POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

Molecular Unfolding with Quantum Annealing

Advisor: Prof. Gianluca Palermo
Co-Advisor: Dr. Riccardo Mengoni

Master Thesis of:
Kevin Mato, matricola 919805

Academic Year 2020-2021

Sommario

Il Molecular Docking è una fase importante del processo di scoperta dei farmaci che mira a predirre la posizione e la forma preferite da una molecola rispetto ad una seconda quando sono legate l'una all'altra.

Durante questa parte dell'analisi, chiamata in silico, una rappresentazione 3D delle molecole viene manipolata in base ai loro gradi di libertà con rototraslazioni rigide e rotazioni dei frammenti lungo i legami chimici ruotabili.

Nel nostro lavoro, ci siamo concentrati su una fase specifica della procedura di docking molecolare chiamata *Small Molecule Unfolding, SMU*, tradotto in italiano come dispiegamento della piccola molecola. Questa fase viene utilizzata per rimuovere il bias geometrico iniziale della molecola, tipicamente dovuto alla sua ricostruzione 3D, espandendone la forma nel rispetto dei legami chimici.

Abbiamo proposto per lo SMU un approccio basato sull'algoritmo del quantum annealing, formulando il problema di ottimizzazione come HUBO (High-order Unconstrained Binary Optimization), e poi lo abbiamo trasformato in QUBO (Quadratic Unconstrained Binary Optimization) per studiare come rendere la soluzione eseguibile sugli hardware più recenti di D-Wave (**D-Wave 2000Q e Advantage**).

Il problema è stato definito considerando la posizione dei legami chimici ruotabili della molecola come variabili di ottimizzazione. Abbiamo assunto dei valori di angoli discreti per le possibili rotazioni dei legami chimici. L'obiettivo del problema SMU è trovare la configurazione di angoli che massimizza il volume molecolare, o equivalentemente, che massimizza le distanze interne fra gli atomi che compongono la molecola.

Abstract

Molecular Docking is an important step of the drug discovery process which aims at calculating the preferred position and shape of one molecule to a second when they are bound to each other.

During this part of the analysis, called in-silico, a 3D representation of the molecules is manipulated according to their degrees of freedoms: rigid roto-translation, and fragment rotations along the rotatable bonds. In our work, we focused on a specific phase of the molecular docking procedure that is called *Small Molecule Unfolding, SMU*. This phase is used for removing the initial geometrical bias of the molecule, typically due to the 3D construction, by expanding the ligand to an unfolded shape.

We proposed a quantum annealing approach to SMU, by formulating the optimization problem as a HUBO (High-order Unconstrained Binary Optimization), and then we transformed it to QUBO (Quadratic Unconstrained Binary Optimization) in order to study make the solution feasible on the latest D-Wave hardware (**D-Wave 2000Q and Advantage**).

The problem has been defined considering the position of the rotatable bonds of the molecule as the optimization variables. We assumed discrete possible rotation angles. The objective of the SMU problem is to find the molecule configuration that maximizes the molecular volume, or equivalently, that maximizes the internal distances of the atoms that compose the molecule.

Contents

Ringraziamenti

Ringrazio il Professore Gianluca Palermo per le sfide, i consigli e le numerose opportunità che mi hanno permesso di crescere, e per avermi introdotto e guidato nel mondo della ricerca. Ringrazio il Dottor Riccardo Mengoni sia per aver trasformato il mio interesse per il Quantum Computing in una vera e propria passione, sia per il prezioso supporto quotidiano durante questi lunghi mesi di lavoro. Ringrazio tutti i miei amici e colleghi per aver reso gli anni dell'università indimenticabili. Infine, il ringraziamento più grande va alla mia famiglia, senza la quale nessuno dei miei traguardi ad oggi sarebbe stato possibile.

