1 never cross on I, and, by the Adiabatic Theorem, there exists a $t_0 > 0$ such that the solutions ψ_{t_0} of the "scaled" equation

$$\forall t \in [0, t_0] :\ i\frac{d}{dt}\psi_{t_0}(t) = H_\phi\left(\frac{t}{t_0}\right)\psi_{t_0}(t) \tag{4.8}$$

are such that $\psi_{t_0}(t)$ gets arbitrarily close, as $t \nearrow t_0$, to a ground state for $H_{E\phi}$. A measurement of such ground state provides an assignment that either satisfies ϕ or maximizes the number of satisfied clauses in ϕ.

4.2 Procedural Hamiltonian construction

In the following we describe a procedural construction of the Hamiltonian operators H_E and H_Z defined in section 4.1.2.

4.2.1 Hyperplanes in the hypercube

Let us enumerate the n-dimensional hypercube with indexes in $[\![0, 2^n - 1]\!]$ associating each $i \in [\![0, 2^n - 1]\!]$ with its length n big-endian base-2 representation:

$$i \leftrightarrow \texttt{rev}\left((i)_2\right) = (\varepsilon_0, \ldots, \varepsilon_{n-1}) \in \{0,1\}^n \quad \text{where } i = \sum_{\nu=0}^{n-1} \varepsilon_\nu 2^\nu. \tag{4.9}$$

By putting each such representation as the i-th row of a rectangular array, a $(2^n \times n)$-matrix $\mathbf{E} \in \{0,1\}^{2^n \times n}$ is obtained. Let us denote by $\mathbf{e}_j^{(1)} \in \{0,1\}^{2^n}$ its j-th column, $j = 0, \ldots, n-1$. On one side, $\mathbf{e}_j^{(1)}$ can be written as the list $(0^{2^j}1^{2^j})^{2^{n-1-j}} = \mathbf{e}_j^{(1)}$, and on the other hand it can be seen as the Boolean map that has as support the hyperplane $E_j^1 : \varepsilon_j = 1$. Let $\mathbf{e}_j^{(0)}$ be the 2^n-vector obtained from $\mathbf{e}_j^{(1)}$ by taking the complement value at each entry. Then $\mathbf{e}_j^{(0)} = (1^{2^j}0^{2^j})^{2^{n-1-j}}$, and it represents the Boolean map with support the hyperplane $E_j^0 : \varepsilon_j = 0$. Clearly:

Remark 4. *Each hyperplane E_j^δ is a $(n-1)$-dimensional affine variety at the hypercube and its characteristic map can be written as the list*

$$\mathbf{e}_j^{(\delta)} = (\overline{\delta}^{2^j}\delta^{2^j})^{2^{n-1-j}}.$$

The lists $\mathbf{e}_j^{(\delta)}$ are easily computable:
Procedure $(n-1)$-`DimensionalVarieties`.
Input: $\delta \in \{0,1\}$, $j \in [\![0, n-1]\!]$ and $k \in [\![0, 2^n - 1]\!]$.
Output: The k-th entry of the list $\mathbf{e}_j^{(\delta)}$.

1. Let $k_0 := k \bmod (2^{n-1-j})$.

2. If $k_0 \geq 2^j$ then output δ else output $\overline{\delta}$.

Two $(n-1)$-dimensional affine varieties are *parallel* if they are of the form E_j^0 and E_j^1, for some index $j \in [\![0, n-1]\!]$.

Remark 5. *The intersection of two parallel $(n-1)$-dimensional varieties is empty, while the intersection of any two non-parallel $(n-1)$-dimensional varieties is a $(n-2)$-dimensional affine variety, thus the intersection of any two non-parallel $(n-1)$-dimensional varieties has cardinality 2^{n-2}. Also, the intersection of three pairwise non-parallel $(n-1)$-dimensional affine varieties has cardinality 2^{n-3}.*

4.2.2 The Hamiltonian operator H_E

For any $\delta \in \{0,1\}$ and $j \in [\![0, n-1]\!]$, the transform $R_{E\delta jn} : \mathbb{H}_n \to \mathbb{H}_n$ defined in section 4.1.2, being the tensor product of transforms represented by diagonal matrices with respect to the canonical basis, is represented, with respect to the basis $(|\varepsilon\rangle)_{\varepsilon\in\{0,1\}^n}$, by a diagonal matrix. Indeed:

Remark 6. *The 2^n-length diagonal determining the diagonal matrix of $R_{E\delta jn}$ coincides with the list $\mathbf{e}_j^{(\delta)} = (\overline{\delta}^{2^j}\delta^{2^j})^{2^{n-1-j}}$.*

For a 3-clause $C = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$, the operator $H_{EC} = R_{E\delta_3 j_3 n} \circ R_{E\delta_2 j_2 n} \circ R_{E\delta_1 j_1 n}$ is also represented by a diagonal matrix and its diagonal is the componentwise product of the lists $\mathbf{e}_{j_1}^{(\delta_1)}$, $\mathbf{e}_{j_2}^{(\delta_2)}$ and $\mathbf{e}_{j_3}^{(\delta_3)}$. Since the indexes j_1, j_2, j_3 are pairwise different, the lists are the characteristic maps of three pairwise non-parallel $(n-1)$-dimensional affine varieties. From remark 5:

Remark 7. *With respect to the canonical basis $(|\varepsilon\rangle)_{\varepsilon\in\{0,1\}^n}$ of $\mathbb{H}_n$, for any 3-clause $C = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$, the operator H_{EC} is represented by a diagonal matrix and its diagonal, $D_C(C) = D_C((j_1,\delta_1),(j_2,\delta_2),(j_3,\delta_3))$, consisting of 2^{n-3} 1's, is such that each entry can be calculated by a slight modification of the procedure $(n-1)$-*`DimensionalVarieties`* outlined above. Namely:*

Procedure `3-ClauseDiagonal`.
Input: A 3-clause $C = \{(j_1,\delta_1),(j_2,\delta_2),(j_3,\delta_3)\}$, and $k \in [\![0, 2^n-1]\!]$.
Output: The k-th entry of the list D_C.

1. For $r = 1$ to 3 do
   (a) $k_{r0} := k \bmod (2^{n-1-j_r})$.
   (b) If $k_{r0} \geq 2^{j_r}$ then $x_r := \delta_r$ else $x_r := \overline{\delta}_r$.
2. Output $x_1 \cdot x_2 \cdot x_3$.

Remark 8. *With respect to the canonical basis $(|\varepsilon\rangle)_{\varepsilon\in\{0,1\}^n}$ of $\mathbb{H}_n$, for any CF $\phi = (C_i)_{i=0}^{m-1}$ the operator $H_{E\phi} = \sum_{i=0}^{m-1} H_{EC_i}$ is represented by a diagonal matrix, and its diagonal is $D_F(\phi) = \sum_{i=0}^{m-1} D_C(C_i)$.*

For any 3-clause C, let $\mathrm{Spt}_C(C) = \{j \in [\![0, 2^n-1]\!] |\ D_C(C)[j] \neq 0\}$ be the collection of indexes corresponding to non-zero entries at the vector in the diagonal $D_C(C)$. Then $\mathrm{card}(\mathrm{Spt}_C(C)) = 2^{n-3}$. Similarly, let $\mathrm{Spt}_F(\phi)$ be the collection of indexes corresponding to non-zero entries at the vector in the diagonal $D_F(\phi)$. Clearly:

$$\phi = (C_i)_{i=0}^{m-1} \implies \mathrm{Spt}_F(\phi) = \bigcup_{i=0}^{m-1} \mathrm{Spt}_C(C_i).$$

The entries at $D_F(\phi)$ are the eigenvalues of the operator $H_{E\phi}$, and the satisfying assignments are determined by the eigenvectors corresponding to the zero eigenvalue (if zero indeed is an eigenvalue). From remark 19 the following results:

Remark 9. *Any zero entry in the 2^n-vector $D_F(\phi)$ determines a satisfying assignment for ϕ. Namely, if $D_F(\phi)[i] = 0$ then $\phi(\texttt{rev}\,((i)_2)) = \texttt{True}$.*

This can also be stated as follows:

Remark 10. *For a given CF $\phi = (C_i)_{i=0}^{m-1}$, ϕ is satisfiable if and only if the following happends $Spt_F(\phi) \neq [\![0, 2^n - 1]\!]$.*

Thus, the satisfiability problem can be rephrased as follows:
Problem QASAT.
Instance: A CF $\phi = (C_i)_{i=0}^{m-1}$.
Solution: "Yes" if $\mathrm{Spt}_F(\phi) \neq [\![0, 2^n - 1]\!]$; "No", if $\mathrm{Spt}_F(\phi) = [\![0, 2^n - 1]\!]$.

SAT is thus reducible to QUSAT in polynomial time, consequently QUSAT is NP-complete as well.

As a second construction of the vector at the diagonal $D_C(C)$ for any 3-clause, let us enumerate these clauses in another rather conventional manner.

In a general setting, let $k \geq 3$. Then the number of k-clauses, $C = \bigvee_{j\in J} X_j^{\delta_j}$, with $\mathrm{card}(J) = k$, in n variables, is $\nu_{kn} = \binom{n}{k}2^k$. For any $i \in [\![0, \nu_{kn} - 1]\!]$ let $i_0 = i \bmod 2^k$ and $i_1 = (i - i_0)/2^k$. Then the map $\eta : i \mapsto (i_1, i_0)$ allows us to identify $[\![0, \nu_{kn} - 1]\!]$ with the Cartesian product $[\![0, \binom{n}{k} - 1]\!] \times [\![0, 2^k - 1]\!]$. The map η can also be seen as the function that to each index $i \in [\![0, \nu_{kn} - 1]\!]$ associates the clause $C = \bigvee_{j\in J_{i_1}} X_j^{\delta_j}$ where J_{i_1} is the i_1-th k-set of $[\![0, n - 1]\!]$ and $i_0 = \sum_{\kappa=0}^{k-1} \delta_{j_\kappa} 2^\kappa$.

Remark 11. *Let $C = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$ be a 3-clause, $0 \leq j_1 < j_2 < j_3 < n$. Then the collection $Spt_C(C)$ of indexes corresponding to non-zero entries at $D_C(C)$ is characterized as follows: For any $k \in [\![0, 2^n - 1]\!]$, $k \in Spt_C(C) \Longleftrightarrow$*

$$\begin{aligned}
&\exists (k_0, k_1, k_2, k_3) \in K : \\
&(k_1 = \delta_1 \bmod 2)\,\&\,(k_2 = \delta_2 \bmod 2)\,\&\,(k_3 = \delta_3 \bmod 2)\,\& \\
&k = k_0 + 2^{j_1}k_1 + 2^{j_2}k_2 + 2^{j_3}k_3
\end{aligned}$$

where $K = [\![0, 2^{j_1} - 1]\!] \times [\![0, 2^{j_2-j_1} - 1]\!] \times [\![0, 2^{j_3-j_2} - 1]\!] \times [\![0, 2^{n-j_3} - 1]\!]$.

The remark 11 is consistent with the calculated cardinality of $\mathrm{Spt}_C(C)$ because: $2^{n-3} = 2^{j_1}2^{j_2-j_1-1}2^{j_3-j_2-1}2^{n-j_3-1}$. And also, it justifies an algorithm to compute $D_C(C)$. Namely:
Procedure `3-ClauseDiagonalBis`.
Input: A 3-clause $C = \{(j_1, \delta_1), (j_2, \delta_2), (j_3, \delta_3)\}$, and $k \in [\![0, 2^n - 1]\!]$.
Output: The k-th entry of the list D_C.

1. $flg :=$ `True` ; $crk := k$;
2. $k_0 := crk \bmod 2^{j_1}$; $crk := (crk - k_0)/2^{j_1}$;
3. $k_1 := crk \bmod 2^{j_2-j_1}$; $crk := (crk - k_1)/2^{j_2-j_1}$;

4. $\mathit{flg} := (k_1 == \delta_1 \bmod 2)$;
5. If flg then
   (a) $k_2 := crk \bmod 2^{j_3-j_2}$; $crk := (crk - k_2)/2^{j_3-j_2}$;
   (b) $\mathit{flg} := (k_2 == \delta_2 \bmod 2)$;
   (c) If flg then
      i. $k_3 := crk \bmod 2^{j_3-j_2}$; $crk := (crk - k_3)/2^{n-j_3}$;
      ii. $\mathit{flg} := (k_3 == \delta_3 \bmod 2)$;
6. If flg then $b := 1$ else $b := 0$;
7. Output b.

4.2.3 The Hamiltonian operator H_{Z_ϕ}

Now let us consider the operators with subindex Z defined in section 4.1.2.

Let us define the following matrices:

$$\begin{array}{rclcrcl} A_0 &=& [1] & ; & B_0 &=& [1] \\ A_1 &=& I_2 \otimes A_0 - \frac{1}{2}\sigma_x \otimes B_0 & ; & B_1 &=& I_2 \otimes B_0 \\ A_2 &=& I_2 \otimes A_1 - \frac{1}{2}\sigma_x \otimes B_1 & ; & B_2 &=& I_2 \otimes B_1 \\ A_3 &=& I_2 \otimes (\frac{1}{2}B_2 + A_2) - \frac{1}{2}\sigma_x \otimes B_2 & ; & B_3 &=& I_2 \otimes B_2 \end{array} \tag{4.10}$$

where I_2 is the (2×2)-identity matrix. For each $k \leq 3$, A_k, B_k are matrices of order $(2^k \times 2^k)$, indeed we have $B_k = I_{2^k}$.

For $n = 3$ and any 3-clause $C_{012} = X_0^{\delta_0} \vee X_1^{\delta_1} \vee X_2^{\delta_2}$ involving the three variables, the transform $H_{ZC_{012}} : \mathbb{H}_3 \to \mathbb{H}_3$ is represented, with respect to the canonical basis of $\mathbb{H}_3$, by the matrix

$$H_{[012],3} = A_3 = \frac{1}{2}\begin{pmatrix} 3 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 3 & -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 3 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 3 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & -1 & 3 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 0 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & -1 & 3 \end{pmatrix}. \tag{4.11}$$

which is a band matrix with the following properties: its upper-right boundary is its diagonal at distance $4 = 2^{3-1}$ above the main diagonal, the lower-left boundary is also at distance 4 below the main diagonal, the main diagonal has constant value $\frac{3}{2}$ and the only values appearing in the matrix are $\frac{3}{2}, 0, -\frac{1}{2}$.

Naturally, for any $n > 3$ the transform $H_{ZC_{012}} : \mathbb{H}_n \to \mathbb{H}_n$ is represented by the matrix

$$H_{[012],n} = H_{[012],n-1} \otimes I_2. \tag{4.12}$$

The tensor product at eq. (4.12) substitutes each current entry at $H_{[012],n-1}$ by the product of that entry by the (2×2)-identity matrix. Thus also $H_{[012],n}$ is a band matrix, its boundaries are diagonals at distance 2^{n-1} from the main diagonal, the main diagonal has constant value $\frac{3}{2}$ and the only values appearing in the matrix are $\frac{3}{2}, 0, -\frac{1}{2}$. The following algorithm results:

Procedure `HZ012`.
Input: An integer $n \geq 3$ and a pair $(i, j) \in [\![0, 2^n - 1]\!]^2$.
Output: The (i, j)-th entry of the matrix $H_{[012],n}$.

1. $k := j - i$;
2. Case k of

   0 : $v := \frac{3}{2}$.
   $\pm 2^\ell$: (a power of 2, with $\ell \geq 1$)
      i. $\iota := \min\{i, j\}$;
      ii. $\iota_0 := \iota \bmod 2^{\ell+1}$;
      iii. If $\iota_0 < 2^\ell$ Then $v := -\frac{1}{2}$ Else $v := 0$;
   Else: $v := 0$;

3. Output v.

For an arbitrary 3-clause $C_{j_1 j_2 j_3} = X_{j_1}^{\delta_1} \vee X_{j_2}^{\delta_2} \vee X_{j_3}^{\delta_3}$, with $0 \leq j_1 < j_2 < j_3 < n$ let $\pi_{j_1 j_2 j_3}$ be a permutation $[\![0, n-1]\!] \to [\![0, n-1]\!]$ such that $j_1 \mapsto 0$, $j_2 \mapsto 1$, $j_3 \mapsto 2$ and the restriction $\pi_{j_1 j_2 j_3}|_{[\![0,n-1]\!] - \{j_1, j_2, j_3\}}$ is a bijection $[\![0, n-1]\!] - \{j_1, j_2, j_3\} \to [\![3, n-1]\!]$. Then, there is a permutation $\rho_{j_1 j_2 j_3} : [\![0, 2^n - 1]\!] \to [\![0, 2^n - 1]\!]$, which can be determined in terms of $\pi_{j_1 j_2 j_3}$, such that the matrix $H_{[j_1 j_2 j_3],n}$ representing the transform $H_{ZC_{j_1 j_2 j_3}}$ is the action of $\rho_{j_1 j_2 j_3}$ over rows and columns on the matrix $H_{[012],n}$. Namely, let

$$\rho_{j_1 j_2 j_3} : [\![0, 2^n - 1]\!] \to [\![0, 2^n - 1]\!] \ , \ \sum_{\kappa=0}^{n-1} \varepsilon_\kappa 2^\kappa \mapsto \sum_{\kappa=0}^{n-1} \varepsilon_{\pi_{j_1 j_2 j_3}(\kappa)} 2^\kappa, \tag{4.13}$$

then when writing $H_{[012],n} = \left[h_{ij}^{(0)} \right]_{0 \leq i,j \leq 2^n - 1}$ one has

$$H_{[j_1 j_2 j_3],n} = \left[h^{(0)}_{\rho_{j_1 j_2 j_3}(i)\, \rho_{j_1 j_2 j_3}(j)} \right]_{0 \leq i,j \leq 2^n - 1}.$$

The following algorithm results:

Procedure `HZFor3Clauses`.
Input: An integer $n \geq 3$, a 3-clause $C = \{(j_1, \delta_1), (j_2, \delta_2), (j_3, \delta_3)\}$ and a pair $(i, j) \in [\![0, 2^n - 1]\!]^2$.
Output: The (i, j)-th entry of the matrix $H_{[j_1 j_2 j_3],n}$.

1. Compute the permutation $\rho_{j_1 j_2 j_3} : [\![0, n-1]\!] \to [\![0, n-1]\!]$ as in (4.13) ;

2. Output `HZ012`$[n; (\rho_{j_1 j_2 j_3}(i), \rho_{j_1 j_2 j_3}(j))]$.

(Evidently, the permutation $\rho_{j_1 j_2 j_3}$ can be computed as a preprocess to be used later for several entries (i, j).)

For a CF $\phi = (C_i = \{(j_{i1}, \delta_{i1}), (j_{i2}, \delta_{i2}), (j_{i3}, \delta_{i3}))_{i=0}^{m-1}$, the Hamiltonian operator $H_{Z\phi} : \mathbb{H}_n \to \mathbb{H}_n$ is represented by the matrix $H_{\phi,n} = \sum_{i=0}^{m-1} H_{[j_{i1} j_{i2} j_{i3}],n}$. Thus it can be computed directly by an iteration of algorithm `HZFor3Clauses`.

Remark 12. *The ground eigenvector of the matrix $H_{\phi,n}$ will tend to a ground state of the matrix $H_{E\phi}$ when solving the Schrödinger equation (4.8), providing thus a solution of SAT for the instance ϕ.*

In this chapter we have provide procedures to construct the initial and final Hamiltonians H_E and H_Z, respectively. The procedure `3-ClauseDiagonal` construct the diagonal elements of the operator H_{EC} for each clause C, as the intersection of three pairwise non-parallel $(n-1)$-dimensional varieties (see remark 5). The diagonal elements of $H_{E\phi}$ is the pairwise addition of the diagonal elements of the operators H_{EC} for every clause C. Although we have described the diagonal elements of $H_{E\phi}$, this construction still requires an exponential number of operations, namely $m2^n$ where m is the number of clauses and n is the number of Boolean variables.