Chapter 1

Introduction

The in-silico approach to drug discovery can be viewed as a multitiered process or a pipeline that encompasses several sequential computational techniques with the aim of screening virtual libraries of the order of billions of compounds for the most suitable molecules to forward to later experiments. In-silico virtual screening often relies on the molecular docking phase.

Molecular docking is a computational technique used to simulate the interactions at the atomic level of a compound, called ligand, inside a protein's binding site, to highlight the possible biochemical reactions between them. From a geometrical point of view, it can be also seen as a method to calculate by means of rototranslations, the preferred position and shape of one molecule to a second when bound to each other. Since the evaluation of each molecule based on pharmacophoric features is computationally expensive, a geometric approach becomes a more viable method for sampling shape combinations.

In the process of docking we considered [45], we can identify three phases: i) the ligand expansion, ii) the initial placement, and iii) the shape refinement. In our work, we focused on the first phase, which takes the same time as the docking and does not belong to the actual procedure. This step cannot be skipped and must be performed for each molecule analyzed in the chemical database.

Problems arise because the shape in the initial pose of the substance in the binding site is chosen from a tool apriori, called *SMILE-to-3D*; the shape is reproducible but at first it is not useful because disconnected from any criteria. This is called a shape bias, that can be removed by expanding the ligand to an unfolded shape. Consequently, in this project we are dealing only with the study of the shape of the compound and not with its spatial location. The expansion of a molecule to its unfolded shape is an optimization problem where the variables are a subset of the chemical bonds that can be rotated around their axis. The search for particular conformations of the molecule has been treated in different works; in particular, the phase of expansion has been studied in the development of tunable geometric docking applications [18], where the ligand expansion was performed with a greedy approach, which creates and analyzes conformations by rotating in order one rotatable bond at a time, by a discrete angle. Other works [33] construct a methodology on the resolution of QUBO (Quadratic Unconstrained Binary

Optimization) problem formulations, with an objective function that aims at reducing the total energy of the molecule. In the literature, we can find approaches to the technique of molecular docking which makes use of particular devices called *quantum computers* [5].

Our model attempts to describe the expansion problem as the maximization of the volume of the molecule, or better expressed by the proxy measure of the total sum of interatomic distances. This measure of the volume is constructed as a function of the angles assumed by the rotatable bonds, and then after rewriting the values of the angles again as a function of the assignment of binary variables, with a one-hot encoding, the total of the internal distances is expressed as a HUBO (High-order Unconstrained Binary Optimization). Afterward, the HUBO is converted into a QUBO, by making use also of these three approximations: (i) the coarse-grained rotations, (ii) the landscape alternations by coefficients chopping, and (iii) the fragments contraction, also called the distances simplification. The three approximations allowed us to reduce the complexity of the model to embed and run all our problems' instances on two QPUs (Quantum Processing Units). The possibility of tuning the thresholds for these approximations in different sections of the implementation gives us the possibility of exploring the capabilities of this method. The project aims to understand if it is possible to improve the quality and the throughput of the ligand expansion phase, therefore we tried to leverage the potential of DWAVE's latest hardware, Advantage and 2000Q, to answer the question.

The thesis is organized as follows:

- In Chapter 2, we introduce the adiabatic quantum computing model, the functioning of devices that make use of the quantum annealing, called quantum annealers, and the problem description to be input to these devices. In the same chapter, the fundamental concepts on molecular docking are provided, together with our assumptions on the type of docking, and we concluded the chapter with examples of applications of quantum computing in medicine.

- In Chapter 3, the unfolding optimization problem is mathematically formalized and translated in the form of a binary optimization as a QUBO. This formulation is one of the central points for the work done.

- In Chapter 4, the whole pipeline for the resolution of the molecular problem is described in detail. We start by explaining how the cost function is extracted from the high-level description of a chemical substance, and we point out the assumptions, substitutions, approximations made for extracting a HUBO out of the initial cost function. Finally, in this chapter, we explain also how the quadratization of the HUBO works, and how we post-processed the results.

- In Chapter 5, we present the experimental results in applying the proposed technique. We introduce the experimental set-up we used in terms of dataset selection, the target machines, and the software we used. Afterward, we proceed by explaining the alternative search strategies to the quantum annealers that have been used

to compare the results, and the exploration performed during the embedding phase on the quantum devices.

- Chapter 6 concludes the work and describes some future extensions.
- Finally, the Appendix includes more plots and table of the results that have been kept out of the Experimental Result section, and that can be used to have a deeper look on the results we obtained.

Chapter 2

Background

In this chapter we discuss the background necessary for understanding the context, the objectives, the construction and finally the functioning of the project. In section 2.1 we are exposing an overview regarding the quantum computing model used, with an explanation of the quantum mechanical principles empowering it and an analysis of the algorithmic implementation of the model. In section 2.2 preliminary knowledge regarding the technique of molecular docking is illustrated, followed by innovative examples of quantum computing in drug discovery.

2.1 Adiabatic Quantum computing: an Overview

The purpose of this section is to familiarize ourselves with what concerns quantum computation, in particular the Adiabatic Quantum Computing (AQC) model. We will start in the first two subsections to understand the history and the reasons for the existence of this new technology; immediately after there will be a detailed exposition of the main elements, and quantum mechanical phenomena that make the model something unseen before, like: new concepts like the quantum bit or qubits, the operations that can be applied to these bits, and what kind of behaviour we can impose to a system of qubits, that we will define as Hamiltonians. In the subsection 2.1.4 we will see the theoretical algorithm of AQC, the Adiabatic Quantum Algorithm (AQA), how many steps it comprises, what operations and what are the requirements for being run. In section 2.1.5 the quantum annealing metaheuristic is introduced and compared to simulated annealing (SA), then the theoretical advantage of the algorithm over SA is proved. In section 2.1.6 we define what is the programmable element of the AQA, which is the problem's Hamiltonian H_p, and how it is used for implementing the algorithm of quantum annealing in terms of an AQA on a physical device. In 2.1.7 we formalize two methods of assessment for the performance of quantum annealing. In section 2.1.8 we define the mathematical representation used for constructing the Hamiltonian H_p and in 2.1.9 we highlight the differences between two QPUs.

2.1.1 Motivations

Moore's law has been known since the 1960s, which tells us that the number of transistors that we can introduce inside our processors doubles every approximately 18 months. It is easy to understand that doubling the elements on a finished surface implies halving the area of the elements themselves.

Unfortunately, the realization of ever smaller transistors has a limit, these technologies are reaching dimensions of the order of magnitude of nanometers. While the manufacture of individual components does not impose particular production limits, the construction of circuits and devices based on these components is not as easy.

It is becoming impractical given that the dimensions of the components introduce new behaviors and effects related to the world of quantum physics, such as the growth of noise signals due to the duality of particle and wave of atoms. We might ask ourselves why we need this trend in the evolution of technologies not to become stationary. The reason lies in the use we make of computational resources. Nowadays, to give some examples, many problems ranging from optimization, to artificial intelligence, from the research of new medicines to that of new materials, and the research on the origin of the matter are solved through the use of algorithms and computing systems.

The last three problems mentioned require the processing of large amounts of data, and in particular the last one would require practically infinite computational resources to be solved. If innovation in "classical" technologies slows down further, it will be necessary to introduce new technologies like quantum computers that will allow the solution of critical problems that would otherwise not be.

2.1.2 History

The first time that the idea of building quantum computers was proposed was by the physicist Richard Feynman during his time at Caltech [16]. His proposal was divided in two main points, the first was to get an efficient way to run simulations of quantum mechanics, which might be the most important application if we will get to see a completed device.

The second reason was to find a proof for his idea that quantum systems would be at least capable of universal classical computation, which was not sure in the early 80's.

Later on, David Deutsch [1] would formalize again the two objectives of quantum computing as a method for proving interpretations of quantum mechanics and to show that nature has the property of computational universality, it means that every physical process can be simulated by a universal computing device, which could be nature itself; David Deutsch is also the creator of the well-known *quantum gate model*.