On the other hand, in section 4.2.3 it is shown a recursive construction of the Hamiltonian H_Z that corresponds to a sparse matrix. The procedure `HZFor3Clauses` returns the (i, j)-th entry of the matrix $H_{[j_1 j_2 j_3],n}$ and for a 3-CF ϕ the Hamiltonian operator $H_{Z\phi}$ is constructed by iterative calls to the procedure `HZFor3Clauses`.

In chapter 5 we show two constructions of the initial Hamiltonian operator for AQC, one based on the Pauli matrices and other on the Hadamard transform.

Chapter 5

AQC for pseudo-Boolean optimization

The pseudo-Boolean maps [21] appears naturally in many areas of mathematics such as in combinatorial optimization, operation research, integer programming and artificial intelligence. Its importance is in modeling many branches of optimization problems into a general scheme based on quadratic forms. This general scheme can be used to design AQC algorithms by means of the Adiabatic Theorem in order to optimize the underlying pseudo-Boolean maps.

In this chapter the *pseudo-Boolean maps* are introduced to build a general model for optimization in combinatorial problems. In particular the quadratic pseudo-Boolean maps are described in order to express combinatorial graph problems. We develop the Adiabatic Quantum Optimization (AQO) for pseudo-Boolean maps and we prove that the defined Hamiltonians are two-local.

We also give a general algorithm to transform any Monadic Second Order Logic sentence into a pseudo-Boolean map, and its corresponding optimization problem for AQO. This part can be seen independently from the above results, and it can be considered as a general framework for optimization that is not restricted to graph problems.

In the background the basis for AQC given in chapter 4 is assumed.

5.1 Basic transformations

Let $Q = \{0, 1\}$ be the set of the integer values 0 and 1. A *Boolean function* on n variables is a function on Q^n into Q^n, where n is a positive integer and Q^n denotes the n-fold Cartesian product of Q with itself.

A *pseudo-Boolean map* of n variables is a function $f : Q^n \to \mathbb{R}$, where n is a positive integer. Consider the following problem:

Pseudo-Boolean Optimization
Instance: A pseudo-Boolean map $f : Q^n \to \mathbb{R}$.
Solution: A minimum point $x^* = \arg\min_{x \in Q^n} f(x)$.

In the following we will deal with the Pseudo-Boolean Optimization Problem.

The set of n Boolean variables will be denoted as $X = \{x_i : 0 \leq i \leq n-1\}$ and the set of literals will be denoted as $L = \{x_i, \overline{x}_i : 0 \leq i \leq n-1\}$ where $\overline{x}_i := 1 - x_i$.

Note that, Q^n is in correspondence with the power set of $[\![0, n-1]\!]$. The following theorem asserts that any pseudo-Boolean map can be expressed as a real-valued map from $\mathcal{P}([\![0, n-1]\!])$ to $\mathbb{R}$.

Theorem 2. *For every pseudo-Boolean map $f : Q^n \to \mathbb{R}$ on n Boolean variables, there exists a unique mapping $c : \mathcal{P}([\![0, n-1]\!]) \to \mathbb{R}$ such that*

$$f(X) = \sum_{S \in \mathcal{P}([\![0,n-1]\!])} c(S) \prod_{j \in S} x_j. \tag{5.1}$$

See a proof in [21].

The size of the largest subset $S \in \mathcal{P}([\![0, n-1]\!])$ for which $c(S) \neq 0$ is called the degree of f, and is denoted by $\deg(f)$.

Remark 13. *Every pseudo-Boolean map has a unique multilinear polynomial representation as in equation (5.1).*

A *posiform* is a polynomial expression with non-negative terms of the form:

$$\phi(L) = \sum_{k=0}^{m-1} b_k \left(\prod_{i \in A_k} x_i \right) \left(\prod_{j \in B_k} \overline{x}_j \right) \tag{5.2}$$

where $b_k \in \mathbb{R}$ and $b_k > 0$; $A_k, B_k \subseteq [\![0, n-1]\!]$, $A_k \cap B_k = \emptyset$ and $A_k \cup B_k \neq \emptyset$ for all $k = 0, \ldots, m-1$.

Proposition 1. *Any pseudo-Boolean map can be represented as a posiform.*

Proof. Let $f : Q^n \to \mathbb{R}$ be a pseudo-Boolean map defined as in (5.1). For any $S \subseteq [\![0, n-1]\!]$, let t_S be the term in f determined by S as $t_S = c(S) \prod_{j \in S} x_j$, and let π_S be a permutation of the elements in S. A posiform is an expression in which for every $S \subseteq [\![0, n-1]\!] : c(S) \geq 0$. Thus, if $c(S) < 0$ then t_S can be written as

$$t_S = c(S)(1 - \overline{x}_{\pi_S(0)} - x_{\pi_S(0)}\overline{x}_{\pi_S(1)} - \cdots - x_{\pi_S(0)} \cdots x_{\pi_S(m-2)}\overline{x}_{\pi_S(m-1)})$$

where $m = \text{card } S$. Repeating this transformation for every negative term of f eventually produces a posiform of f. □

From proposition 1, it can be seen that any pseudo-Boolean map can have many different posiforms representing it.

Of particular interest are the quadratic pseudo-Boolean maps $f_{ue} : Q^n \to \mathbb{R}$ (i.e., $\deg(f_{ue}) \leq 2$) expressed by polynomials of the form

$$f_{ue}(X) = \sum_{j \in [\![0,n-1]\!]} u_j x_j + \sum_{\{i,j\} \in [\![0,n-1]\!]^{(2)}} e_{ij}\, x_i x_j, \tag{5.3}$$

for some coefficient vector $u \in \mathbb{R}^n$ and coefficient matrix $e \in \mathbb{R}^{\frac{n(n-1)}{2}}$.

For instance, given a graph $\mathbf{G} = (V, E)$, with $V = [\![0, n-1]\!]$ and $E \subseteq [\![0, n-1]\!]^{(2)}$, a representative quadratic pseudo-Boolean map is obtained as

$$f_{\mathbf{G}}(X) = \sum_{j \in [\![0,n-1]\!]} x_j - \sum_{\{i,j\} \in E} x_i x_j.$$

It can be seen that the problem to find a maximal independent vertex subset in $\mathbf{G}$ is equivalent to maximize the map $f_{\mathbf{G}}(X)$ over the hypercube Q^n.

Also, quadratic maps can be considered over the n-fold Cartesian power of the set $\mathbb{S} = \{-1, +1\}$. In fact:

Proposition 2. *Any maximization problem of a quadratic pseudo-Boolean map over the hypercube Q^n is equivalent to a minimization problem of a quadratic map over the power $\mathbb{S}^n$. In symbols: $\forall e \in \mathbb{R}^{\frac{n(n-1)}{2}},\ u \in \mathbb{R}^n\ \exists w \in \mathbb{R}^n\ \varepsilon \in Q^n$:*

$$\varepsilon = \arg\max_{Q^n} f_{we}(X) \ \Leftrightarrow\ \theta(\varepsilon) = \arg\min_{Q^n} f_{we}(X). \tag{5.4}$$

Proposition 3. *Every pseudo-Boolean function f expressed as in equation (5.1) can be reduced to a quadratic pseudo-Boolean function.*

Let us consider the following algorithm:
Procedure `Reduce`.
Input: A pseudo-Boolean function $S_f = \sum_{S \subseteq [\![0,n-1]\!]} c_S \prod_{j \in S} X_j$.
Output: A quadratic pseudo-Boolean function f_{ue}.

1. $M = 1 + \sum_{S \subseteq [\![0,n-1]\!]} |c_S|$; $m = n$;
2. While $\exists S^* \subseteq [\![0, n-1]\!]$ with ($|S^*| > 2$ & $c_{S^*} \neq 0$) do

   Choose $\{i, j\} \subset S^*$ and let
   $c_{\{i,j\}} := c_{\{i,j\}} + M$;
   $c_{\{i,m+1\}} := -2M$; $c_{\{j,m+1\}} := -2M$;
   $c_{\{m+1\}} := 3M$;
   For all subsets $S \supseteq \{i, j\}$ with $c_S \neq 0$ define
   $c_{(S \setminus \{i,j\}) \cup \{m+1\}} := c_S$;
   $c_S := 0$;
   $m := m + 1$;
3. Output $f_{ue} := \sum_{S \subseteq [\![0,m-1]\!]} c_S \prod_{k \in S} X_k$.

Quadratic pseudo-Boolean maps appear naturally in many areas of mathematics. For instance, consider the following:

Proposition 4. *Any instance of the 3-SAT problem can be reduced to a quadratic pseudo-Boolean map.*

Proof. Let $\phi = (C_i)_{i=0}^{m-1}$ be an instance of the 3-SAT problem i.e. a 3-CF over the set of Boolean variables $\mathcal{X} = \{X_j | 0 \leq j \leq n-1\}$, where $C_i = X_{j_1}^{\delta_{j_1}} \vee X_{j_2}^{\delta_{j_2}} \vee X_{j_3}^{\delta_{j_3}}$. Let $f_\phi : Q^n \to \mathbb{R}$ be the map defined as

$$f_\phi(X) = \sum_{C_i \in \phi} (\delta_{j_1} + (-1)^{\delta_{j_1}} X_{j_1})(\delta_{j_2} + (-1)^{\delta_{j_2}} X_{j_2})(\delta_{j_3} + (-1)^{\delta_{j_3}} X_{j_3})$$

such that $f_\phi(\varepsilon) = 0$ if and only if ε satisfies ϕ, for some assignment $\varepsilon \in Q^n$. By given as input f_ϕ to the algorithm `Reduce`, f_ϕ can be reduced to a quadratic pseudo-Boolean map. □

From proposition 4 it follows that:

Remark 14. *For every 3-CF ϕ, minimizing h_ϕ as defined in section 4.1.1 is equivalent to minimize f_ϕ.*

5.2 AQC for quadratic pseudo-Boolean maps

The adiabatic quantum optimization (AQO) developed in chapter 4 can be generalized as follows:

Let $(|\varepsilon\rangle)_{\varepsilon \in Q^n}$ be an orthonormal basis for $\mathbb{H}_n$ also called the *computational basis.*

Given a pseudo-Boolean map $f : Q^n \to \mathbb{R}$, let us define

$$H_f : \mathbb{H}_n \to \mathbb{H}_n, \qquad H_f = \sum_{\varepsilon \in Q^n} f(\varepsilon) \, |\varepsilon\rangle \langle\varepsilon| \,. \tag{5.5}$$

For any $\mathbf{x} \in \mathbb{H}_n$, if $\mathbf{x} = \sum_{\varepsilon \in Q^n} x_\varepsilon |\varepsilon\rangle$ then $H_f(\mathbf{x}) = \sum_{\varepsilon \in Q^n} x_\varepsilon f(\varepsilon) \, |\varepsilon\rangle$ and consequently

$$\langle \mathbf{x} | H_f(\mathbf{x}) \rangle = \sum_{\varepsilon \in Q^n} |x_\varepsilon|^2 f(\varepsilon). \tag{5.6}$$

From equation (5.6), H_f is a positive operator and $\forall \varepsilon \in Q^n : H_f |\varepsilon\rangle = f(\varepsilon) |\varepsilon\rangle$, then H_f is diagonal in the computational basis, i.e. $\mathrm{diag}(H_f) = (f(\varepsilon))_{\varepsilon \in Q^n}$.

In order to construct explicitly H_f, it is necessary to evaluate the map f at every point in Q^n (see chapter 4).

The Hamiltonian problem H_f for AQO is constructed as a sum of one-dimensional projectors along every possible direction in the computational basis. Now, let us define a more convenient Hamiltonian for quadratic pseudo-Boolean maps.

Let $f_{ue} : Q^n \to \mathbb{R}$ be a quadratic pseudo-Boolean map of the form:

$$f_{ue}(X) = a + \sum_{j \in A} u_j x_j + \sum_{\{i,j\} \in B} e_{ij} x_i x_j \tag{5.7}$$

where $A \subseteq [\![0, n-1]\!]$, $B \subseteq [\![0, n-1]\!]^{(2)}$, and $\forall j \in A : u_j \in \mathbb{R}$, $\forall \{i, j\} \in B : e_{ij} \in \mathbb{R}$ and $a \in \mathbb{R}$.

Remark 15. *Any quadratic pseudo-Boolean map can be represented as in equation (5.7).*

Remark 15 also asserts that any quadratic pseudo-Boolean map has an inherent graph structure.

For any $j \in [\![0, n-1]\!]$ and $\delta \in \{0, 1\}$ let $\sigma_{z,\delta}^{j} = \bigotimes_{\nu=0}^{n-1} s_\nu : \mathbb{H}_n \to \mathbb{H}_n$ where $s_\nu = \frac{1}{2}(I_2 + (-1)^\delta \sigma_z)$ if $\nu = j$ and $s_\nu = \mathrm{Id}$ otherwise.

Let $H_{f_{ue}} : \mathbb{H}_n \to \mathbb{H}_n$ be defined as follows:

$$H_{f_{ue}} = a(\sigma_{z,0}^{b} + \sigma_{z,1}^{b}) + \sum_{j \in A} u_j \sigma_{z,0}^{j} + \sum_{\{i,j\} \in B} e_{ij} \sigma_{z,0}^{i} \sigma_{z,0}^{j} \tag{5.8}$$

where $b \in [\![0, n-1]\!]$.

For any $\varepsilon \in Q^n : H_{f_{ue}} |\varepsilon\rangle = f_{ue}(\varepsilon) |\varepsilon\rangle$, then $H_{f_{ue}}$ is diagonal in the computational basis.

Similar constructions for some specific combinatorial graph problems can be seen in [24, 25, 26, 66, 67].

The construction of $H_{f_{ue}}$ is more efficient than the construction of H_f. H_f is an addition of 2^n projections, while $H_{f_{ue}}$ is a sum of $\mathrm{card}(A) + \mathrm{card}(B) + 1$ Pauli operator products.

The Hamiltonian problem $H_{f_{ue}}$ can be implemented by the well known Ising model in QM (see [10, 48]).

In the following we consider the construction and structure of two initial Hamiltonians for AQO, the first one is based on the Hadamard transform and the last one is based on the σ_x Pauli operator.

5.2.1 Hadamard transform

In the following we describe the initial Hamiltonian for AQO based on the Hadamard transform similar to the proposed in [80], and we show its matrix structure, i.e. its construction from the computational point of view.

Let us recall that the Hadamard transform is the unitary map $W : \mathbb{H}_1 \to \mathbb{H}_1$ whose matrix, relative to the canonical basis is

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Let $W^{\otimes n} : \mathbb{H}_n \to \mathbb{H}_n$ be the n-fold tensor product of W. For $n \geq 1$,

$$W^{\otimes n} = (w_{ijn})_{0 \leq i,j \leq 2^n - 1} \tag{5.9}$$

such that $\forall i, j \in [\![0, 2^n - 1]\!] : w_{ijn} = \frac{1}{2^{\frac{n}{2}}} (-1)^{i \cdot j}$ where $i \cdot j$ is the bitwise dot product of the binary representations of the numbers i and j.

Remark 16. *The set of states* $(W^{\otimes n}|\varepsilon\rangle)_{\varepsilon\in Q^n}$ *form an orthonormal basis for* $\mathbb{H}_n$*, also called the Hadamard basis.*

Thus, any vector $\mathbf{x}\in\mathbb{H}_n$ can be written as $\mathbf{x}=\sum_{\varepsilon\in Q^n}x_\varepsilon W^{\otimes n}|\varepsilon\rangle$ where $\forall\varepsilon\in Q^n: x_\varepsilon\in\mathbb{C}$. Observe that $(W^{\otimes n}|\varepsilon\rangle)^H x=x_\varepsilon$ for all $\varepsilon\in Q^n$ and therefore

$$\begin{aligned}\left(\sum_{\varepsilon\in Q^n}(W^{\otimes n}|\varepsilon\rangle)(W^{\otimes n}|\varepsilon\rangle)^H\right)\mathbf{x} &= \sum_{\varepsilon\in Q^n}(W^{\otimes n}|\varepsilon\rangle)(W^{\otimes n}|\varepsilon\rangle)^H\mathbf{x}\\ &= \sum_{\varepsilon\in Q^n}x_\varepsilon W^{\otimes n}|\varepsilon\rangle\\ &= \mathbf{x},\end{aligned}$$

it follows that

$$\sum_{\varepsilon\in Q^n}(W^{\otimes n}|\varepsilon\rangle)(W^{\otimes n}|\varepsilon\rangle)^H=I, \tag{5.10}$$

this equation is known as the *completeness relation* [61].

Let $h:Q^n\to\mathbb{R}^+$ be a map such that $h(0^n)=0$ and $h(\varepsilon)\geq 1$ for all $\varepsilon\in Q^n-\{0^n\}$.

Let $H_h:\mathbb{H}_n\to\mathbb{H}_n$ be defined as follows:

$$\begin{aligned}H_h &= W^{\otimes n}\left(\sum_{\varepsilon\in Q^n}h(\varepsilon)\ |\varepsilon\rangle\langle\varepsilon|\right)(W^{\otimes n})^H\\ &= \sum_{\varepsilon\in Q^n}h(\varepsilon)\ \left(W^{\otimes n}|\varepsilon\rangle\right)\ \left(W^{\otimes n}|\varepsilon\rangle\right)^H.\end{aligned} \tag{5.11}$$

The ground state of H_h is given by $x_0=W^{\otimes n}|0^n\rangle=\frac{1}{2^{\frac{n}{2}}}\sum_{\varepsilon\in Q^n}|\varepsilon\rangle$. Then,

$$H_h(x_0)=\sum_{\varepsilon\in Q^n}h(\varepsilon)\ \left(W^{\otimes n}|\varepsilon\rangle\right)\ \left(W^{\otimes n}|\varepsilon\rangle\right)^H x_0=0.$$

i.e., x_0 is an eigenvector corresponding to the eigenvalue 0.

Now, let us describe explicitly the construction of the matrix H_h. Consider the correspondence from Q^n to $[\![0,2^n-1]\!]$ by the map $\varepsilon\mapsto\sum_{\nu=0}^{n-1}\varepsilon_\nu 2^\nu$. For each $j\in[\![0,2^n-1]\!]$, $W^{\otimes n}|j\rangle$ corresponds to the j-th column of $W^{\otimes n}$, and $\forall i,j\in[\![0,2^n-1]\!]$:

$$\left(W^{\otimes n}|i\rangle\right)\ \left(W^{\otimes n}|j\rangle\right)^H=(v_{kl})_{0\leq k,l\leq 2^n-1} \tag{5.12}$$

such that $v_{kl}=\frac{1}{2^{\frac{n}{2}}}(-1)^{k\cdot i+l\cdot j}$ and if $i=j$ then $v_{kl}=\frac{1}{2^{\frac{n}{2}}}(-1)^{(k\oplus l)\cdot i}$.

H_h can be rewritten as follows:

$$H_h=(u_{kl})_{0\leq k,l\leq 2^n-1}$$

such that

$$\begin{aligned}u_{kl} &= \frac{1}{2^n}\sum_{i=0}^{2^n-1}h(i)(-1)^{(k\oplus l)\cdot i}\\ &= \frac{1}{2^n}\left[\ (-1)^{(k\oplus l)\cdot 0}\ \cdots\ (-1)^{(k\oplus l)\cdot(2^n-1)}\ \right]\left[\ h(0)\ \cdots\ h(2^n-1)\ \right]^T\end{aligned} \tag{5.13}$$

and since $k \oplus l = l \oplus k$, then H_h is a symmetric matrix.