We must highlight the fact that none of them were wondering if we can solve problems that cannot be solve with classical randomized algorithms in polynomial time with a "quantum polynomial time", which is the main question of contemporary quantum computing, actually NP-complete problems were not even mentioned!

Today quantum computers cannot solve NP-complete problems in polynomial time, and probably if a quantum algorithm that solves NP-complete problems existed at all,

as Prof. Umesh Vazirani reminds us, it would be different from anything we know today.

The great optimism and misconception comes from what is called *quantum speedup*, which is the idea that quantum algorithms might be able to leverage quantum phenomena in order to deliver time and space complexity asymptotically lower than the best classical algorithms. One example is Grover's algorithm [38] that can search a database in a $O(\sqrt{n})$ [1] time complexity, rather than $\Omega(n)$ [2] required by every classical algorithm. The speed-up is of order $\sqrt{n}$. Another example is Shor's algorithm [38], that can factor an n-bit number in polynomial time on a quantum computer with gate model, where the best classical algorithm takes superpolynomial time. The repercussions of this algorithm are that modern encryption methods like RSA are in danger since it is based on the complexity of factoring.

Adiabatic quantum computing is one of the alternatives to the gate model. The first main results came from Farhi et al. [15] and then by Aharonov et al.[2] that proved that the two models are polynomially equivalent. Since quantum computing is probabilistic, the main key for analyzing the goodness of execution is the trade-off between time-to-solution [3] and quality of the result.

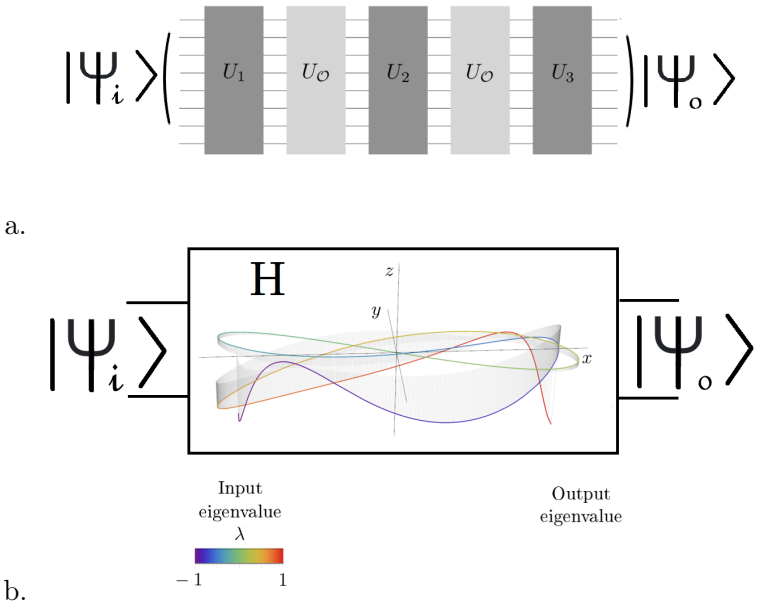

Figure 2.1: Intuitive representations of the gate model (a) that possesses a discrete nature, and of AQC (b) that is one continuous analog operation. Bottom image from [29].

[1] $O(g(n))$ = { $f(n)$: there exist constants $c > 0, n_0 > 0$ such that $0 \leq f(n) \leq cg(n), \forall n \geq n_0$.}
[2] $\Omega(g(n))$ = { $f(n)$: there exist constants $c > 0, n_0 > 0$ such that $0 \leq cg(n) \leq f(n), \forall n \geq n_0$.}
[3] explained at the end of section 2.1.7

2.1.3 Basics of Quantum Mechanics and Computation

In this section we are going to explain the basic unit of information of quantum computers, the quantum bit or qubit, then the operations we can perform on them, and what kind of behaviour we can impose to a system of qubits, that we call Hamiltonians.