For every $j \in [\![0, 2^n-1]\!]$, let $W_j^{\otimes n} = (w_{klj})_{0\leq k,l\leq 2^n-1}$ such that $w_{klj} = 2^{-\frac{n}{2}}(-1)^{(k\oplus j)\cdot l}$. From equation (5.13), every column of H_h can be expressed as $2^{-\frac{n}{2}} W_j^{\otimes n}\mathbf{h}$ where $\mathbf{h} = \left[\begin{array}{ccc} h(0) & \cdots & h(2^n-1) \end{array}\right]^T$.

Proposition 5. *For every $j \in [\![0, 2^n-1]\!]$: $P_j = W_j^{\otimes n} W^{\otimes n}$ is a permutation matrix.*

Proof. A permutation matrix is a square matrix which has exactly one 1 in every row and every column, and the other elements are zeros. Now, by definition P_j is a square matrix, and let $W_j^{\otimes n} W^{\otimes n} = (w_{pq})_{0\leq p,q\leq 2^n-1}$ where $w_{pq} = \frac{1}{2^n}\sum_{l=0}^{2^n-1}(-1)^{(j\oplus p\oplus q)\cdot l}$, and $w_{pq} = 1$ if $(j \oplus p \oplus q) = 0$ and $w_{pq} = 0$ otherwise. For fixed $j, p \in [\![0, 2^n-1]\!]$, there is a unique $q \in [\![0, 2^n-1]\!]$ such that $j \oplus p \oplus q = 0$, then the proposition follows. □

From proposition 5 it follows that:

Remark 17. *For every $j \in [\![0, 2^n-1]\!]$, the j-th column of H_h can be expressed as $2^{-\frac{n}{2}} P_j W^{\otimes n}\mathbf{h}$.*

Thus, from remark 17 every column of H_h can be constructed by permuting the elements of the column vector $2^{-\frac{n}{2}} W^{\otimes n}\mathbf{h}$.

5.2.2 σ_x transform

Let us consider the Pauli transform $\sigma_x : \mathbb{H}_1 \to \mathbb{H}_1$ whose matrix with respect to the canonical basis is

$$\sigma_x = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right]. \tag{5.14}$$

σ_x has eigenvalues $+1, -1$ with respective eigenvectors $c_0 = W|0\rangle$ and $c_1 = W|1\rangle$.

For every $\varepsilon \in Q^n$, let

$$c_\varepsilon = \bigotimes_{j=0}^{n-1} c_{\varepsilon_j},$$

$(c_\varepsilon)_{\varepsilon\in Q^n}$ is the Hadamard basis of $\mathbb{H}_n$.

For any index $j \in [\![0, n-1]\!]$ and $\delta \in \{0, 1\}$ let $\sigma_{x,\delta,j} = \bigotimes_{\nu=0}^{n-1} \tau_{\nu,\delta} : \mathbb{H}_n \to \mathbb{H}_n$, where $\tau_{\nu,\delta} = \frac{1}{2}(I + (-1)^\delta \sigma_x)$ if $\nu = j$ and $\tau_{\nu,\delta} = \mathrm{Id}$ otherwise. Notice that $\tau_{\nu,\delta}$ can be written as $\tau_{\nu,\delta} = c_\delta c_\delta^H$.

For any $\varepsilon \in Q^n, \delta \in \{0, 1\}$ and $j \in [\![0, n-1]\!]$:

$$\begin{aligned} \sigma_{x,\delta,j}(c_\varepsilon) &= (I \otimes \cdots \otimes c_\delta c_\delta^H \otimes \cdots \otimes I)(c_{\varepsilon_0} \otimes \cdots \otimes c_{\varepsilon_j} \otimes \cdots \otimes c_{\varepsilon_{n-1}}) \\ &= c_{\varepsilon_0} \otimes \cdots \otimes c_\delta c_\delta^H c_{\varepsilon_j} \otimes \cdots \otimes c_{\varepsilon_{n-1}} \\ &= c_{\varepsilon_0} \otimes \cdots \otimes (c_\delta, c_{\varepsilon_j}) c_\delta \otimes \cdots \otimes c_{\varepsilon_{n-1}}, \end{aligned} \tag{5.15}$$

where $(c_\delta, c_{\varepsilon_j})$ is the inner product of c_δ and c_{ε_j}. Since that $\{c_0, c_1\}$ form a basis for $\mathbb{H}_1$, $(c_\delta, c_{\varepsilon_j}) = 1$ if $\delta = \varepsilon_j$ and $(c_\delta, c_{\varepsilon_j}) = 0$ otherwise.

From equation (5.15) it is satisfied that

$$\sigma_{x,\delta,j}(c_\varepsilon) = \neg(\delta \oplus \varepsilon_j)c_\varepsilon. \tag{5.16}$$

Now, let $\Delta : [\![0, n-1]\!] \to \mathbb{R}$ be a weighting map. For any $\delta \in \{0,1\}$ let us define the operator:

$$H_x : \mathbb{H}_n \to \mathbb{H}_n \quad , \quad H_x = \sum_{j=0}^{n-1} \Delta(j)\sigma_{x,\delta,j}. \tag{5.17}$$

From (5.16) it is satisfied that:

$$\forall \varepsilon \in Q^n : \quad H_x(c_\varepsilon) = \left(\sum_{j=0}^{n-1} \neg(\delta \oplus \varepsilon_j)\Delta(j) \right) c_\varepsilon. \tag{5.18}$$

The ground state of H_x is the state $x_0 = \frac{1}{2^{\frac{n}{2}}} \sum_{\varepsilon \in Q^n} |\varepsilon\rangle$ with corresponding eigenvalue equal to 0.

The Hamiltonians in equations (5.11) and (5.17) are both diagonal in the Hadamard basis and can be written as follows:

$$H_h = W^{\otimes n} D_h W^{\otimes n}, \qquad H_x = W^{\otimes n} D_x W^{\otimes n}$$

where D_h and D_x are diagonal matrices.

From equations (5.11) and (5.18), $\mathrm{diag}(D_h) = (h(\varepsilon))_{\varepsilon \in Q^n}$ and $\mathrm{diag}(D_x) = (\eta(\varepsilon))_{\varepsilon \in Q^n}$ such that for any $\delta \in \{0,1\}, \forall \varepsilon \in Q^n : \eta(\varepsilon) = \sum_{j=0}^{n-1} \neg(\delta \oplus \varepsilon_j)\Delta(j)$.

Remark 18. *The Hamiltonian given in equation (5.17) can be expressed as in equation (5.11). The converse is not always true.*

5.3 k-local Hamiltonian problems

For each $n \in \mathbb{N}$ let Q^n be the set of n-length words over Q and let $Q^* = \bigcup_{n \geq 0} Q^n$ be the *dictionary*, i.e. the set of finite length words, of Q.

A *promise problem* consists of a partition $\{Y, N\}$ of Q^*. For any word instance $\sigma \in Q^*$ the corresponding solution is a decision whether $\sigma \in Y$ (σ is a *Yes-instance*) or $\sigma \in N$ (σ is a *No-instance*).

For each $n \in \mathbb{N}$, $B_n = (|\sigma\rangle)_{\sigma \in Q^n} \subset S_n$ is the canonical basis of $\mathbb{H}_n$. Let $B_* = \bigcup_{n \geq 0} B_n$.

A *verifier* is a map of the form $V : B_* \times B_* \to Q$. If $V(|\sigma\rangle, |\tau\rangle) = 1$ then it is said that the verifier *accepts* σ as a Yes-instance with *proof*, or *certificate*, τ.

Let $\varepsilon : \mathbb{N} \to [0,1]$ be such that

$$\forall \sigma \in Q^* : \quad 2^{-\Omega(|\sigma|)} \leq \varepsilon(|\sigma|) \leq \frac{1}{3}. \tag{5.19}$$

The class QMA_ε consists of those promise problems $\{Y, N\}$ such that there is a quantum polynomial time verifier V satisfying:

- $\forall \sigma \in Y \ \exists \tau \in Q^*$: $\Pr\left(V(|\sigma\rangle, |\tau\rangle) = 1\right) \geq 1 - \varepsilon(|\sigma|)$.
- $\forall \sigma \in N \ \forall \tau \in Q^*$: $\Pr\left(V(|\sigma\rangle, |\tau\rangle) = 1\right) \leq \varepsilon(|\sigma|)$.

Remark 19 ([52]). *If* $\varepsilon_0, \varepsilon_1 : \mathbb{N} \to [0,1]$ *satisfy condition (5.19) then* $\mathrm{QMA}_{\varepsilon_0} = \mathrm{QMA}_{\varepsilon_1}$.

The common class resulting from remark 19 is QMA.

Let $k \leq n$. Let $K \subset [\![0, n-1]\!]$ be an index set of cardinality k. Let

$$B_K = \{\bigotimes_{j=0}^{n-1} \mathbf{b}_j |\ \mathbf{b}_j = |s_j\rangle \text{ with } s_j \in Q \text{ if } j \in K,\ \mathbf{b}_j = |0\rangle \text{ otherwise } \}$$

be the collection of basic vectors in $\mathbb{H}_n$ whose "non-horizontal tensor factors" appear just at indexes in K. Let $V_K = \mathcal{L}(B_K)$ be the space spanned by B_K. Then V_K is isomorphic to $\mathbb{H}_k$ and there is a complementary space V'_K isomorphic to $\mathbb{H}_{n-k}$ such that $\mathbb{H}_n = V_K \otimes V'_K$.

Let $H : \mathbb{H}_n \to \mathbb{H}_n$ be a Hamiltonian operator. It is said that H *acts on* k*-qubits* if there is an index set $K \subset [\![0, n-1]\!]$ of cardinality k such that there is a Hamiltonian operator $H_K : V_K \to V_K$ with

$$H = H_K \otimes \mathrm{Id}_{2^{n-k}} \tag{5.20}$$

where $\mathrm{Id}_{2^{n-k}}$ is the identity map in the complementary space V'_K.

A Hamiltonian $H : \mathbb{H}_n \to \mathbb{H}_n$ is k*-local* if it can be expressed as the addition of Hamiltonian operators, each acting on k-qubits, with the additional conditions stated below:

1. $H = \sum_{j \in J} H_j$, with $\mathrm{card}\,(J) = n^{O(1)}$.
2. $\forall j \in J$: $\|H_j\| \leq n^{O(1)}$.

The second condition is equivalent to state that $\forall j$, both H_j and $\mathrm{Id}_n - H_j$ are non-negative.

For any Hamiltonian $H : \mathbb{H}_n \to \mathbb{H}_n$ let us denote by $\lambda_0(H)$ the smallest, in absolute value, eigenvalue of H.

Instance: k-local Hamiltonian
Solution: A k-local Hamiltonian $H : \mathbb{H}_n \to \mathbb{H}_n$ and two real numbers a, b such that $b - a \geq n^{-O(1)}$ and either $\lambda_0(H) \leq a$ or $\lambda_0(H) \geq b$. 1 if H has an eigenvalue below a and 0 if all eigenvalues of H are at least b.

k-local Hamiltonian is NP-hard for $k \geq 2$ [83]. 5*-local Hamiltonian*, 3*-local Hamiltonian* and 2*-local Hamiltonian* were proved QMA-complete, respectively, in [52], in [51], and in [50].

5.3.1 Reduction of graph problems to the 2-local Hamiltonian problem

Let $\mathbf{G} = (\llbracket 0, n-1 \rrbracket, E)$ be a graph. Let us assume that there is a quadratic Boolean map $f : Q^n \to \mathbb{R}$,

$$\varepsilon \mapsto f(\varepsilon) = \sum_{ij \in E} \left[a_3 \varepsilon_i \varepsilon_j + a_2 (1-\varepsilon_i)\varepsilon_j + a_1 \varepsilon_i (1-\varepsilon_j) + a_0 (1-\varepsilon_i)(1-\varepsilon_j) \right] \tag{5.21}$$

that should be minimized. The map f is determined by the edges at the graph $\mathbf{G}$. Equivalently, the map f can be expressed as

$$\forall \varepsilon \in Q^n : \ f(\varepsilon) = \sum_{ij \in E} \left[c_0 + c_1 \varepsilon_i + c_2 \varepsilon_j + c_3 \varepsilon_i \varepsilon_j \right] \tag{5.22}$$

with

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{5.23}$$

Let us consider the projection

$$\pi_2 = a_0 \left|00\right\rangle \left\langle 00\right| + a_1 \left|01\right\rangle \left\langle 01\right| + a_2 \left|10\right\rangle \left\langle 10\right| + a_3 \left|11\right\rangle \left\langle 11\right| : \mathbb{H}_2 \to \mathbb{H}_2, \tag{5.24}$$

represented by the matrix $\pi_2 = \mathrm{diag}[a_0 \ a_1 \ a_2 \ a_3]$, hence its eigenvalues are a_0, a_1, a_2, a_3 with corresponding eigenspaces $\mathcal{L}(|00\rangle)$, $\mathcal{L}(|01\rangle)$, $\mathcal{L}(|10\rangle)$, $\mathcal{L}(|11\rangle)$ respectively. Namely, for the basis vector $|\varepsilon_0 \varepsilon_1\rangle \in B_2$ we have

$$\pi_2 \left|\varepsilon_0 \varepsilon_1\right\rangle = a_{(\varepsilon_0 \varepsilon_1)_2} \left|\varepsilon_0 \varepsilon_1\right\rangle . \tag{5.25}$$

For any index pair ij let $\pi_2[ij]$ be the Hamiltonian defined by (5.20), with $K = \{i, j\}$ and $H_K = \pi_2$. The eigenvalues of each $\pi_2[ij]$ are the coefficients a_i and after (5.25)

$$\forall \varepsilon = (\varepsilon_0, \ldots, \varepsilon_{n-1}) \in Q^n : \ \pi_2[ij]\varepsilon = a_{(\varepsilon_i \varepsilon_j)_2} \varepsilon. \tag{5.26}$$

Let

$$H = \sum_{ij \in E} \pi_2[ij]. \tag{5.27}$$

Clearly $\|H\| = \max_i |a_i|$, thus H is a 2-local Hamiltonian if $|a_i| \leq 1$. From (5.26)

$$\forall \varepsilon = (\varepsilon_0, \ldots, \varepsilon_{n-1}) \in Q^n : \ H\varepsilon = \left(\sum_{ij \in E} a_{(\varepsilon_i \varepsilon_j)_2} \right) \varepsilon. \tag{5.28}$$

Thus, the ground states correspond to the eigenvalue

$$\lambda_0(H) = \min_{\varepsilon} \left| \sum_{ij \in E} a_{(\varepsilon_i \varepsilon_j)_2} \right| . \tag{5.29}$$

2-local Hamiltonian would then provide a solution of the minimization problem of the quadratic Boolean map.

The above technique is just a generalization of the reduction of *Max Cut* and *Independent Set* to 2-local Hamiltonian presented at [83].

A *cut* in a weighted graph $\mathbf{G} = (\llbracket 0, n-1 \rrbracket, E, w)$, where $w : E \to \mathbb{R}^+$ is a *weighting map*, is a partition $C = \{V_0, V_1\}$ of the vertex set $\llbracket 0, n-1 \rrbracket$. For $\delta, \varepsilon \in Q$ let $E_{\delta\varepsilon} = \{ij \in E |\ i \in V_\delta\ ,\ j \in V_\varepsilon\}$ be the collection of edges with an extreme in V_δ and the other in V_ε. The *weight of the cut* is $w(C) = \sum_{\delta \neq \varepsilon, ij \in E_{\delta\varepsilon}} w(ij)$.

Max Cut
Instance: A weighted graph $\mathbf{G} = (\llbracket 0, n-1 \rrbracket, E, w)$ and a threshold $w_0 \in \mathbb{R}^+$.
Solution: A decision about whether there exists a cut C such that $w(C) \geq w_0$.

Simple Max Cut is the restriction of *Max Cut* to weighted graphs with constant weights 1. Both *Max Cut* and *Simple Max Cut* are NP-complete.

Let us consider $\mathbf{G} = (\llbracket 0, n-1 \rrbracket, E, 1)$ as a weighted graph with unitary weights. Let $X = (X_j)_{j=0}^{n-1}$ be a collection of Boolean variables. For any assignment $\varepsilon = (\varepsilon_j)_{j=0}^{n-1} \in Q^n$ and each $\delta \in Q$, let $V_\delta = \{j |\ \varepsilon_j = \delta\}$. This determines a correspondence among assignments and cuts. For any cut $C = \{V_0, V_1\}$ we have

$$w(C) = \mathrm{card}\,(E_{01}) + \mathrm{card}\,(E_{10}) = \sum_{ij \in E} \left[(1-\varepsilon_i)\varepsilon_j + \varepsilon_i(1-\varepsilon_j)\right].$$

Thus, given a threshold $w_0 \in \mathbb{Z}^+$ we have

$$w(C) \geq w_0 \iff \sum_{ij \in E} \left[\varepsilon_i\varepsilon_j + (1-\varepsilon_i)(1-\varepsilon_j)\right] \leq \mathrm{card}\,(E) - w_0.$$

Hence *Max Cut* can be stated as an optimization problem for the quadratic Boolean map

$$Q^n \to \mathbb{R}\ ,\ \varepsilon \mapsto \sum_{ij \in E} \left[\varepsilon_i\varepsilon_j + (1-\varepsilon_i)(1-\varepsilon_j)\right],$$

which is indeed of the form (5.21) with $a_0 = a_3 = 1$ and $a_1 = a_2 = 0$.

On the other side, a vertex set $V \subset \llbracket 0, n-1 \rrbracket$ is *independent* if $E[V] = \emptyset$, i.e. no edge exists among two points in V. The following is a well known NP-complete problem:

Independent Set
Instance: A graph $\mathbf{G} = (\llbracket 0, n-1 \rrbracket, E)$ and a threshold $w_0 \in \mathbb{Z}^+$.
Solution: A decision about whether there exists an independent set V such that $\mathrm{card}\,(V) \geq w_0$.

For any instance graph let us consider the quadratic Boolean map

$$f : Q^n \to \mathbb{R}\ ,\ \varepsilon \mapsto f(\varepsilon) = \sum_{i=0}^{n-1} \varepsilon_i - \sum_{ij \in E} \varepsilon_i\varepsilon_j.$$

Thus, given a threshold $w_0 \in \mathbb{Z}^+$ we have

$$f(\varepsilon) \geq w_0 \iff g(\varepsilon) = n - f(\varepsilon) = \sum_{i=0}^{n-1} [1 - \varepsilon_i] + \sum_{ij \in E} \varepsilon_i \varepsilon_j . \leq n - w_0.$$

The second sum is a quadratic of the form (5.21) with $a_0 = a_1 = a_2 = 0$ and $a_3 = 1$. For the first sum, let us consider the projections $\rho_1 = |0\rangle \langle 0| : \mathbb{H}_1 \to \mathbb{H}_1$ and for the second $\pi_2 = |11\rangle \langle 11| : \mathbb{H}_2 \to \mathbb{H}_2$. Let

$$H = \sum_{i=0}^{n-1} \rho_1[i] + \sum_{ij \in E} \pi_2[ij]. \tag{5.30}$$

By choosing $a = n - w_0 + \frac{1}{2}$ and $b = a + \frac{1}{4}$ a solution of 2-*local Hamiltonian* gives a solution of *Independent Set*.

5.4 Graph structures and optimization problems

In *Descriptive Complexity* the class of problems defined in *Computational Complexity* are characterized by expressions in first and second order logic. There are many results in this area such as the Fagin's Theorem that asserts that the class of problems NP is equal to the set of Boolean queries in existential second order logic [47, 57].

On the other hand, Monadic second order logic has play an important role to express many NP-complete problems, for instance the graph coloring problem. In [28] shows that any problem expressible in monadic second order logic can be stated as an optimization problem (see also [64, 27, 47]).

Monadic second order logic expressions provide a syntactical representation of many optimization problems. This representation is in correspondence with a Boolean algebra. Following [21], we consider the correspondence between second order logic expressions and Boolean formulas, in order to define a corresponding pseudo-Boolean map. This scheme can be applied to any optimization problem expressible in monadic second order logic, and subsequently to be solved by AQO.