Qubits

The first difference between classical and quantum computing is the representation of information. In classical computers, the state of a machine can be stored in a register that saves the value of N *bits*. The classical bit can be either 0 or 1, depending on its voltage. In a quantum computer, the state is still represented by a register, but this time it will contain N quantum bits, or *Qubits*. A qubit [42] is a quantum mechanical system described by a two-dimensional Hilbert space denoted by $\mathcal{H}$ and called qubit space. The information stored in a qubit is contained in the qubit state in $\mathcal{H}$ in which the system is, and it is manipulated and read according to the postulates of quantum mechanics. These qubits have properties of *superposition*, which means its state $|\psi\rangle$ (Dirac's bra-ket notation for a state) can be 0 or 1 at the same time .

$|\psi\rangle$ is a vector made of two complex numbers α and β. The coefficients α and β denote a linear combination of the two bases states $|0\rangle = \Big(1,0\Big)$ and $|1\rangle = \Big(0,1\Big)$ like this $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Unfortunately, the superposition cannot be observed directly. Instead, what we can do is measure the qubit according to a basis and its state will collapse to a vector, either $|0\rangle$ or $|1\rangle$.

The probability of measuring $|0\rangle$ is $|\alpha|^2$ (the squared magnitude of the complex number) and for $|1\rangle$ is $|\beta|^2$; since these two are probabilities their sum must be $|\alpha|^2+|\beta|^2 = 1$.

This property is the first point of one of the fundamental postulates of quantum mechanics, the **Born Rule**.

Notice that we could have situations where the magnitude of α could be much greater than that of β, so we will measure much more times $|0\rangle$ than $|1\rangle$ with a repetition of the experiment, which consists in resetting the qubit to the $|\psi\rangle$ state and then measuring again. It is important to understand that once we measure the qubit, its state has changed to ϕ ($|0\rangle$ or $|1\rangle$) .

In Figure 2.2 you can see a three-dimensional sphere of unitary radius, the *Bloch sphere*, where the state vector will be a point laying on the surface; the sphere is just a theoretical representation that doesn't exist in reality.

It is a usual convention to take the north pole as $|0\rangle$ and the south pole as $|1\rangle$, the two basis vectors. Pay attention that we may change the basis of reference and take the vector that corresponds to the orthogonal set of vectors laying on the x-axis, $|+\rangle$ and $|-\rangle$, where

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

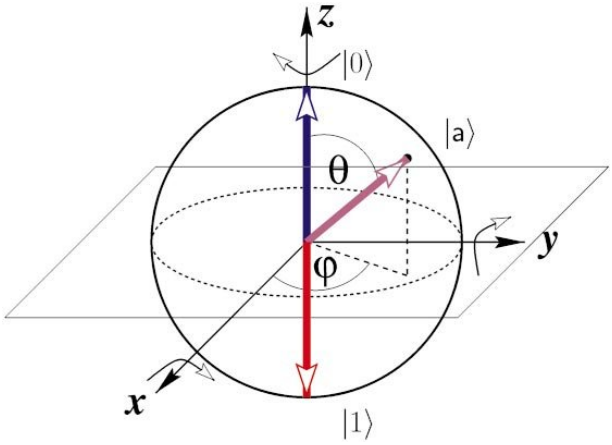

Figure 2.2: The Bloch sphere, a graphical representation [43] of the qubit.

If you square the coefficients of these last vectors, it is possible to see that the probability of measuring 1 or 0 is equal for both and it is 50%. These states are maybe the most important ingredient of quantum algorithms.

Quantum Operators

Operators in physics are functions that map a space of states to another one. Every observable quantity in the system, for instance, the spin of a particle, that we might want to measure, is represented by a self-adjoint operator O on that space. If one builds a device to retrieve the measure of the observable, the machine when used will output an eigenvalue λ of that observable. This was the second point of the Born Rule. In the case of a complex finite-dimensional Hilbert space (a vector space with an inner product), these functions are just matrices with particular properties.

Three fundamental operators are the *Pauli matrices* $\sigma^x, \sigma^y, \sigma^z$.

These matrices are what define the gates in the gate model and they have the physical meaning of performing rotations and symmetries of the states or dots on the Bloch sphere, with the particular property of reversibility.

For example, the Pauli-x will bit flip the the basis vector as a classical not operation.

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.1}$$

$$\sigma^y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} \tag{2.2}$$

$$\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.3}$$

Each Pauli matrix presents as eigenvalues -1 and +1, which will be easy to convert to 0 and 1 as our classical information. Let us see what are the respective eigenvectors.

For Pauli-z:

$$\sigma^z|0\rangle = +1|0\rangle$$
$$\sigma^z|1\rangle = -1|1\rangle$$

For Pauli-x:

$$\sigma^x|+\rangle = +1|+\rangle$$
$$\sigma^x|-\rangle = -1|-\rangle$$

One last thing to mention is the composition of multiple qubits. When we are performing a computation on multiple qubits, the state of the entire system is represented by the tensor product of the single state of the qubits.

For example, if we had a state $|\psi_1\rangle = \Big(\alpha_1, \beta_1\Big)$ and a second state $|\psi_2\rangle = \Big(\alpha_2, \beta_2\Big)$, the state space of the system would be [4]:

$$|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_s\rangle = \Big(\alpha_1\alpha_2, \alpha_1\beta_2, \beta_1\alpha_2, \beta_1\beta_2\Big)$$

The two qubit base will be: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ [5]. If we put all together:

$$|\psi_s\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle$$

Again, for the Born rule, the sum of the squared magnitude of the coefficients of the linear combination will be 1.

A system composed of N qubits could be in a state of *Entanglement*.

This is a nice property of quantum mechanics where if some qubits directly interact with the system, they are no longer independent from each other, and this dependency will be preserved even if the system is torn apart. Entaglement is preserved even if one of the interested qubits is subject to some operation, e.g. a Pauli-z operation, or a measurement.

Mathematically, the entaglement of N qubits results in a state $|\psi_E\rangle$ that cannot be written as a simple tensor product between all qubits anymore.

The most popular example is the *Bell state*

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

Hamiltonian

The evolution and dynamical behaviour of closed quantum systems is described by Schrodinger's equation:

[4] Please look yourself how to compute a tensor product

[5] these are the short notations for $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle$ etc...

$$i\hbar\frac{\partial|\psi(x,t)\rangle}{\partial t} = H(t)|\psi(x,t)\rangle \tag{2.4}$$

We know that $\hbar = h/2\pi$ and h is Planck's constant. The equation says that the instantaneous rate of change of the system's state is proportional to the energy of the system.

Usually $\hbar$ is put inside the H matrix. The matrix H is called *Hamiltonian* and it is an operator of great interest since it is the operator that represents the total energy of the system. Inside the square brackets of equation 2.5 you can see two contributions, the kinetic and potential energy function depending on the position of the particles and of time. The formula here presents a time-independent version of the Hamiltonian in eq. 2.4

$$i\frac{\partial\psi(x)}{\partial t} = \left[-\frac{1}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right]\psi(x) \tag{2.5}$$

In the case of an N-qubit time-varying system, the evolution is directed by the application of a physical action on it. Part of these actions are determined externally and part is the result of the interaction between qubits.

All these actions can be described by a time-varying Hamiltonian. Since the Hamiltonian is an operator in a finite dimensional Hilbert space, it is a square matrix of $N \times N$ and it is Hermitian with real-valued eigenvalues.

The Hamiltonian describes how the state of the system varies in time. For those who have a good familiarity with the gate model, it represents a circuit applied in continuous time rather than discrete time.