The necessary terminology and definitions on first and second order logic are given.

5.4.1 Relational signatures

A *signature* $\Sigma = (\Phi, \Pi)$ consists of a set Φ of function symbols and positive integers $(\rho(f))_{f \in \Phi}$, and of a set of relation symbols Π and positive integers $(\rho(R))_{R \in \Pi}$. The numbers $\rho(f)$ and $\rho(R)$ assert that f is a function of $\rho(f)$ variables and R is a $\rho(R)$-ary relation.

A signature without relation symbols is called an *algebraic signature* and a signature without function symbols is called a *relational signature*.

Let $\mathcal{R}$ be a relational signature, let $\mathcal{R}_0 := \{R \in \mathcal{R} | \rho(R) = 0\}$ be the set of relation symbols of arity zero called *constant symbols*, let $\mathcal{R}_i := \{R \in \mathcal{R} | \rho(R) = i\}$ be the

set of relation symbols of arity i and let $\mathcal{R}_+ := \bigcup\{\mathcal{R}_i | i \geq 1\}$ be the set of relation symbols of positive arity.

A $\mathcal{R}$-*structure* is a tuple $S = \langle D_S, (R_S)_{R\in\mathcal{R}_+}, (c_S)_{c\in\mathcal{R}_0}\rangle$ where D_S is a finite set called the *domain* of S, for each $R \in \mathcal{R}_+, R_S \subseteq D_S^{\rho(R)}$ is called the *interpretation* of R, and for each $c \in \mathcal{R}_0, c_S \in D_S$ is called the *interpretation* of c.

5.4.2 First order logic

Let $\mathcal{V}_0$ be a countable set of variables and let $\mathcal{R}$ be a relational signature. A *term* is either a variable in $\mathcal{V}_0$ or a constant symbol in $\mathcal{R}_0$. An *atomic formula* s is either $s = t$ or $s = R(t_1, \ldots, t_{\rho(R)})$ where $R \in \mathcal{R}_+$ and $t, t_1, \ldots, t_{\rho(R)}$ are terms. If ϕ and ψ are atomic formulas, then ϕ, $\neg\phi$, $(\phi\wedge\psi)$, $(\phi\vee\psi)$, $(\phi \Rightarrow \psi)$ and $(\phi \Leftrightarrow \psi)$ are *first-order formulas* over $\mathcal{R}$. If φ is a first-order formula and $x \in \mathcal{V}_0$, then $\exists x\varphi$ and $\forall x\varphi$ are first-order formulas as well.

It is said that a variable $x \in \mathcal{V}_0$ is *free* in a formula ϕ, if it is not inside a $\exists x$ or $\forall x$ quantifier; otherwise, it is *bound*. A formula without free variables over a relational signature $\mathcal{R}$ is called *closed*.

The set of all first-order formulas over a relational signature $\mathcal{R}$ with free variables in $\mathcal{X} \subseteq \mathcal{V}_0$ is denoted as $\mathrm{FO}(\mathcal{R}, \mathcal{X})$. A formula $\varphi \in \mathrm{FO}(\mathcal{R}, \{x_1, \ldots, x_n\})$ will be written as $\varphi(x_1, \ldots, x_n)$ to specify its free variables. The set of all $\mathcal{R}$-structures will be denoted as $\mathrm{STR}(\mathcal{R})$. For a $S \in \mathrm{STR}(\mathcal{R})$ given as $S = \langle D_S, (R_S)_{R\in\mathcal{R}_+}, (c_S)_{c\in\mathcal{R}_0}\rangle$, if $\varphi(x_1, \ldots, x_n) \in \mathrm{FO}(\mathcal{R}, \{x_1, \ldots, x_n\})$ and $d_1, \ldots, d_n \in D_S$, then $S \models \varphi(d_1, \ldots, d_n)$ denotes that φ is true in S when $x_i = d_i$ for $i = 1, \ldots, n$.

5.4.3 Second order logic

Let $\mathcal{V}_\omega$ be a countable set consisting of first-order variables and relation variables denoted by upper-case letters $X_1, \ldots, X_m$. Each relation variable $X \in \mathcal{V}_\omega$ has arity $\rho(X)$, and there are countably many relation variables of each arity.

The *second-order formulas* over a relational signature $\mathcal{R}$ are defined as follows: For any $R \in \mathcal{R}$, $R(t_1, \ldots, t_{\rho(R)})$ is an atomic formula where $t_1, \ldots, t_{\rho(R)}$ are terms, and for any $X \in \mathcal{V}_\omega$, $X(t_1, \ldots, t_{\rho(X)})$ is also an atomic formula where $t_1, \ldots, t_{\rho(X)}$ are terms. Then, the second-order formulas are constructed from the atomic formulas together with the propositional connectives and quantifications over first-order and relational variables.

The set of all second-order formulas over a relational signature $\mathcal{R}$ with free variables in $\mathcal{X} \subseteq \mathcal{V}_\omega$ is denoted as $\mathrm{SO}(\mathcal{R}, \mathcal{X})$. In the following we will represent a formula $\varphi \in \mathrm{SO}(\mathcal{R}, \{X_1, \ldots, X_m, x_1, \ldots, x_n\})$ as $\varphi(X_1, \ldots, X_m, x_1, \ldots, x_n)$ to specify its free variables. For a structure $S \in \mathrm{STR}(\mathcal{R})$ given by $S = \langle D_S, (R_S)_{R\in\mathcal{R}_+}, (c_S)_{c\in\mathcal{R}_0}\rangle$, for a formula $\varphi \in \mathrm{SO}(\mathcal{R}, \{X_1, \ldots, X_m, x_1, \ldots, x_n\})$, $E_1 \subseteq D_S^{\rho(X_1)}, \ldots, E_m \subseteq D_S^{\rho(X_m)}$ and $d_1, \ldots, d_n \in D_S$, $S \models \varphi(E_1, \ldots, E_m, d_1, \ldots, d_n)$ denotes that φ is true in S for the values $E_1, \ldots, E_m$ of $X_1, \ldots, X_m$ and $d_1, \ldots, d_n$ of $x_1, \ldots, x_n$.

Let $\mathcal{V}_1$ be a set of the countable set $\mathcal{V}_0$ of first-order variables and the countably many relation variables of arity one from $\mathcal{V}_\omega$. A *monadic second-order formula* is a second-order formula written with variables from $\mathcal{V}_1$. The set of all monadic second-order formulas with free variables in $\mathcal{X} \subseteq \mathcal{V}_1$, over a relational signature $\mathcal{R}$ is denoted by $\mathrm{MSOL}(\mathcal{R}, \mathcal{X})$.

Note that, any relation variable can be interpreted as a set. Thus, any relation variable will be called a *set variable*.

5.4.4 Monadic second-order logic decision and optimization problems

The following definitions are from [28]:

Definition 10. *A decision problem is an MSOL($\mathcal{R}$) decision problem over STR($\mathcal{R}$), if it can be expressed in the following form: Given an $\mathcal{R}$-structure $S \in STR(\mathcal{R})$, does $S \models \varphi$ hold? where φ is a closed MSOL($\mathcal{R}$) formula.*

Example 1: Let $\mathbf{G} = (V, E)$ be graph and let $\mathcal{R}_s := \{edg\}$ be a relational signature with $\rho(edg) = 2$. An $\mathcal{R}_s$-structure for $\mathbf{G}$ is defined by $\lfloor \mathbf{G} \rfloor := \langle V_{\mathbf{G}}, edg_{\mathbf{G}} \rangle$ where $edg_{\mathbf{G}} \subseteq V_{\mathbf{G}}^{[2]}$ such that $\forall u, v \in V : [\{u, v\} \in edg_{\mathbf{G}} \Leftrightarrow \{u, v\} \in E]$ and $V_{\mathbf{G}}$ is the vertex set of $\mathbf{G}$.

The 3-colorability problem is an $\mathrm{MSOL}(\mathcal{R}_s)$ decision problem since it can be stated as follows: Let $\mathbf{G}$ be a graph and let $\lfloor \mathbf{G} \rfloor$ be a structure for $\mathbf{G}$, does $\lfloor \mathbf{G} \rfloor \models \gamma_3$? where γ_3 is the closed $\mathrm{MSOL}(\mathcal{R}_s)$ formula defined as

$$\begin{aligned} \gamma_3 \;=\; & \exists X_1, X_2, X_3 \Big(Part(X_1, X_2, X_3) \wedge \\ & \forall u, v \big(edg_{\mathbf{G}}(u, v) \wedge u \neq v \Rightarrow \neg(X_1(u) \wedge X_1(v)) \wedge \\ & \neg(X_2(u) \wedge X_2(v)) \wedge \neg(X_3(u) \wedge X_3(v)) \big) \Big) \end{aligned} \tag{5.31}$$

and $Part(X_1, X_2, X_3)$ is defined as

$$\begin{aligned} Part(X_1, X_2, X_3) \;=\; & \forall v \Big(\big(X_1(v) \vee X_2(v) \vee X_3(v) \big) \wedge \big(\neg(X_1(v) \wedge X_2(v)) \wedge \\ & \neg(X_2(v) \wedge X_3(v)) \wedge \neg(X_1(v) \wedge X_3(v)) \big) \Big). \end{aligned}$$

Thus $\lfloor \mathbf{G} \rfloor \models \gamma_3$ if and only if $\mathbf{G}$ is 3-colorable.

Definition 11. *An optimization problem P is said to be a LinEMSOL($\mathcal{R}$) optimization problem over STR($\mathcal{R}$), if it can be expressed in the following form: Given $S \in STR(\mathcal{R})$ and m evaluation functions $f_1, \ldots, f_m$ associating values to the elements of S, find relations $E_1, \ldots, E_l \subseteq Dom(S)$ for the free variables of the formula $\varphi(X_1, \ldots, X_l) \in MSOL(\mathcal{R}, \{X_1, \ldots, X_l\})$ such that:*

$$\sum_{\substack{1 \leq i \leq l \\ 1 \leq j \leq m}} a_{ij} E[X_i]_j = opt \Big\{ \sum_{\substack{1 \leq i \leq l \\ 1 \leq j \leq m}} a_{ij} E'[X_i]_j : S \models \varphi(E_1', \ldots, E_l') \Big\} \tag{5.32}$$

where $E[X_i]_j := \sum_{b \in E_i} f_j(b)$*, opt is either* min *or* max *and* $\{a_{ij} : 1 \leq i \leq l, 1 \leq j \leq m\}$ *is a set of* ml *integers.*

Example 2: Let $\mathbf{G} = (V, E, \delta)$ be a weighted graph where $\delta : V \to \mathbb{Z}$ is a map. For a subset $A \subseteq V$, the weight of A is defined as $w(A) := \sum_{a \in A} \delta(a)$. The *maximum weighted clique problem* (MWC) consists in finding a clique in $\mathbf{G}$ with maximum weight.

The MWC problem is an LinEMSOL($\mathcal{R}$) optimization problem since it can be expressed as follows: Given a weighted graph $\mathbf{G} = (V, E, \delta)$, a $\mathcal{R}_s$-structure $\lfloor \mathbf{G} \rfloor$ for $\mathbf{G}$ and an evaluating function $f_1 : V \to \mathbb{Z}$ with $a \mapsto \delta(a)$. Find an instance relation $V_1 \subseteq V_{\mathbf{G}}$ to the free variable X_1 in θ such that:

$$\sum_{a \in V_1} f_1(a) = \max \Big\{ \sum_{a \in V_1^{'}} f_1(a) : \lfloor \mathbf{G} \rfloor \models \theta(V_1^{'}) \Big\} \tag{5.33}$$

where

$$\theta(X_1) = \forall u, v((X_1(u) \wedge X_1(v) \wedge u \neq v) \Rightarrow edg_{\mathbf{G}}(u, v)). \tag{5.34}$$

Remark 20. *Every MSOL($\mathcal{R}$) decision problem can be expressed as a LinEMSOL($\mathcal{R}$) optimization problem.*

Note that, in order to optimize the objective function (5.33), every instance relation $V_1^{'} \subseteq V_{\mathbf{G}}$ must satisfies that $\lfloor \mathbf{G} \rfloor \models \theta(V_1')$. It is possible to define another objective function as in (5.33) without restrictions if a penalty function is added. Considers the following definition:

Let P be a LinEMSOL optimization problem as defined in (5.32), then there is an MSOL expression $\varphi(X_1, \ldots, X_l)$ over a structure S, and evaluation functions $f_1, \ldots, f_m$, and a set of integers $\{a_{ij} : 1 \leq i \leq l, 1 \leq j \leq m\}$. For any instance relations $E_1, \ldots, E_l \subseteq \mathrm{Dom}(S)$, let us define

$$\Gamma_P(E_1, \ldots, E_l) = \sum_{\substack{1 \leq i \leq l \\ 1 \leq j \leq m}} a_{ij} E[X_i]_j + g_{\varphi}(E_1, \ldots, E_l) \tag{5.35}$$

where g_{φ} is a penalty function such that $g_{\varphi}(E_1, \ldots, E_l)$ is equal to some constant c if and only if $S \models \varphi(E_1, \ldots, E_l)$, and $g_{\varphi}(E_1, \ldots, E_l) \gg c$ otherwise.

Remark 21. *Minimizing (5.32) is equivalent to minimizing (5.35). A similar result can be stated for maximization.*

5.4.5 MSOL optimization problems and pseudo-Boolean maps

A *pseudo-Boolean map* $f : Q^n \to \mathbb{R}^+$ on n Boolean variables is a non-negative real valued map on the hypercube Q^n. From theorem 2, f can be represented as:

$$f(X) = \sum_{S \in \mathcal{P}([\![0, n-1]\!])} c(S) \prod_{j \in S} x_j$$

for some map $c : \mathcal{P}([\![0, n-1]\!]) \to \mathbb{R}$.

The set of Boolean variables will be denoted by $X = \{x_i : 0 \leq i \leq n-1\}$ and the set of literals will be denoted by $L = \{x_i, \overline{x}_i : 0 \leq i \leq n-1\}$ where $\overline{x}_i := 1 - x_i$.

A *disjunctive form* (DF) is an expression of the form

$$\phi = \bigvee_{k=1}^{m} \left(\bigwedge_{i \in A_k} x_i \wedge \bigwedge_{j \in B_k} \overline{x}_j \right) \tag{5.36}$$

where $A_k \cap B_k = \emptyset$ for $k = 1, \ldots, m$.

A DF ϕ is said to be *orthogonal* if $(A_k \cap B_l) \cup (A_l \cap B_k) \neq \emptyset$ for all $k, l \in \{1, \ldots, m\}$ with $k \neq l$.

It is said that a DF ϕ *represents* a Boolean function g if the true valued points of g coincide with the true valued points of ϕ.

Theorem 3. *Every Boolean function $g : Q^n \to Q$ can be represented through an orthogonal DF ϕ_g.*

See a proof in [31].

Theorem 4. *Every Boolean function $g : Q^n \to Q$ represented by an orthogonal DF, has an associated multilinear polynomial given as*

$$g(X) = \sum_{k=1}^{m} \left(\prod_{i \in A_k} x_i \prod_{j \in B_k} (1 - x_j) \right).$$

See a proof in [31].

From theorem 4 it follows that every FO sentence has an associated multilinear polynomial. Let us consider the following algorithm to obtain a multilinear polynomial of a given FO sentence.

FO sentence into multilinear polynomial (1)
Input: A FO(Σ) sentence φ where $\Sigma = (\Phi, \Pi)$.
Output: A multilinear polynomial p_φ over a set of Boolean variables X.

1. Transform φ into a DF (See [75] for a standard procedure)
2. Apply the equivalences $x \wedge y = xy, x \vee y = x + y - xy$ and $\overline{x} = 1 - x$ on φ to obtain an arithmetic expression p_φ
3. For every atomic formula s in p_φ introduce a Boolean variable X_s to obtain a multilinear polynomial over $X = \{X_s | s \text{ is an atomic formula in } \varphi\}$

The step 1 in the algorithm 1 drop all existential quantifier in the sentence φ by Skolemization.

Example 3: Let $\mathbf{G} = (V, E)$ be a undirected graph, and let

$$\begin{aligned}\varphi &= \forall x(\neg edg(x,x)) \wedge \neg\exists w,x,y,z(edg(w,x) \wedge edg(x,y) \wedge edg(y,z) \\ &\quad \wedge \neg edg(w,y) \wedge \neg edg(w,z) \wedge \neg edg(x,z)).\end{aligned}$$

Then, it is satisfied that $\lfloor \mathbf{G} \rfloor \models \varphi$ if and only if $\mathbf{G}$ has no loops and no induced subgraph isomorphic to P_4 (P_4 is the graph $\bullet - \bullet - \bullet - \bullet$).

By transforming φ into a DNF we obtain

$$\begin{aligned}\varphi &= (\neg edg(u,u) \wedge \neg edg(w,x)) \vee (\neg edg(u,u) \wedge \neg edg(x,y)) \vee \\ &\quad (\neg edg(u,u) \wedge \neg edg(y,z)) \vee (\neg edg(u,u) \wedge edg(w,y)) \vee \\ &\quad (\neg edg(u,u) \wedge edg(w,z)) \vee (\neg edg(u,u) \wedge edg(x,z)),\end{aligned}$$

and after applying the steps 2 and 3 of the algorithm (1)

$$\begin{aligned}p_\varphi &= \left(\sum_{u\in V}(1 - X_{uu})\right) \sum_{w,x,y,z\in V} (1 - X_{wx}X_{xy}X_{yz} + X_{wx}X_{xy}X_{yz}X_{wy} + \\ &\quad X_{wx}X_{xy}X_{yz}X_{wz} - X_{wx}X_{xy}X_{yz}X_{xz} - X_{wx}X_{xy}X_{yz}X_{wy}X_{wz} - \\ &\quad X_{wx}X_{xy}X_{yz}X_{wy}X_{xz} - X_{wx}X_{xy}X_{yz}X_{wz}X_{xz} + X_{wx}X_{xy}X_{yz}X_{wy}X_{wz}X_{xz})\end{aligned}$$

where $\forall u, v \in V : X_{uv} := edg(u, v)$.

Finally, given p_φ as input to the procedure `Reduce` in section 5.1, it produces a quadratic pseudo-Boolean map.

Remark 22. *Given a undirected graph* $\mathbf{G} = (V, E)$, $p_\varphi(X) = (\text{card } V)^5$ *if and only if,* $\mathbf{G}$ *has no loops and no induced subgraph isomorphic to* P_4. $p_\varphi(X) < (\text{card } V)^5$ *otherwise.*

In the second order logic it is allowed to quantify over relations of any arity, then in order to obtain a polynomial expression of a given SO sentence ψ, we will write ψ in existential second-order logic form.

An sentence of *existential second-order logic* (ESOL) ψ over a signature $\Sigma = (\Phi, \Pi)$ is of the form

$$\psi = \exists R_1 \cdots \exists R_r \varphi,$$

where $R_1, \ldots, R_r$ are relational symbols of respective arities $\rho(R_1), \ldots, \rho(R_r)$ and φ is a first-order sentence over the signature $\Sigma' = (\Phi, \Pi \cup \{R_1, \ldots, R_r\})$. A structure $S \in \text{STR}(\Sigma)$ satisfies an ESOL sentence $\exists R_1 \cdots \exists R_r \varphi$, if there are relations $E_1 \subseteq D_S^{\rho(R_1)}, \ldots, E_r \subseteq D_S^{\rho(R_r)}$ such that S, augmented with $\{E_1, \ldots, E_r\}$ to comprise a structure for Σ', satisfies φ.

An ESOL sentence $\exists R_1 \cdots \exists R_r \varphi$ is an *existential monadic second-order logic* (EMSOL) sentence if the relations $R_1, \ldots, R_r$ are of arity one.

Theorem 5 (Fagin, 74)**.** *Every decision problem on finite graphs is in* NP *if and only if it is expressible in existential second-order logic.*

Examples of graph problems that can be written as an EMSOL expression are 3-colorability, Maximum Clique problem, Hamiltonian circuit problem, and TSP problem.

Let us consider the following algorithm to obtain a multilinear polynomial of a given EMSOL expression.

EMSOL sentence into multilinear polynomial (2)
Input: A SO(Σ) sentence ψ where $\Sigma = (\Phi, \Pi)$.
Output: A multilinear polynomial p_ψ over a set of Boolean variables X.

1. Transform ψ into an EMSOL sentence $\exists R_1, \ldots, R_r.\varphi$
2. Transform φ into a DF (See [75] for a standard procedure)
3. Apply the equivalences $x \wedge y = xy, x \vee y = x + y - xy$ and $\overline{x} = 1 - x$, on φ to obtain an arithmetic expression p_φ
4. For every atomic formula s in p_φ introduce a Boolean variable X_s to obtain a multilinear polynomial p_ψ over $X = \{X_s | s \text{ is an atomic formula in } \varphi\}$

Let us consider the following example:

Example 4: Let $\mathbf{G} = (V, E)$ be a simple undirected graph, and let

$$\begin{aligned} \gamma_3 &= \exists X, Y, Z \Big(Part(X, Y, Z) \wedge \\ &\quad \forall u, v \big(edg_{\mathbf{G}}(u, v) \Rightarrow \neg(X(u) \wedge X(v)) \wedge \\ &\quad \neg(Y(u) \wedge Y(v)) \wedge \neg(Z(u) \wedge Z(v)) \big) \Big) \end{aligned}$$

and $Part(X, Y, Z)$ is defined by

$$\begin{aligned} Part(X, Y, Z) &= \forall v \Big(\big(X(v) \vee Y(v) \vee Z(v) \big) \wedge \big(\neg(X(v) \wedge Y(v)) \wedge \\ &\quad \neg(Y(v) \wedge Z(v)) \wedge \neg(X(v) \wedge Z(v)) \big) \Big). \end{aligned}$$

Then, it is satisfied that $\lfloor \mathbf{G} \rfloor \models \gamma_3$ if and only if, $\mathbf{G}$ is 3-colorable.

It can be seen that γ_3 is already in EMSOL form, then we can apply the algorithm (2). We claim that there are polynomials p_{φ_1} and p_{φ_2} such that $p_\psi = p_{\varphi_1} \cdot p_{\varphi_2}$, where

$$\begin{aligned} p_{\varphi_1} &= \sum_{u \in V} (X_u + Y_u + Z_u - X_u Y_u - X_u Z_u - Y_u Z_u + X_u Y_u Z_u) \cdot \\ &\quad (1 - X_u Y_u)(1 - Y_u Z_u)(1 - X_u Z_u) \end{aligned} \tag{5.37}$$

and

$$p_{\varphi_2} = \sum_{u,v \in V} ((1 - X_{uv}) + (1 - X_u X_v)(1 - Y_u Y_v)(1 - Z_u Z_v) X_{uv}) \,. \tag{5.38}$$

For any non-empty sets $X, Y, Z \subseteq V$, $p_\psi(X, Y, Z)$ is a multilinear polynomial over the set of Boolean variables $X = \{X_{uv} | u, v \in V\} \cup \{X_u | u \in X\} \cup \{Y_u | u \in Y\} \cup \{Z_u | u \in Z\}$.

Remark 23. *Given a simple undirected graph* $\mathbf{G} = (V, E)$ *and any subsets* $X, Y, Z \subseteq V$ *then* $\mathbf{G}$ *is 3-colorable if and only if,* $p_\psi(X, Y, Z) = (\text{card } V)^2$. $p_\psi(X, Y, Z) < (\text{card } V)^2$ *otherwise.*

In example 4 the polynomial expressions given in (5.37) and (5.38) depend on the given partition (X, Y, Z). Then, it is not possible to state the optimization problem using the algorithm 2 for the 3-coloring problem. A possible alternative to construct a polynomial expression for the 3-coloring problem is by considering an objective function over all possible partitions such that a partition (X, Y, Z) is a 3-coloring if and only if, the objective function is minimized.

Proposition 6. *Any existential monadic second order logic sentence is suitable to be expressed through a polynomial map defined on the hypercube.*

Proposition 6 can be proved by considering the algorithm 2, but restricted to subset graph problems, i.e. the Maximum Clique Problem.

Remark 24. *The polynomial expression obtained by the algorithm 2 can be considered as a penalty function in (5.35).*

The polynomial expression returned by the algorithm 2 can be reduced to a quadratic form and subsequently to be optimized using an AQO algorithm. Also, from the remark 24 the objective function defined in (5.35) can also be used in AQO. These polynomial expressions provide us a general scheme to deal with optimization problems that are expressible in MSOL.

Chapter 6

A general strategy to solve NP-hard problems

In computational complexity, it is well known that NP-hard problems are the most difficult problems to solve and in many cases only an approximation to the optimal solution is given (see [8]). On the other hand, logical characterization of NP-problems has provided a classification of NP optimization problems in terms of first and second order logic expressions [47, 55, 4, 88]. *Tree-decomposition* and *treewidth* of graphs are important concepts introduced in a series of publications on graph minors [69, 70, 71, 68, 72]. In [28, 29, 27] show that on graphs of bounded treewidth, every decision or optimization problem expressible in Monadic Second Order Logic (MSOL) has a linear time algorithm. In [54, 13, 14, 15, 81, 18, 17, 16] show that there is a *dynamic programming approach* on tree-decomposition on graphs of bounded treewidth to solve optimization problems in linear time.

Another important concept related to the treewidth is the *Clique-width* [28, 41, 30], it has been considered to show linear time algorithms on graphs of bounded clique-width. Recently, it has been considered the *Tree-Depth* as a parameter to build efficient algorithms [60]. This chapter is divided into two parts, the first one is devoted to a study on tree-decompositions, we analyze the iterative construction and updating of tree-decompositions, this is done by adding one edge at a time [33]. In the second part we consider the dynamic programming approach to solve optimization problems, we propose a solution to the *classical Ising spin glass model* based on the Dynamic Programming approach. We also propose a composition strategy of local Hamiltonians for AQC on tree-decompositions of graphs.

6.1 Background

6.1.1 Basic notions

Let $\mathbf{G} = (V, E)$ be a graph with $V(\mathbf{G}) = V$ as set of vertices and $E(\mathbf{G}) = E$ as set of edges, let $n = |V|$ be the number of vertices, or *graph order*. For any subset

$S \subseteq V$ of vertices, the subgraph of $\mathbf{G}$ *induced* over S, denoted by $\mathbf{G}[S]$, is the graph $\mathbf{S} = (S, E_S)$ where $E_S = \{\{x, y\} \in E | x, y \in S\}$. A *clique* in $\mathbf{G}$ is a complete subgraph of $\mathbf{G}$. The *clique number* $\omega(\mathbf{G})$ is the size of the largest clique in $\mathbf{G}$.

If $i < j$, then $[\![i, j]\!]$ will denote the set of integers $\{i, i+1, \ldots, j-1, j\}$, and from now on we will identify V with $[\![0, n-1]\!]$.

A *Hamiltonian path* in $\mathbf{G}$ may be realized as a subgraph $\mathbf{P}_\pi = (V, E_\pi)$ of $\mathbf{G}$, where π is a permutation of V and $E_\pi = \{\{\pi(i), \pi(i+1)\} | 0 \leq i < n-1\}$. The vertices $\pi(0)$ and $\pi(n-1)$ are *linked* by $\mathbf{P}_\pi$ and are called the *ending points* of the path.

A *Hamiltonian cycle* has the form $\mathbf{C}_\pi := \mathbf{P}_\pi + \{\pi(0), \pi(n-1)\}$ where $\mathbf{P}_\pi$ is a Hamiltonian path. In other words, a Hamiltonian cycle is a Hamiltonian path whose ending points form an edge.

A *cycle* is a Hamiltonian cycle in a subgraph of $\mathbf{G}$. The *length* of a cycle is the number of its edges. A *chord* is an incident edge to two vertices that are not adjacent within the cycle. The graph $\mathbf{G}$ is *triangulated* (or *chordal*) if every cycle of length at least 4 has a chord. A *triangulation* of $\mathbf{G}$ is a graph $\mathbf{H}$ with the same set of vertices such that $\mathbf{G}$ is a subgraph of $\mathbf{H}$ and $\mathbf{H}$ is triangulated, and it is a *minimal triangulation* of $\mathbf{G}$ if there is no triangulation of $\mathbf{G}$ that is a proper subgraph of $\mathbf{H}$.

Definition 12. *The notion of* k-tree *is defined recursively as follows:*

1. *A clique with* $k+1$ *vertices is a* k*-tree.*

2. *Given a* k*-tree* $\mathbf{T}_n$ *with* n *vertices, it is expanded to a* k*-tree with* $n+1$ *vertices as follows: add a new vertex* x_{n+1}*, choose a* k*-clique of* $\mathbf{T}_n$ *and connect* x_{n+1} *with each vertex in the chosen* k*-clique.*

A *partial* k*-tree* is a subgraph of a k-tree with the same set of vertices. The *treewidth* of a graph $\mathbf{G}$ is the minimum value k for which $\mathbf{G}$ is a partial k-tree. Any k-tree has treewidth k.

Problem 1 (Treewidth Problem)**.** *Given a graph* $\mathbf{G}$ *and an integer* $k \geq 1$*, decide whether the treewidth of* $\mathbf{G}$ *is at most* k*.*

As was shown in [5] the treewidth problem is NP-complete. However, the Treewidth Problem restricted to graphs with treewidth bounded by a parameter $k_b \in \mathbb{Z}^+$, is decidable in linear time [54].

A simple characterization of k-trees was shown in [73]:

Lemma 1 (Rose, 1974)**.** *A graph* $\mathbf{G}$ *with* n *vertices is a* k*-tree if and only if* $\mathbf{G}$ *is triangulated,* $\omega(\mathbf{G}) = k+1$*, and* $|E(\mathbf{G})| \geq nk - \frac{1}{2}k(k+1)$*.*

A characterization of triangulated graphs was shown in [39]. Let us recall it: Let $\mathbf{G} = (V, E)$ be a graph, and let $x \in V$. The vertex x is *simplicial* if the subgraph induced by $\mathbf{G}$ over the neighborhood $N(x) := \{y \in V | \{x, y\} \in E\}$ is a clique. Let σ be a permutation of V. For an index $i \in [\![0, n-1]\!]$, let $\mathbf{G}[\sigma(i, n)]$ denote the subgraph $\mathbf{G}[S_i]$ induced by $\mathbf{G}$ over $S_i = \{\sigma(i), \ldots, \sigma(n-1)\}$. It is said that σ is a *perfect*

elimination scheme (PES) in $\mathbf{G}$ if for each $i \in [\![0, n-1]\!]$, the vertex $\sigma(i)$ is simplicial in $\mathbf{G}[\sigma(i, n)]$.

The triangulated graphs are determined as follows:

Lemma 2 (Fulkerson & Gross, 1965)**.** *A graph $\mathbf{G}$ is triangulated if and only if there exists a PES for $\mathbf{G}$. Furthermore, if a graph is triangulated, any simplicial vertex can start a PES for the graph.*

Now let us recall the *Minimum Triangulation Problem*. Let σ be a permutation of the vertex set V. The *fill-in* produced by σ, denoted $Fill(\sigma)$, is a set of new edges that should be added to the graph $\mathbf{G}$ in such a way that for each $i \in [\![0, n-1]\!]$, the vertex $\sigma(i)$ becomes simplicial in $\mathbf{G}[\sigma(i, n)]$. Consequently, if σ is a PES then $Fill(\sigma) = \emptyset$.

Problem 2 (Minimum Fill-in)**.** *Given a graph $\mathbf{G} = (V, E)$, find a permutation σ of V such that $Fill(\sigma)$ is minimum.*

Equivalently, the Minimum Fill-in Problem can be stated as finding the minimum set of edges whose addition to the graph $\mathbf{G}$ is a chordal graph. The Minimum Fill-in Problem is indeed NP-complete [85], however, the decision of whether a given graph $\mathbf{G}$ is triangulated, can be done in linear time with respect to the number of vertices [74, 59, 53, 44].

The algorithm 1 Fill-in($\mathbf{G}, \pi$) below receives a graph $\mathbf{G}$ and a permutation π of the set of vertices, and constructs a triangulation $\mathbf{H}$ of $\mathbf{G}$ such that π is a PES in $\mathbf{H}$, by adding a minimum number of edges to $\mathbf{G}$. Clearly, the time complexity of this algorithm is of the order $O(nm)$, where $m = |E|$ is the number of edges of the input graph.

Algorithm 1 Fill-in($\mathbf{G}, \pi$)

Input: A graph $\mathbf{G} = (V, E)$ with $n = |V|$ and a permutation $\pi : [\![0, n-1]\!] \to V$.
Output: A triangulation $\mathbf{H}$ of $\mathbf{G}$ such that π is a PES for $\mathbf{H}$.

```
H := G;
for all i = 0, ..., n − 1 do
    Let v = π(i) be the i-th vertex according to π;
    for all pair w, u ∈ N(v) such that π^{-1}(w) > i, π^{-1}(u) > i do
        if w and u not adjacent in H then
            Add {w, u} to H
        end if
    end for
end for
Return H.
```

6.1.2 Tree decompositions

Definition 13. *A tree decomposition of a graph* $\mathbf{G} = (V, E)$ *is a pair* $(\mathbf{T}, \mathcal{X})$ *where* $\mathbf{T} = (T, F)$ *is a tree, and* $\mathcal{X} = (X_t)_{t \in T}$ *is a family of subsets of* V *such that the following conditions are satisfied:*

1. $\bigcup_{t \in T} X_t = V$,
2. $\forall \{u, v\} \in E \ \exists t \in T$: $u, v \in X_t$, *and*
3. $\forall x \in V$ *the subgraph induced by* $\mathbf{T}$ *over* $\{t \in T | x \in X_t\}$ *is a subtree of* $\mathbf{T}$.

Alternatively the condition 3. can be formulated as follows:

3'. *For all* $t_1, t_2, t_3 \in T$, *if* t_2 *is on the path connecting* t_1 *with* t_3 *in* $\mathbf{T}$ *then* $X_{t_1} \cap X_{t_3} \subset X_{t_2}$.

For each tree vertex $t \in T$, the subset $X_t \subset V$ of graph vertices is called its *bag*. The *width* of a tree decomposition $(\mathbf{T}, \mathcal{X})$ is $\max_{t \in T} |X_t| - 1$. The *treewidth* of a graph $\mathbf{G}$ is the minimum width over all possible tree decompositions of $\mathbf{G}$, and it is written as $\mathrm{tw}(\mathbf{G})$.

Definition 14. *A* branch decomposition *of a graph* $\mathbf{G} = (V, E)$ *is a tree decomposition* $(\mathbf{T}, \mathcal{X})$ *such that* $\mathbf{T}$ *is just a branch, namely a path.*

The *branchwidth* of the graph $\mathbf{G}$ is the minimum width over all possible branch decompositions of $\mathbf{G}$.

The algorithm 2 below produces a tree decomposition of $\mathbf{G}$, with the same set of vertices, assuming that a PES π in $\mathbf{G}$ is given. For each vertex at $\mathbf{G}$, it is required the computation of the subgraph $\mathbf{G}[\pi(k, n)]$, and then an exploration on the edges is necessary in order to find the index j in the main cycle of the algorithm, thus, the time complexity of the algorithm is of the order $O(n^2 m)$.

In order to produce a tree decomposition of a graph in a general setting, with the algorithm 1, and any permutation π, a triangulation $\mathbf{H}$ is produced with π as PES and then the algorithm 2 produces the tree decomposition. Since a triangulation of a graph with n vertices will have $O(n)$ edges, the composition of algorithm 1 with algorithm 2 has time complexity $O(n^2 m)$.

6.2 Procedural modification of tree decompositions

Let us consider a general algorithm to construct a tree decomposition using elimination schemes: Given a graph $\mathbf{G} = (V, E)$, a tree decomposition $(\mathbf{T}, \mathcal{X})$ of $\mathbf{G}$ is obtained as follows:

$$(\mathbf{G}, \pi) \overset{\Psi}{\longmapsto} \mathbf{H}_\pi \overset{\Phi}{\longmapsto} (\mathbf{T}, \mathcal{X}) \tag{6.1}$$

where Ψ is a procedure to triangulate $\mathbf{G}$ in such a way that π is a PES in $\mathbf{H}_\pi$ (for instance, the algorithm 1), and Φ is the transformation calculated by the algorithm 2.

Algorithm 2 GeneralTreeDecomposition($\mathbf{G}, \pi$)

Input: A graph $\mathbf{G} = (V, E)$, $n = |V|$, and a PES $\pi : [\![0, n-1]\!] \to V$ of $\mathbf{G}$.
Output: A tree decomposition $(\mathbf{T} = (V, F), \mathcal{X})$ of $\mathbf{G}$.

```
Let ((T, F), X) = ((V, ∅), ∅) be the initial empty tree decomposition;
for all k = n − 1, . . . , 0 do
  if k == n − 1 then
    X_{π(k)} = {π(k)};
  else
    Let G' = (V', E') := G[π(k, n)];
    Let π(j) be the lowest numbered neighbor of π(k) in G', i.e.,
    j := min{i ∈ [[0, n − 1]] | {π(k), π(i)} ∈ E'};
    Let X_{π(k)} := N(π(k), G') ∪ {π(k)}: the neighborhood of π(k) in G';
    Let F := F ∪ {π(k), π(j)};
  end if
end for
Return (T := (V, F), X).
```

Any tree decomposition obtained using the general procedure (6.1), depends in the vertex ordering determined by π. In general, for a fixed triangulation, different PES's will produce different tree decompositions.

6.2.1 Modification by the addition of an edge

Now, suppose that we have an already constructed tree decomposition $(\mathbf{T}, \mathcal{X})$ of a graph $\mathbf{G}$. Let us modify the graph by the addition of an edge. Let $\mathbf{G} + e$ be the new graph, and let us pose as a task to build a corresponding tree decomposition $(\mathbf{T}', \mathcal{X}')$ of $\mathbf{G} + e$, see the following diagram:

$$\begin{array}{ccccc} \mathbf{G} & \overset{A}{\longmapsto} & \mathbf{G}+e & & \\ \Big\downarrow{\scriptstyle \Phi\circ\Psi} & & \vdots & \searrow{\scriptstyle \Phi\circ\Psi} & \\ (\mathbf{T},\mathcal{X}) & \underset{B}{\rightsquigarrow} & (\mathbf{T}',\mathcal{X}') & & (\mathbf{T}'',\mathcal{X}'') \end{array} \tag{6.2}$$

where $(\Phi \circ \Psi)$ is the general algorithm sketched at diagram (6.1), A is the addition of an edge to a graph, and B is the sought corresponding tree decomposition transformation. Naturally, the procedure $(\Phi \circ \Psi)$ can be applied to the modified graph $\mathbf{G}+e$, producing thus a tree decomposition $(\mathbf{T}'', \mathcal{X}'')$. Both trees $(\mathbf{T}', \mathcal{X}')$ and $(\mathbf{T}'', \mathcal{X}'')$ are tree decompositions of the graph $\mathbf{G} + e$.

It is important to note that, when an edge e is added to $\mathbf{G}$, it is not generally true that the current triangulation $\mathbf{H}_\pi = \Psi(\mathbf{G})$ remains a triangulation of the modified graph $\mathbf{G} + e$, thus it would be necessary to construct a new triangulation for $\mathbf{G} + e$. However, the lemma 2 asserts that there exists a PES which can be used to test whether $\mathbf{H}_\pi + e$ is triangulated or not. In the affirmative case no new computation of the triangulation is required.

The next section gives some results about this approach.

6.2.2 Iterative modification

Let $\mathbf{G} = (V, E)$ be a graph with $n = |V|$ vertices and let $\mathbf{H}$ be a triangulation of the graph $\mathbf{G}$. By the lemma 2, there exists a PES π in $\mathbf{H}$. For each index $i \in [\![0, n-1]\!]$, let $N_{\pi(i)}$ be the neighborhood of $\pi(i)$ in $\mathbf{H}[\pi(i, n)]$. Then $\mathbf{H}[\pi(i, n)]$ induces a clique over $N_{\pi(i)}$.

Remark 25. *Let $e = \{u, v\}$ be an edge not in $\mathbf{H}$ and let $j_1 := \min\{\pi^{-1}(u), \pi^{-1}(v)\}$ and $j_2 := \max\{\pi^{-1}(u), \pi^{-1}(v)\}$. Then the following conditions are equivalent:*

1. *$\mathbf{H}' := \mathbf{H} + e$ is triangulated and it has π as a PES.*
2. *$\mathbf{H}'[\pi(j_1, n)]$ induces a clique over $N_{\pi(j_1)} \cup \{\pi(j_2)\}$.*

Let us say that any edge $e \in E$ *maintains the triangulation* if condition 1. in remark 25 holds. In a procedural way, the checking of whether an edge maintains the triangulation can be done, through the condition 2., in time complexity $O(\text{degree}_{\mathbf{H}}(j_1)^2) \leq O(|E|^2)$.

Hence if an edge e is added to a triangulated graph $\mathbf{H}_\pi$, and π remains as a PES for $\mathbf{H}_\pi + e$, then the neighborhood $N_{\pi(j)}$ of just one vertex $\pi(j)$ increases by one element, while the other neighborhoods do not change.

Claim 1. *Let $\mathbf{H}$ be a triangulation of a graph $\mathbf{G}$, different than the whole clique K_V, and let π be a PES in $\mathbf{H}$. Then, there exists an edge e, not in $E(\mathbf{G})$, that maintains the triangulation.*

The selection of such an edge e can be applied iteratively until the arrival to the whole clique K_V.

Proof. Let $e \in E$ be an edge not in the triangulated graph $\mathbf{H}_\pi$, and let j_1, j_2 be two indexes defined as in remark 25. Let j be an index such that $j_1 < j$ and $\pi(j) \in N_{\pi(j_1)}$, and let $N' = \{\pi(j_2)\} \cup (N_{\pi(j_1)} \backslash \{\pi(j)\})$. Then $\mathbf{H} + e$ remains triangulated while $N' \subset N_{\pi(j)}$, just because $\mathbf{H}[\pi(j, n)]$ induces a clique over N' and $\pi(j)$ is connected to N'. □

The claim 1 is the basis of an iterative procedure: Choose a PES π for $\mathbf{H}$, then pick an edge $e \in E - E(\mathbf{H})$, and check whether π is a PES for $\mathbf{H}' := \mathbf{H} + e$, in which case update $\mathbf{H}'$ as the triangulated graph. Repeat the procedure.

Now, let $(\mathbf{T} = (V, F), \mathcal{X})$ be a tree decomposition of a graph $\mathbf{G} = (V, E)$ obtained using the general algorithm $\Phi \circ \Psi$, then there exists a triangulation $\mathbf{H}_\pi$ of $\mathbf{G}$, for a PES π in $\mathbf{H}$, such that Ψ produces $(\mathbf{T}, \mathcal{X})$ from $\mathbf{H}_\pi$.

Remark 26. *Let $e = \{u, v\}$ be an edge such that $\mathbf{H}_\pi + e$ is triangulated. Let $(\mathbf{T}' = (V, F'), \mathcal{X}')$ be its tree decomposition as defined in algorithm 2. Using the notation at remark 25, let $\pi(j)$ be the lowest numbered neighbor of $\pi(j_1)$ in $\mathbf{H}_\pi[\pi(j_1, n)]$. Then, initially $(\mathbf{T}', \mathcal{X}') = (\mathbf{T}, \mathcal{X})$ and consecutively it is modified according to the following cases:*

1. *If* $j < j_2$ *then* $\tau'(\pi(j_1)) := \tau'(\pi(j_1)) \cup \{\pi(j_2)\}$.

2. *If* $j > j_2$ *then* $\tau'(\pi(j_1)) := \tau'(\pi(j_1)) \cup \{\pi(j_2)\}$, $F' := F' \backslash \{\pi(j_1), \pi(j)\}$ *and* $F' := F' \cup \{\pi(j_1), \pi(j_2)\}$.

From the remark 26, it is easy to see that the tree decomposition $\mathbf{T}'$ does not change by adding new edges that maintain the triangulation and satisfies the case 1., while the family of bags $\mathcal{X}'$ grows in one graph vertex at just one tree vertex when a new edge is added. Also, it is important to note that the width of the tree $\mathbf{T}'$ increases in one only when the cardinality of the bag at vertex $\pi(j_1)$ is the greatest in $\mathbf{T}$.

If an edge e does not maintain the triangulation, due to claim 1 there exists a sequence of edges $e_1, \ldots, e_k$ such that for the sequence of triangulated graphs $(\mathbf{H}_i)_{i=0}^{k}$, with $\mathbf{H}_0 = \mathbf{H}$ and $\mathbf{H}_i = \mathbf{H}_{i-1} + e_i$, the edge e_i maintains the triangulation $\mathbf{H}_{i-1}$ and the last edge e_k coincides with the original edge e. Let us say that the edge sequence $e_1, \ldots, e_k$ is a *climbing sequence* for e.

Then, the following problem can be posed:

Problem 3. *For a given edge* e *that does not maintain the triangulation* $\mathbf{H}$*, find the minimum length among all possible climbing sequences for* e*.*

The algorithm 3 below solves this problem by adding the necessary edges to the triangulation $\mathbf{H}$, in such a way that $\mathbf{H}[\pi(j_1, n)]$ induces a clique over $N_{\pi(j_1)} \cup \{\pi(j_2)\}$ and by repeating the same task for each element in $N_{\pi(j_1)}$.

We see that the array A acts as a queue in order to perform a breadth-first examination of potential edges in a climbing sequence. Hence, the time complexity of this algorithm is of the order $O(m)$, where m is the number of edges in the input triangulation.

6.2.3 Branch decompositions

An application of the iterative modification of tree decompositions is the following: Let $\mathbf{G}$ be a graph, $\mathbf{H}_\pi = \Psi(\mathbf{G})$ be a triangulation of $\mathbf{G}$ and let $(\mathbf{T}, \mathcal{X}) = \Phi \circ \Psi(\mathbf{G})$ be its tree decomposition as defined in algorithm 2 from $\mathbf{H}_\pi$. Then, it is possible to obtain a branch decomposition from $(\mathbf{T}, \mathcal{X})$ by adding new edges to $\mathbf{H}_\pi$ in order to maintain the triangulation with respect to π.

The idea behind this transformation from a tree to a branch decompositions is a consequence of the case 2. in remark 26. Namely, when the condition 2. is fulfilled, the tree shrinks. Then, by adding the necessary edges to the triangulation $\mathbf{H}_\pi$ satisfying the condition 2., the tree becomes a branch.

Claim 2. *Let* $\mathbf{G} = (V, E)$ *be a graph with* $n = |V|$*,* $\mathbf{H}_\pi = \Psi(\mathbf{G})$ *be a triangulation of* $\mathbf{G}$ *and let* $(\mathbf{T}, \mathcal{X}) = \Phi \circ \Psi(\mathbf{G})$ *be its tree decomposition as defined in algorithm 2. Then, there exists a set of edges* $\{e_1, \ldots, e_k\}$ *such that when adding to* $\mathbf{H}_\pi$ *successively (satisfying remark 26),* $(\mathbf{T}, \mathcal{X})$ *becomes a branch decomposition* $(\mathbf{T}', \mathcal{X}')$ *where* $\mathbf{T}' =$

Algorithm 3 MinimumClimbing

Input: An edge $e = \{u, v\}$, a triangulation $\mathbf{H}$ of a graph and a PES π for N.
Output: The length of the minimal climbing sequence for e.

```
Let A := ∅; Let j1, j2 be defined as in remark 25;
for all w ∈ N_{π(j1)} do
  if w is not adjacent to π(j2) then
    Add {w, π(j2)} to N; A = A ∪ {w};
  end if
end for
B = A;
while A is not empty do
  Let a ∈ A and j = π^{-1}(a);
  for all w' ∈ N_{π(j)} do
    if w' is not adjacent to π(j2) then
      Add {w', π(j2)} to H;
      B = B ∪ {w'}; A = A ∪ {w'}
    end if
  end for
  A = A − {a};
end while
Return |B|.
```

(V, F'), $F' = \{\{\pi(i), \pi(i+1)\} | 0 \leq i \leq n-2\}$ *and* $\mathcal{X}'$ *is described according to the remark 26.*

Proof. By the case 2. in remark 26, when a new edge e is added to $\mathbf{H}_\pi$, the edge $\{\pi(j_1), \pi(j)\}$ is deleted from the tree and replaced by the edge $\{\pi(j_1), \pi(j_2)\}$ where the index $j_2 < j$. Hence, the lowest index j' such that $j' < j$ satisfies $j' = j_1 + 1$, and it corresponds indeed to the edge $\{\pi(j_1), \pi(j_1 + 1)\}$. □

Remark 27. *Let* $\mathbf{G} = (V, E)$ *be a graph,* $\mathbf{H}_\pi$ *be a triangulation of* $\mathbf{G}$ *and let* $(\mathbf{T}, \mathcal{X})$ *be its tree decomposition as defined in algorithm 2. By the claim 1 it is possible to arrive to the whole clique* K_V*, adding edges successively to* $\mathbf{H}_\pi$*. Then,* $(\mathbf{T}, \mathcal{X})$ *evolves into a branch decomposition of the complete graph* K_V*.*

The remark 27 determines an upper bound of the number of required edges to transform a tree decomposition into a branch decomposition.

Then, the following problem can be posed:

Problem 4. *For a given tree decomposition* $(\mathbf{T}, \mathcal{X})$ *of a graph* $\mathbf{G}$*, find the minimum number of edges satisfying the claim 2, in order to transform* $(\mathbf{T}, \mathcal{X})$ *into a branch decomposition.*

Let us considerer the following example: Let $\mathbf{G} = (V, E)$ be a graph where:

$$\begin{aligned} V &= \{0, 1, 2, 3, 4, 5, 6\} \text{ and} \\ E &= \{\{0,1\}, \{0,6\}, \{2,1\}, \{2,3\}, \{2,6\}, \{3,5\}, \{3,4\}, \{4,5\}, \{5,6\}\}. \end{aligned}$$

Let $\pi := (6, 3, 4, 5, 1, 0, 2)$ be a permutation of V, the algorithm 1 with $(\mathbf{G}, \pi)$ as input returns a triangulation $\mathbf{H}_1 = (V, E')$ of $\mathbf{G}$ where:

$$E' = E \cup \{\{0, 2\}, \{0, 5\}, \{2, 5\}, \{2, 4\}\}$$

and π is a PES for $\mathbf{H}_1$. Using the algorithm 2 with $(\mathbf{H}_1, \pi)$ as input a tree decomposition T_1 is obtained. Note that π remains a PES for $\mathbf{H}_2 := \mathbf{H}_1 + \{1, 5\}$ and from the remark 26, T_1 becomes a tree decomposition T_2 that satisfy condition 2. In the same way, π remains a PES for $\mathbf{H}_3, \mathbf{H}_4$ where $\mathbf{H}_3 := \mathbf{H}_2 + \{0, 4\}, \mathbf{H}_4 := \mathbf{H}_3 + \{0, 3\}$, and its corresponding tree decompositions T_3 and T_4 that satisfy condition 1 in remark 26.

Finally, π is a PES for $\mathbf{H}_5 := \mathbf{H}_4 + \{3, 6\}$ and T_4 becomes a branch decomposition T_5 that satisfies condition 2 in remark 26.

Note also that $\{1, 5\}, \{0, 4\}, \{0, 3\}, \{3, 6\}$ are the minimum number of edges to transform T_1 into the branch decomposition T_5.

6.2.4 Comparison of time complexities

As in last section, let $\Phi \circ \Psi$ be the map calculated by the algorithm 1 Fill-in$(\mathbf{G}, \pi)$, followed by the algorithm 2 GeneralTreeDecomposition$(\mathbf{G}, \pi)$.

According to the diagram (6.2), a tree decomposition $(\mathbf{T}', \mathcal{X}')$ of the graph of the form $\mathcal{G} + e$, can be done either through the transformation $B \circ \Phi \circ \Psi$ or through $\Phi \circ \Psi \circ A$.

If the edge e maintains the triangulation $\mathbf{H} = \Psi(\mathbf{G})$, then $\mathbf{T}' = \mathbf{T}$, hence $B \circ \Phi \circ \Psi(\mathbf{G}) = \Phi \circ \Psi(\mathbf{G})$. By remark 25 the checking of whether e maintains the triangulation is proportional to the square of the degree of the triangulation. Thus, in general the time complexity of $B \circ \Phi \circ \Psi(\mathbf{G})$ is $O(n^2 m)$, where $n = |V(\mathbf{G})|$ and $m = |E(\mathbf{G})|$.

On the other hand, the processing of $\Phi \circ \Psi \circ A$ entails a time complexity $O(n^2(m + 1))$, since a new edge e is added, and also the checking of whether e maintains the triangulation is performed.

Thus we see that the processing of $B \circ \Phi \circ \Psi$ is more convenient than that of $\Phi \circ \Psi \circ A$.

This work can be generalized as in [19, 20] in order to provide a deterministic algorithm for tree decompositions for a greater graph class. The generalization entails a potential application in the solution of graph problems with the use of dynamic programming approaches.

6.3 A strategy to solve NP-hard problems

In this section a brief introduction to the applications of the tree decomposition of graphs is given. We show how to exploit a tree decomposition to solve optimization problems using a Dynamic Programming approach. We use the language of MSOL to express properties of graph problems. Finally, the *Courcelle theorem* is introduced

and a possible application in the design of local Hamiltonian operators for AQC is given.

6.3.1 Dynamic programming approach

Let $k \in \mathbb{Z}^+$ be an integer. Let $\mathcal{B}_k$ be the class of graphs $\mathbf{G} = (V, E)$ such that $\mathrm{tw}(\mathbf{G}) \leq k$.

Theorem 6 (Robertson-Seymour, 1986)**.** *For each $k \in \mathbb{Z}^+$, $\mathcal{B}_k$ can be characterized by finite sets of forbidden minors.*

The Dynamic Programming approach for solving NP-hard problems consists in solving partial instances and then to ensemble the corresponding solutions into a solution of the whole initial instance. The deal is *computing tables of characterizations of partial solutions.*

A *nice tree decomposition* $(\mathbf{T} = (T, F), \mathcal{X})$ is a rooted binary tree having nodes of just four types:

- a *start*, a node with no children,
- a *join*, a node with two children, and whose bag is the union of its children's bags,
- a *forget*, a node with just one node, and whose bag is a subset of its child bag, and
- a *introduce*, a node with just one node, and whose bag is a superset of its child bag.

Proposition 7 (Bodlaender, 1998)**.** *Any tree decomposition of width k of a graph $\mathbf{G}$ can be transformed into a nice decomposition tree of the same width with $O(k \operatorname{card}(V(\mathbf{G})))$ nodes in linear time.*

Thus from now on, it can be assumed that all tree decompositions are nice.

A *partial solution* of a problem corresponds to a bag in the decomposition tree. The computation of partial solutions is performed bottom-up ("bottom" corresponds to the leaves, "up" to the root), thus the partial solution at any node is computed from the partial solutions of its children. The partial solution at the root will be the *whole solution.*

For any bag X_t with $t \in T$ in the tree decomposition, let $\mathbf{G}_t$ be the subgraph of $\mathbf{G}$ whose nodes are the vertexes at the bag X_t and its descendants:

$$V(\mathbf{G}_t) = \bigcup \{t' \in T |\ X_t = X_{t'} \text{ or } [t' \text{ is a descendant of } t \text{ in } \mathbf{T}]\}.$$

In a general way, the following procedural steps synthesizes the Dynamic Programming reduction: Let P be a problem that given a graph $\mathbf{G}$ has associated a solution $\mathrm{sol}_P(\mathbf{G})$.

1. Define the notion of a *partial solution*: For a bag X_t for some $t \in T$, it should be the restriction to $\mathbf{G}_t$ of a solution $\text{sol}_P(\mathbf{G})$. Observe that this may coincide or not with $\text{sol}_P(\mathbf{G}_t)$.
2. Define the notion of *partial solutions extension* within a tree decomposition.
3. Define the notion of *partial solution characteristic* within a tree decomposition. Most generally, the characteristic of a partial solution at a bag X_t for some $t \in T$ is the restriction to the bag X_t of the partial solution at the graph $\mathbf{G}_t$.
4. Show that for any of the three bag types, there is a polynomial-time algorithm to find the characteristic.
5. Show that the characteristic of the root produces indeed a whole solution.

6.3.2 The Courcelle Theorem

Courcelle showed that every problem definable in Monadic Second-Order Logic (MSOL) can be solved in linear time on graphs with bounded treewidth. The Courcelle theorem has important applications for several fixed parameter tractability results. Many problems can be expressed in MSOL such as Minimum Vertex Cover, Minimum Dominating Set, and Maximum Independent.

Using the definitions of the previous section on tree decompositions, let

$$\mathcal{B}_k = \{\mathbf{G} | \mathbf{G} \text{ a graph and } \text{tw}(\mathbf{G}) \leq k\}.$$

We say that φ be a well formed sentence in MSOL for a graph $\mathbf{G}$ then $\mathbf{G}$ is called a model for φ.

Theorem 7 (Courcelle Theorem [27])**.** *Let $k \geq 1$ and φ a sentence in MSOL. There exists a linear time algorithm such that for each graph $\mathbf{G} \in \mathcal{B}_k$ decide if $\mathbf{G}$ is a model of φ.*

An application of the Courcelle theorem A *kernel* in a directed graph $\mathbf{G} = (V, E)$ is a subset K of V such that, no two vertices in K are adjacent and for every vertex $a \in V \backslash K$ there is a vertex $b \in K$ such that $(a, b) \in E$.

Let $k \geq 1$ and $\mathbf{G}$ be a graph with $\text{tw}(\mathbf{G}) \leq k$. There exist a liner time algorithm which decide whether $\mathbf{G}$ has a kernel.

6.3.3 Examples of second order formulae

Independent set

An *independent set* U in a graph $\mathbf{G} = (V, E)$ is a set of vertexes, $U \subseteq V$, containing no pair of edge extremes. As a second order formula, this can be stated by:

$$\phi(U, V) \equiv \forall v_0, v_1 \in V : [v_0, v_1 \in U \ \Rightarrow \ \{v_0, v_1\} \notin E] .$$

Three-coloring

For a graph $\mathbf{G} = (V, E)$ a *coloring* with *n-colors* or *n-coloring*, is a map $\gamma_n : V \to N$ where N is a set of cardinality n, such that no pair of adjacent vertexes share a common color:

$$\forall v_0, v_1 \in V : \ \{v_0, v_1\} \in E \implies \gamma_n(v_0) \neq \gamma_n(v_1).$$

Namely, an n-coloring can be realized as the partition $(\gamma_n^{-1}(i))_{i \in N}$ of its monochromatic sets, in which all edges traverse them.

In order to put the notion of 3-*colorability* as a second order formula let us introduce the following formulae:

- Set inclusion:
$$\phi_{10}(V, W) \equiv W \subseteq V.$$
- Three sets form a partition of the set of vertexes:
$$\begin{aligned} \phi_1(V, W_0, W_1, W_2) \ \equiv \ & \phi_{10}(V, W_0) \wedge \phi_{10}(V, W_1) \wedge \phi_{10}(V, W_2) \wedge \\ & [\forall v \in V : (v \in W_0) \vee (v \in W_1) \vee (v \in W_2)] . \end{aligned}$$
- Two vertexes lie in different subsets:
$$\phi_{20}(v_0, v_1, W) \equiv \neg \left(v_0 \in W \wedge v_1 \in W\right).$$
- No edge lies within a monochromatic set:
$$\begin{aligned} & \phi_2(V, W_0, W_1, W_2) \equiv \forall v_0, v_1 \in V : \\ & [\{v_0, v_1\} \in E \Rightarrow \phi_{20}(v_0, v_1, W_0) \wedge \phi_{20}(v_0, v_1, W_1) \wedge \phi_{20}(v_0, v_1, W_2)] . \end{aligned}$$
- 3-colorability is thus stated by the sentence
$$\phi_3(V) \equiv \exists W_0, W_1, W_2 : \ \phi_1(V, W_0, W_1, W_2) \wedge \phi_2(V, W_0, W_1, W_2).$$

Hamiltonian cycle

A *Hamiltonian cycle* in a graph $\mathbf{G} = (V, E)$ is a permutation of the vertexes such that any pair of contiguous vertexes, modulus the order of the graph, form an edge. In order to put it as a second order formula let us define some predicates:

- Permutation definition: A map is a list of pairs $P = ((i, v_i))_{i=0}^{n-1}$ associating to each $i \in \{0, \ldots, n-1\}$ an unique point $v_i \in V$:
$$\phi_{00}(P, V) \equiv \bigwedge_{i=0}^{n-1} \left[\forall v_0, v_1 \in V : [(i, v_0) \in P \wedge (i, v_1) \in P \ \Rightarrow \ v_0 = v_1]\right],$$
and P is a permutation if it is one-to-one:
$$\phi_{01}(P, V) \equiv \bigwedge_{0 \leq i < j \leq n-1} \left[\forall v_0, v_1 \in V : [(i, v_0) \in P \wedge (j, v_1) \in P \ \Rightarrow \ v_0 \neq v_1]\right].$$
Let $\phi_0(P, V) = \phi_{00}(P, V) \wedge \phi_{01}(P, V)$.

- Each vertex has an index:

$$\phi_1(P,V) \equiv \forall v \in V : \bigvee_{i=0}^{n-1} [(i,v_0) \in P] .$$

- Two indexes correspond to an edge:

$$\phi_{2ij}(P,V) \equiv \forall v_0, v_1 \in V : \ [(i,v_0) \in P \wedge (j,v_1) \in P \ \Rightarrow \ \{v_0,v_1\} \in E] .$$

- Hamiltonian cycle:

$$\phi_3(P,V) \equiv \phi_0(P,V) \wedge \phi_1(P,V) \wedge \left[\bigwedge_{i=0}^{n-2} \phi_{2,i,i+1}(P,V) \right] \wedge \phi_{2,n-1,0}(P,V).$$

6.3.4 Dynamic Programming applied to NP-hard problems

The Dynamic Programming approach for solving NP-hard problems consists in solving partial instances and then to ensemble the corresponding solutions into a solution of the whole initial instance.

A *partial solution* of a problem corresponds to a bag in the tree decomposition. The computation of partial solutions is performed bottom-up ("bottom" corresponds to the leaves, "up" to the root), thus the partial solution at any node is computed from the partial solutions of its children. The partial solution at the root will be the *whole solution*.

In the following we expand the procedural steps of the Dynamic Programming approach sketched in section 6.3.1, we based on [16]:

A *terminal graph* is a triple $\mathbf{H} = (V, E, X)$ where (V, E) is a graph and the elements of $X \subseteq V$ are called the *terminals* of (V, E). Let $\mathbf{H}_1$ and $\mathbf{H}_2$ be two terminal graphs, $\mathbf{H}_1 \oplus \mathbf{H}_2$ is the disjoint union of $\mathbf{H}_1$ and $\mathbf{H}_2$. A terminal graph $\mathbf{H}_1$ is a terminal subgraph of a graph $\mathbf{G}$ if and only if, there exists a terminal graph $\mathbf{H}_2$ such that $\mathbf{G} = \mathbf{H}_1 \oplus \mathbf{H}_2$.

1. Define a notion of *solution*. Let Π be a problem or a graph property. Let $\mathrm{sol}_\Pi(\mathbf{G}, s)$ be a formula with two variables with $\mathbf{G}$ a graph and s a solution for the instance problem $\mathbf{G}$, such that:

$$\Pi(\mathbf{G}) \Longleftrightarrow \exists s : \mathrm{sol}_\Pi(\mathbf{G}, s).$$

2. Define the notion of *partial solution*: A partial solution is an object associated with a terminal graph. Let $\mathrm{psol}_\Pi(\mathbf{H}, s)$ be a formula with two variables with $\mathbf{H}$ a terminal graph and s a partial solution.

3. Define a notion of *extension of partial solutions*. Let $\mathrm{ex}_\Pi(\mathbf{G}, s, \mathbf{H}, s')$ be a formula with four variables with $\mathbf{G}$ a graph, s a solution for the instance problem

$\mathbf{G}$, $\mathbf{H}$ a terminal graph and s' a terminal solution for H. The following must hold, for all $\mathbf{G}, s, \mathbf{H}, s'$:

$$\mathrm{ex}_\Pi(\mathbf{G}, s, \mathbf{H}, s') \Longrightarrow \exists \mathbf{H}' : \mathbf{G} = \mathbf{H} \oplus \mathbf{H}' \wedge \mathrm{sol}_\Pi(\mathbf{G}, s) \wedge \mathrm{sol}_\Pi(\mathbf{H}, s').$$

The following condition expresses that every solution has a partial solution on any terminal graph:

$$\forall \mathbf{G}, s, \mathbf{H}, \mathbf{H}' : (\mathrm{sol}_\Pi(\mathbf{G}, s) \wedge \mathbf{G} = \mathbf{H} \oplus \mathbf{H}') \Longrightarrow \exists s' : \mathrm{psol}_\Pi(\mathbf{H}, s') \wedge \mathrm{ex}_\Pi(\mathbf{G}, s, \mathbf{H}, s').$$

4. Define a notion of *characteristic of a partial solution.* It is meant to describe what is needed to know about the partial solution to see whether it can be extended to a solution. Let $\mathrm{ch}_\Pi(\mathbf{H}, s) \mapsto \mathrm{psol}_\Pi(\mathbf{H}, s)$ be a function with $\mathbf{H}$ a terminal graph and s a terminal solution for $\mathbf{H}$. It must fulfill, for all terminal graphs $\mathbf{H}, \mathbf{H}', \mathbf{H}''$ and terminal solutions s, s':

   $$(\mathrm{ch}_\Pi(\mathbf{H}, s) = \mathrm{ch}_\Pi(\mathbf{H}', s')) \Longrightarrow (\exists s'' : \mathrm{ex}_\Pi(\mathbf{H} \oplus \mathbf{H}'', s'', \mathbf{H}, s)) \Longleftrightarrow$$
   $$(\exists s''' : \mathrm{ex}_\Pi(\mathbf{H}' \oplus \mathbf{H}'', s''', \mathbf{H}', s')).$$

5. A *full set of characteristics* for a terminal graph $\mathbf{G}$ is the set of all characteristics of partial solutions. For instance, let $\mathbf{H}$ be a terminal graph, the full set of characteristics for $\mathbf{H}$ is:

   $$\mathrm{full}_\Pi(\mathbf{H}) = \{\mathrm{ch}_\Pi(\mathbf{H}, s) \mid \mathrm{psol}_\Pi(\mathbf{H}, s)\}.$$

   Show that for every type of nodes in a nice tree decomposition, there is a polynomial-time algorithm to find the full set of characteristics.

6. Show that the characteristic of the root produces indeed a whole solution.

A solution to the Maximum Weight Independent Set problem:

Let $\mathbf{G} = (V, E, c)$ be a *weighted graph* where $c : V \to \mathbb{Z}^+$ is a weighted map. An *independent set* is a subset $S \subseteq V$ such that $\forall u, v \in S : \{u, v\} \notin E$. For any subset $S \subseteq V$, the *cost* of S is defined as $c(S) = \sum_{v \in S} c(v)$.

The Maximum Weight Independent Set problem
Input: A weighted graph $\mathbf{G} = (V, E, c)$
Solution: An independent set $S \subseteq V$ with maximum cost.

Let $(\mathbf{T} = (T, F), \mathcal{X})$ be a nice tree decomposition of $\mathbf{G}$. For each $t \in T$, let $\mathbf{G}_t = (V_t, E_t)$ be the subgraph of $\mathbf{G}$ at node t where $V_t = \{v | v \in X_{t_1}$ & ($t_1 = t$ or t_1 is a descendant of t in $\mathbf{T}$)$\}$ and $E_t = \{\{u, v\} \in E | u, v \in V_t\}$. The table of characteristics C_t at node t is such that, $\forall S \subseteq X_t, C_t(S) = \max_{W \subseteq V_t}\{c(W) | X_t \cap$

$W = S$ & W is an independent set}, in case that such independent set does not exist $C_t(S) = -\infty$.

The procedural steps of the Dynamic Programming approach will compute the tables C_t for all $t \in T$ in a bottom-up order. We proceed by cases according to the types of nodes on $\mathbf{T}$:

- *Start.* If t is a leaf node of $\mathbf{T}$, then $|X_t| = 1$ and $X_t = \{v\}$. The table C_t has only two entries: $C_t(\emptyset) = 0$ and $C_t(\{v\}) = c(v)$.
- *Introduce.* If t is an introduce node with a child t_1, then $X_t = X_{t_1} \cup \{v\}$ for a vertex v. Observe that $\mathbf{G}_t$ is formed from $\mathbf{G}_{t_1}$ by adding v and zero or more edges from v to vertices in X_{t_1}. It is satisfied that v is not adjacent to any vertex in $V_{t_1} - X_{t_1}$. For each $S \subseteq X_{t_1}$:
  1. $C_t(S) = C_{t_1}(S)$.
  2. If there is a vertex $w \in S$ with $\{v, w\} \in E$ then $C_t(S \cup \{v\}) = -\infty$.
  3. If for all $w \in S$, $\{v, w\} \notin E$ then $C_t(S \cup \{v\}) = C_{t_1}(S) + c(v)$.
- *Forget.* If t is a forget node with a child t_1, then $X_t = X_{t_1} - \{v\}$ for some vertex v, and the graphs $\mathbf{G}_t, \mathbf{G}_{t_1}$ are the same. Suppose that $v \in X_{t_1} - X_t$ is that vertex.

  For each $S \subseteq X_t : C_t(S) = \max\{C_{t_1}(S), C_{t_1}(S \cup \{v\})\}$.
- *Join.* If t is a join node with children t_1, t_2, then $X_t = X_{t_1} = X_{t_2}$.

  For each $S \subseteq X_t : C_t(S) = C_{t_1}(S) + C_{t_2}(S) - c(S)$.

Lemma 3. *The maximum weight of an independent set in* $\mathbf{G}$ *is* $\max_{S \subseteq X_{root}} C_{root}(S)$.

See [16] for a proof.

The independent set with maximum weight given by the Dynamic Programming solution can be constructed from the characteristic tables and does not introduce additional time complexity.

In the following we propose a solution to the *classical Ising Spin Glass* model based on the Dynamic Programming approach, we also introduce the Quantum Ising model and its relation with AQC.

6.3.5 The Classical Ising model

Let $\mathbf{G} = (V, E)$ be a graph with vertex set V and edge set $E \subset V^{(2)}$. Let $\mathbb{S} = \{-1, +1\}$ be the set of signs. An *assignment* is a map $\sigma : V \to \mathbb{S}$. An *edge weight* map is of the form $e : E \to \mathbb{R}$ and a *vertex weight* map is of the form $w : V \to \mathbb{R}$. In a physical context, an assignment is called a *spin configuration*, a positive edge weight e is said *ferromagnetic* and a negative edge weight is said *antiferromagnetic*. Let us enumerate $V = (v_i)_{i=0}^{n-1}$, thus there are 2^n assignments. For respective edge and vertex weight

e, w, let us write $e_{ij} = e(v_i, v_j)$ and $w_i = w(v_i)$. For those weight maps and an assignment σ, their *energy* is

$$\eta(e, w; \sigma) = -\sum_{\{v_i, v_j\} \in E} e_{ij}\sigma(v_i)\sigma(v_j) - \sum_{v_k \in V} w_k \sigma(v_k). \tag{6.3}$$

An assignment with minimum energy is called a *ground state*.

For a positive constant $\beta > 0$, let us consider the map

$$\phi(e, w, \beta; \cdot) : \sigma \mapsto \phi(e, w, \beta; \sigma) = \exp\left(-\beta\, \eta(e, w; \sigma)\right). \tag{6.4}$$

Let $\Phi(e, w, \beta) = \sum\{\phi(e, w, \beta; \sigma) |\ \sigma \text{ is an assignment}\}$. Thus a probability density on the space of assignments is given as

$$\pi(e, w, \beta; \cdot) : \sigma \mapsto \frac{\phi(e, w, \beta; \sigma)}{\Phi(e, w, \beta)}. \tag{6.5}$$

From relation (6.3) it is evident that if the vertex weight w is null then the energy map is "even":

$$\forall \sigma : \text{assignment} \quad : \quad \eta(e, 0; \sigma) = \eta(e, 0; -\sigma). \tag{6.6}$$

For an assignment σ, its *support* is $\mathrm{Spt}\,(\sigma) = \{v \in V |\ \sigma(v) = +1\}$. A 2-*partition* of V is a collection of the form $\{U, V - U\}$, such that $U \subseteq V$. Clearly $\sigma \leftrightarrow \{\mathrm{Spt}\,(\sigma), V - \mathrm{Spt}\,(\sigma)\}$ is a bijective correspondence among assignments and 2-partitions of V.

For any set $U \subseteq V$, let

$$c(U) = \{e \in E |\ \mathrm{card}\,(e \cap U) = 1 \ \&\ \mathrm{card}\,(e \cap (V - U)) = 1\} \tag{6.7}$$

be the collection of edges with an extreme in U and the other in its complement. Since an assignment is a $\mathbb{S}$-valued map:

$$\begin{aligned} \forall \sigma : \text{assignment} \quad : \quad \eta(e, 0; \sigma) &= -\sum_{\{v_i, v_j\} \in E} e_{ij} + 2 \sum_{\{v_i, v_j\} \in c(\mathrm{Spt}(\sigma))} e_{ij} \\ &=: \eta_s(e; \mathrm{Spt}\,(\sigma)). \end{aligned} \tag{6.8}$$

Let us introduce the following problem:

Minimum weight cut
Instance: An edge weighting map e.
Solution: A ground state.

Clearly, this problem is equivalent to minimize the energy operator $\eta(e, 0; \cdot)$ as defined by (6.3), or equivalently to find a vertex set U which minimizes $\eta_s(e; U)$ as defined by (6.8).

A similar problem is the following:

Two dimensional magnetic field
Instance: A planar graph $\mathbf{G} = (V, E)$.
Solution: A ground state σ_0 for the operator

$$\sigma \mapsto \eta(-1, -1; \sigma) = \sum_{\{v_i, v_j\} \in E} \sigma(v_i)\sigma(v_j) + \sum_{v_k \in V} \sigma(v_k). \tag{6.9}$$

Theorem 8 (Barahona [10])**.** Two dimensional magnetic field *is an NP-complete problem.*

For edge and vertex weight e and w, we refer to the Ising spin model as the map:

$$\forall \sigma \in \text{assignment} : \eta(-e, -w; \sigma) = \sum_{\{v_i, v_j\} \in E} e_{ij}\sigma(v_i)\sigma(v_j) + \sum_{v_k \in V} w_k \sigma(v_k) \tag{6.10}$$

and for any subset $S \subseteq V$, let $\eta(-e, -w; \sigma, S) = \sum_{\{v_i, v_j\} \in E(\mathbf{G}[S])} e_{ij}\sigma(v_i)\sigma(v_j) + \sum_{v_k \in S} w_k \sigma(v_k)$ be the Ising spin model defined on $\mathbf{G}[S]$.

The Ising spin model can be solved using the Dynamic Programming approach by solving partial solutions at every node on a tree-decomposition, and by the Courcelle Theorem there exists an algorithm with polynomial time complexity that solve the Ising spin glass model over graph instances with bounded treewidth.

The Dynamic Programming solution is as follows:

Let $\mathbf{G} = (V, E)$ be a graph with edge and vertex weigths e and w. Let $(\mathbf{T} = (T, F), \mathcal{X})$ be a nice tree decomposition of $\mathbf{G}$ and assume that the treewidth of $(\mathbf{T}, \mathcal{X})$ is bounded by a constant k. We compute the tables C_t for all $t \in T$ in a bottom-up order. We proceed by cases according to the types of nodes on $\mathbf{T}$:

- *Start.* If t is a leaf node of $\mathbf{T}$, then $|X_t| = 1$ and $X_t = \{v_i\}$ for $v_i \in V$. The table C_t has only two entries: $\forall \tau \in \{0, 1\} : C_t(\{v_i\}, \tau) = \eta(-e, -w; \sigma(v_i) = \tau, \{v_i\})$.

- *Introduce.* If t is an introduce node with a child t_1, then $X_t = X_{t_1} \cup \{v_i\}$ for $v_i \in V$. For all $\tau \in \{0, 1\}^{|X_t|-1}$:

  1. For all $v_j \in X_{t_1}, b \in \{0, 1\}$: if $\{v_i, v_j\} \in E$ then $C_t(X_t, \tau \cdot b) = C_{t_1}(X_{t_1}, \tau) + e_{ij}\sigma(v_i)\sigma(v_j) + w_i\sigma(v_i)$, otherwise $C_t(X_t, \tau \cdot b) = C_{t_1}(X_{t_1}, \tau) + w_i\sigma(v_i)$.

- *Forget.* If t is a forget node with a child t_1, then $X_t = X_{t_1} - \{v_i\}$ for $v_i \in V$. For all $\tau \in \{0, 1\}^{|X_t|}$:

  1. $C_t(X_t, \tau) = \min\{C_t(X_{t_1}, \tau \cdot 0), C_t(X_{t_1}, \tau \cdot 1)\}$.

- *Join.* If t is a join node with children t_1, t_2 then $X_t = X_{t_1} = X_{t_2}$. For all $\tau \in \{0, 1\}^{|X_t|} : C_t(X_t, \tau) = C_{t_1}(X_{t_1}, \tau) + C_{t_2}(X_{t_2}, \tau) - \eta(-e, -w; \sigma(X_t) = \tau, \{X_t\})$.

It is easy to prove that the energy ground state of $\mathbf{G}$ can be obtained from the table C_{root}. A similar algorithm for the Ising spin model is the proposed in [9], where a conditional restriction is imposed on the set of nodes of the tree-decomposition.

6.3.6 Quantum Ising model

Let us consider the Pauli transforms $\sigma_x, \sigma_z : \mathbb{H}_1 \to \mathbb{H}_1$ whose matrices with respect to the canonical basis are

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad , \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{6.11}$$

Over the canonical basis, we have,

$$\forall \varepsilon \in Q : \quad \sigma_z(|\varepsilon\rangle) = \theta(\varepsilon)\ |\varepsilon\rangle\,, \tag{6.12}$$

where $\theta : Q \to \mathbb{S}$, $\varepsilon \mapsto 1 - 2\varepsilon$.

Let $n \in \mathbb{Z}^+$ be a positive integer and let $[\![0, n-1]\!] = \{0, \ldots, n-1\}$ be the initial segment of the natural numbers with n elements.

For any index $j \in [\![0, n-1]\!]$ let $\sigma_z^j = \bigotimes_{\nu=0}^{n-1} s_\nu : \mathbb{H}_n \to \mathbb{H}_n$, where $s_\nu = \sigma_z$ if $\nu = j$ and $s_\nu = \mathrm{Id}$ otherwise. In other words, σ_z^j applies the transform σ_z at the j-th qubit of any n-quregister in $\mathbb{H}_n$. Then, as in (6.12):

$$\forall j \in [\![0, n-1]\!],\ \varepsilon \in Q^n : \quad \sigma_z^j(|\varepsilon\rangle) = \theta(\varepsilon_j)\ |\varepsilon\rangle\,, \tag{6.13}$$

Let $\mathbf{G} = (V, E)$ be a graph whose vertices are the first n indexes, $V = [\![0, n-1]\!]$, and $E \subseteq [\![0, n-1]\!]^{(2)}$ is a set of index pairs.

For any vertex weighting map $w : V \to \mathbb{R}$ let us consider the operator

$$H_w : \mathbb{H}_n \to \mathbb{H}_n\ ,\ H_w = \sum_{j=0}^{n-1} w_j \sigma_z^j. \tag{6.14}$$

From (6.13) we have

$$\forall \varepsilon \in Q^n : \quad H_w(|\varepsilon\rangle) = \left(\sum_{j=0}^{n-1} w_j \theta(\varepsilon_j) \right) |\varepsilon\rangle \tag{6.15}$$

hence, H_w is a diagonal operator.

Similarly, for any edge weighting map $e : E \to \mathbb{R}$, let us consider the operator

$$H_e : \mathbb{H}_n \to \mathbb{H}_n\ ,\ H_e = \sum_{\{i,j\} \in E} e_{ij}\, \sigma_z^i \circ \sigma_z^j. \tag{6.16}$$

Since the operators σ_z^j are pairwise commutative, again from (6.13) we have

$$\forall \varepsilon \in Q^n : \quad H_e(|\varepsilon\rangle) = \left(\sum_{\{i,j\} \in E} e_{ij} \theta(\varepsilon_i) \theta(\varepsilon_j) \right) |\varepsilon\rangle \tag{6.17}$$

hence, H_e is as well a diagonal operator.

As in eq. (6.3) let us define the operator

$$H(e,w;\cdot):\mathbb{H}_n\to\mathbb{H}_n\quad,\quad H(e,w;\cdot)=-H_e-H_w. \tag{6.18}$$

From (6.15) and (6.17) we have

$$\forall\varepsilon\in Q^n:\quad H(e,w;|\varepsilon\rangle)=\left(-\sum_{\{i,j\}\in E}e_{ij}\theta(\varepsilon_i)\theta(\varepsilon_j)-\sum_{j=0}^{n-1}w_j\theta(\varepsilon_j)\right)|\varepsilon\rangle \tag{6.19}$$

Between the greatest parenthesis, an energy map $\eta(e,w;\cdot):Q^n\to\mathbb{R}$ appears of the type of eq. (6.3), and a ground state $|\varepsilon_0\rangle$ of H corresponds naturally with a ground state ε_0 of $\eta(e,w;\cdot)$.

On the other hand, the Pauli transform σ_x (see (6.11)) has eigenvalues $+1,-1$ with respective eigenvectors $c_0=W|0\rangle$ and $c_1=W|1\rangle$, where W is the Hadamard transform. For any index $j\in[\![0,n-1]\!]$ let $\sigma_x^j=\bigotimes_{\nu=0}^{n-1}r_\nu:\mathbb{H}_n\to\mathbb{H}_n$, where $r_\nu=\sigma_x$ if $\nu=j$ and $r_\nu=\mathrm{Id}$ otherwise. In other words, σ_x^j applies the transform σ_x at the j-th qubit of any n-quregister in $\mathbb{H}_n$. Then, as in (6.12):

$$\forall j\in[\![0,n-1]\!],\ \varepsilon\in Q^n:\quad \sigma_x^j(c_\varepsilon)=\theta(\varepsilon_j)\,c_\varepsilon, \tag{6.20}$$

where $c_\varepsilon=\otimes_{i=0}^{n-1}c_{\varepsilon_i}$.

For any vertex weighting map $h:V\to\mathbb{R}$ let us introduce the operator

$$H_h:\mathbb{H}_n\to\mathbb{H}_n\quad,\quad H_h=\sum_{j=0}^{n-1}h_j\sigma_x^j.$$

From (6.20) we have

$$\forall\varepsilon\in Q^n:\quad H_h(c_\varepsilon)=\left(\sum_{j=0}^{n-1}h_j\theta(\varepsilon_j)\right)c_\varepsilon \tag{6.21}$$

hence, H_d is a diagonal operator, and a ground state has the form c_{ε_0} for $\varepsilon_0\in Q^n$ minimizing $\sum_{j=0}^{n-1}h_j\theta(\varepsilon_j)$, which in turn can easily be calculated depending on the map h.

Thus, the problem to find a ground state of the operator $H(e,w;\cdot)$ determined by (6.19) can be solved using the Adiabatic Theorem with the operator path:

$$H_t=\left(1-\frac{t}{T}\right)H_h+\frac{t}{T}H(e,w;\cdot)$$

for some large enough $T\in\mathbb{R}^+$.

Chapter 7

Conclusions and future work

We investigate the application of the adiabatic quantum computing to solve NP-hard problems. We have shown a procedural construction of the Hamiltonian operators for adiabatic computing for the MAX-SAT problem. This construction can be extended to describe the Hamiltonian operators of other NP-hard problems with similar structure.

We investigate the construction of local Hamiltonian operators for Adiabatic Quantum Computing. It is based on the dynamic programming approach and the monadic second order logic. We have shown results to modify an initial tree decomposition when new edges to the input graph are added. We have shown a general methodology to construct local Hamiltonian operators based on the Ising model in which the energy function minimization is equivalent to the optimization problem of pseudo Boolean functions. The Ising Hamiltonians have been used to design local Hamiltonian for the Adiabatic Quantum Computing (AQC), and they have a natural use to describe graph problems and other optimization problems.

We investigated the properties of the optimization problem of pseudo Boolean functions to construct local Hamiltonian operators. We consider the quadratic optimization problem, since every pseudo boolean optimization problem is equivalent to the quadratic case. Hence at present time we are involved in the decomposition of initial and ending Hamiltonians, arose from the AQC approach, into a sum of local Hamiltonians.

The future work of this thesis are the following: To improve the symbolic characterization provided in chapter 4of the AQC solution to many others combinatorial optimization problems. It is important to have a general characterization of the initial and final Hamiltonians for any NP-hard problems. In chapter 6 we have shown a general methodology to construct AQC algorithms, we have shown that every MSOL expression has associated quadratic pseudo-Boolean forms. It is important to classify optimization problems in terms of its representability, for instance according to the polynomial hierarchy.

The Dynamic Programming approach studied in chapter 6 has been considered in [9]. This approach depends on the possible physical implementation of quantum memories, the Dynamic Programming solution requires to store partial solutions.

This still remain as an open problem in the design of quantum algorithms.

Finally, a new tool in the design of quantum algorithms is the geometric Berry phase, these kind of algorithms are robust to some sources of errors. The Berry phase can encode the solution of search problems, it is important to find a general technique to encode the solution of optimization problems into the Berry phase.

Appendix

The following is a list of computer programs developed in this research. They have been used to analysis the properties of the AQC algorithms. The programming languages used were the Mathematica environment v8 and C++ on a Mac Pro computer with operating system v10.6., 16 GB of memory and 2 processors with 6 cores each at 2.4 GHz. The source code of programs can be downloaded from `https://computacion.cs.cinvestav.mx/~cwilliam`

1. `Adiabatic.nb`

   This notebook implements the AQC solution for the MAX-SAT problem. The input instance is an adjacency matrix of a graph and the procedure `hc2satList` reduce the graph problem into a SAT instance problem i.e., into a conjunctive normal form. The Hamiltonian construction proposed in [37] is implemented and the eigenpaths of the linear interpolation Hamiltonian are computed using the procedure `Eigensystem`. The number of vertices n of the graph instances that can be computed with our computer specifications was $n = 8$, and the number of Boolean variables of the SAT instances was $2n = 16$.

2. `EfficientAdiabatic.nb`

   This notebook implements the efficient AQC solution proposed in chapter 4 for the MAX-SAT problem. The SAT instances are generated randomly and all possible clauses with n variables are listed, this have been used to explore all possible AQC algorithms. Also, all clauses with n variables and m clauses have been listed. The efficient Hamiltonian construction compute the diagonal elements of the initial Hamiltonian rather than the matrix tensor products, and the final Hamiltonian is constructed recursively. The number of Boolean variables of the SAT instances used to simulate the AQC solution was $n = 16$.

3. `qboolAdiabatic.nb`

   This notebook implements the AQC optimization problem for quadratic pseudo-Boolean maps. The pseudo-Boolean maps are obtained using the reduction given in chapter 5 for the SAT problem (see Proposition 4). The procedure `Reduce` implements the reduction into quadratic forms of pseudo-Boolean maps. Hard instances with unique solution of the SAT problems are generated, and the Hamiltonian construction implemented in the notebook `EfficientAdiabatic.nb`

was used. The number of Boolean variables of the SAT instances used to simulate the AQC solution was $n = 16$.

4. `hamilsim.nb`

   This notebook implements the simulation of the adiabatic evolution by splitting the evolution time into small intervals. At each interval of time the matrix exponentiation of the total Hamiltonian is computed, given thus a unitary matrix. This simulation corresponds to the implementation of the AQC solution into the QCM proposed in [2].

   MatLab versions of the .nb notebooks were also implemented.

5. `treeDecomp` project

   This C++ project implements the tree-decomposition of graphs given in chapter 6. The nice tree-decomposition was implemented and all graphs instances were taken from the TSPLIB webpage. In order to display graphs and tree-decompositions, the GraphViz tool was used and a routine to format conversion from `.dgf` into `.gv` was implemented. All results and performance of our computer programs were compared with the project `treewidth.org` leaded by Bodlaender.

6. `dynProgram` project

   This C++ project implements the Dynamic Programming approach given in chapter 6. This project uses the tree-decomposition developed in the `treeDecomp` project, and the Dynamic Programming solution to the maximum Weight independent set was implemented. This implementation can be also be used without majors modifications to other problems. All results and performance of our computer programs were compared with the project `TSPLIB` project.

7. `plasmaAdiabatic` project

   An implementation for multicore architecture was also developed for AQC, the PLASMA code for high performance processing was used.

References

[1] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation. *SIAM J. Comput.*, 37:47–82, April 2007.

[2] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM J. Comput.*, 37:166–194, May 2007.

[3] Boris Altshuler, Hari Krovi, and Jérémie Roland. Adiabatic quantum optimization fails for random instances of np-complete problems. *CoRR*, abs/0908.2782, 2009.

[4] Stefan Arnborg. Decomposable structures, boolean function representations, and optimization. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science*, MFCS '95, pages 21–36, London, UK, UK, 1995. Springer-Verlag.

[5] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[6] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[7] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45:70–122, January 1998.

[8] Giorgio Ausiello, G. Gambosi, P. Crescenzi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[9] Nikhil Bansal, Sergey Bravyi, and Barbara M. Terhal. Classical approximation schemes for the ground-state energy of quantum and classical ising spin hamiltonians on planar graphs. *Quantum Information & Computation*, 9(7):701–720, 2009.

[10] Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.

[11] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26:11–20, 1997.

[12] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders. Efficient Quantum Algorithms for Simulating Sparse Hamiltonians. *Communications in Mathematical Physics*, 270:359–371, March 2007.

[13] Hans Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Prívara and Peter Ruđicka, editors, *Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin / Heidelberg, 1997. 10.1007/BFb0029946.

[14] Hans L. Bodlaender. Treewidth of graphs. In *Encyclopedia of Algorithms*. 2008.

[15] Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008.

[16] Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008.

[17] Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations ii. lower bounds. *Inf. Comput.*, 209(7):1103–1119, 2011.

[18] Hans L. Bodlaender and Arie M.C.A. Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259 – 275, 2010.

[19] R. B. Borie. Generation of polynomial-time algorithms for some optimization problems on tree-decomposable graphs. *Algorithmica*, 14:123–137, 1995. 10.1007/BF01293664.

[20] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Deterministic decomposition of recursive graph classes. *SIAM J. Discret. Math.*, 4:481–501, September 1991.

[21] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Appl. Math.*, 123:155–225, November 2002.

[22] Sergey Bravyi, David P. DiVincenzo, Roberto Oliveira, and Barbara M. Terhal. The complexity of stoquastic local hamiltonian problems. *Quantum Information & Computation*, 8(5):361–385, 2008.

[23] A. M. Childs. *Quantum information processing in continuous time. PhD thesis.* Massachusetts Institute of Technology, 2004.

[24] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Information Processing*, 7:193–209, 2008. 10.1007/s11128-008-0082-9.

[25] Vicky Choi. Different adiabatic quantum optimization algorithms. *Quantum Information & Computation*, 11(7&8):638–648, 2011.

[26] Vicky Choi. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing*, 10:343–353, 2011. 10.1007/s11128-010-0200-3.

[27] Bruno Courcelle. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2011.

[28] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

[29] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[30] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101(1-3):77–114, April 2000.

[31] Yves Crama and Peter L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2011.

[32] William Cruz-Santos and Guillermo Morales-Luna. On the hamiltonian operators for adiabatic quantum reduction of sat. In *LATA*, pages 239–248, 2010.

[33] William Cruz-Santos and Guillermo Morales-Luna. Guided evolution of tree decompositions of graphs. *International Journal of Computational and Applied Mathematics*, 7(1):13–24, 2012.

[34] L.-M. Duan, J. I. Cirac, and P. Zoller. Geometric Manipulation of Trapped Ions for Quantum Computation. *Science*, 292:1695–1697, 2001.

[35] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[36] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Daniel Nagaj. How to make the quantum adiabatic algorithm fail. *International Journal of Quantum Information*, 06, 2008.

[37] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106v1, 2000.

[38] Richard P. Feynman. Simulating Physics with Computers. *Int. J. Theor. Phys.*, 21(6/7):467–488, 1982.

[39] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. 1965.

[40] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, Published in US in May 2008.

[41] Martin Charles Golumbic and Udi Rotics. On the clique-width of some perfect graph classes. *Int. J. Found. Comput. Sci.*, 11(3):423–443, 2000.

[42] David J. Griffiths. *Introduction to Quantum Mechanics*. Pearson Prentice Hall, 2005.

[43] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[44] Pinar Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297 – 317, 2006. Minimal Separation and Minimal Triangulation.

[45] Dorit S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[46] Tad Hogg. Adiabatic quantum computing for random satisfiability problems. arXiv:quant-ph/0206059, 2002.

[47] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[48] Sorin Istrail. Statistical mechanics, three-dimensionality and np-completeness: I. universality of intracatability for the partition function of the ising model across non-planar surfaces (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 87–96, New York, NY, USA, 2000. ACM.

[49] Jonathan A. Jones, Vlatko Vedral, Artur Ekert, and Giuseppe Castagnoli. Geometric quantum computation using nuclear magnetic resonance. *Nature*, 403(6772):869–871, February 2000.

[50] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM J. Comput.*, 35(5):1070–1097, 2006.

[51] Julia Kempe and Oded Regev. 3-local Hamitonian is QMA-complete. *Quantum Information and Computation*, 3:258–264, May 2003.

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

[53] T. Kloks, H. Bodlaender, H. Müller, and D. Kratsch. Computing treewidth and minimum fill-in: All you need are the minimal separators. In Thomas Lengauer, editor, *Algorithms ESA' 93*, volume 726 of *Lecture Notes in Computer Science*, pages 260–271. Springer Berlin / Heidelberg, 1993.

[54] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[55] P.G. Kolaitis and M.N. Thakur. Logical definability of np optimization problems. *Information and Computation*, 115(2):321 – 353, 1994.

[56] A.E. Margolin, V.I. Strazhev, and A.Ya. Tregubovich. Geometric phases and quantum computations. *Physics Letters A*, 303(2Ľ$\hat{U}$Ò3):131 – 134, 2002.

[57] David Marker. *Introduction to model theory*. Graduate texts in mathematics. Springer, 2002.

[58] Albert Messiah. *Quantum mechanics*. Dover Publications, 1999.

[59] Assaf Natanzon, Ron Shamir, and Roded Sharan. A polynomial approximation algorithm for the minimum fill-in problem. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 41–47, New York, NY, USA, 1998. ACM.

[60] J. Nešetřil and P.O. de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Algorithms and Combinatorics. Springer, 2012.

[61] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 1 edition, October 2000.

[62] Roberto Oliveira and Barbara M. Terhal. The complexity of quantum spin systems on a two-dimensionalkitaev square lattice. *Quantum Information & Computation*, 8(10):900–924, 2010.

[63] Ognyan Oreshkov, Todd A. Brun, and Daniel A. Lidar. Fault-tolerant holonomic quantum computation. 2008.

[64] Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[65] Anargyros Papageorgiou and Chi Zhang. On the efficiency of quantum algorithms for hamiltonian simulation. *Quantum Information Processing*, 11:541–561, 2012. 10.1007/s11128-011-0263-9.

[66] Alejandro Perdomo, Colin Truncik, Ivan Tubert-Brohman, Geordie Rose, and Alán Aspuru-Guzik. Construction of model hamiltonians for adiabatic quantum computation and its application to finding low-energy conformations of lattice protein models. *Phys. Rev. A*, 78(1):012320, Jul 2008.

[67] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alan Aspuru-Guzik. Finding low-energy conformations of lattice protein models by quantum annealing. *Sci Rep*, 2:571, 2012.

[68] N. Robertson and P. D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Comb. Theory Ser. B*, 48:227–254, April 1990.

[69] N. Robertson and P. D. Seymour. Graph minors XIII: The disjoint path problem. *Journal of Combinatorial Theory(Series B)*, 63:65–110, 1995.

[70] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

[71] Neil Robertson and P D Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory Ser. B*, 41:92–114, August 1986.

[72] Neil Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153 – 190, 1991.

[73] Donald J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7(3-4):317 – 322, 1974.

[74] Donald J. Rose and R. Endre Tarjan. Algorithmic aspects of vertex elimination. In *Proceedings of seventh annual ACM symposium on Theory of computing*, STOC '75, pages 245–254, New York, NY, USA, 1975. ACM.

[75] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[76] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Washington, DC, USA, 1994. IEEE Computer Society.

[77] Erik Sjöqvist. A new phase in quantum computation. *Physics*, 1:35, Nov 2008.

[78] Luca Trevisan. Inapproximability of combinatorial optimization problems. *CoRR*, cs.CC/0409043, 2004.

[79] Avatar Tulsi. Adiabatic quantum computation with a one-dimensional projector hamiltonian. *Phys. Rev. A*, 80:052328, Nov 2009.

[80] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 279 – 287, oct. 2001.

[81] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *ESA*, pages 566–577, 2009.

[82] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[83] Pawel Wocjan and Thomas Beth. The 2-local Hamiltonian problem encompasses NP. `quant-ph/0301087`, 2003.

[84] L.-A. Wu, P. Zanardi, and D. A. Lidar. Holonomic quantum computation in decoherence-free subspaces. *Phys. Rev. Lett.*, 95:130501, Sep 2005.

[85] Mihalis Yannakakis. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79, 1981.

[86] Paolo Zanardi and Mario Rasetti. Holonomic quantum computation. *Physics Letters A*, 264(2–3):94 – 99, 1999.

[87] Chi Zhang, Zhaohui Wei, and Anargyros Papageorgiou. Adiabatic quantum counting by geometric phase estimation. *Quantum Information Processing*, 9:369–383, 2010. 10.1007/s11128-009-0132-y.

[88] M. Zimand. Weighted np optimization problems: Logical definability and approximation properties. *SIAM Journal on Computing*, 28(1):36–56, 1998.