The fact that the matrix is *Hermitian* gives us particular properties of decomposition. For example, by applying the *Trotter formula* [38] to simplify the simulation of the matrix, we could decompose it in more unitary matrices to be applied in small intervals of time one after the other.

Another interesting property of the Hamiltonian is that it can be written as the sum of multiple local interactions, that will still be Hamiltonians themselves.

$$H = \sum_{n=1}^{N} H_n \tag{2.6}$$

Every single H_n will be a Hamiltonian that will act on a subportion of the system, it will be a one-body operation (on a single qubit) or it could be a two-body interaction [6].

The *Ising model* has this type of Hamiltonian, but it will be explained later.

This type of locality is reasonable since the spatial location of the particles in the system and their state decoherence determine also their interactions.

Since the observables of the system are all the possible combinations of $|q_1, \ldots q_N\rangle$ in terms on 0 and 1, every possible outcome will have an energy level corresponding to

[6] it means it operates as a function on two qubits

it. All possible energies will not be just +1 and -1 like before, but we will have a whole *energy spectrum*[15].

The instantaneous eigenvalues and eigenvectors of the Hamiltonian will be all the possible energies and states of the system, instantaneous because they will depend on how long the Hamiltonian will have been applied to the system.

We will have at most N real values $E_0(s) < \ldots < E_{N-1}(s)$ [7]. The lowest energy or eigenvalue will be the **ground state**, all the rest will be excited states.

2.1.4 Adiabatic Quantum Algorithm

Now we will explain the algorithm which is the basis for the Adiabatic Quantum Computing (AQC). AQC is an ideal quantum computational model, equivalent to others, like the quantum gate model, but it has an application which is solving optimization problems.

Given an objective function f that we must *minimize*, f maps the domain of binary variables to the real one, as $f : \{0,1\}^N \mapsto \mathbb{R}$. The function f usually represents a CSP (constraint satisfaction problem), and it can be divided into several components, for example a sum of constraints and a function g.

In order to run the Adiabatic Quantum Algorithm (AQA) [15] the following points are required:

1. The system has to start from an easily constructible state, which means we must know the ground state of the initial Hamiltonian H_B (beginning).

2. A time dependent Hamiltonian $H(t)$ based on the problem we need to solve.

3. A total evolution time T.

4. The evolution must execute according to Schrodinger's equation 2.4.

5. The final state of the system $|\psi(T)\rangle$, for a high enough T will be close to the ground state of the Hamiltonian H_p constructed from the problem instance; since the system will always be in ground state, we will be able to retrieve the optimal solution.

6. The algorithm finishes with a measurement of the N qubits on $Z_1 Z_2 \ldots Z_N$ [8]. If repeated many times, the device will return all the assignments related to the ground state [9].

Let us explain the various steps.

[7] s is the amount of time already passed

[8] measurement of the energy of each qubit on a space spanned by the basis of the eigenvectors of Pauli-z

[9] In non-ideal versions of this model, it's possible that many solutions will be invalid, even though they minimized our function f

The general form of H_B is defined as $\Gamma(t)\sum_{i=1}^{N}\sigma_i^x$ [37], and the choice of the coefficient g_i can be done arbitrary, but one choice could be

$$H_B = \sum_{i=1}^{N} \frac{1-\sigma_i^x}{2} \tag{2.7}$$

called the *driver Hamiltonian* [20]. We should notice that the Pauli-x operator is the main ingredient and it has an i-index that is due to the property of composition of the Hamiltonian. Each Pauli-x will act on the single i-th qubit of the system. A system that is left in a quiet state for long enough, tends to settle into a conformation that allows it to have the lowest possible energy according to its Hamiltonian.

By applying H_B after some time the system will try to reach its ground state. The lowest energy it could reach is 0, and to reach this each qubit will settle in a $|+\rangle$ state, because the eigenvalue of plus is 1, then by filling the formula $\frac{1-1}{2} = 0$.

If you look carefully the choice for H_B then isn't so arbitrary, because by the end of the evolution each of the qubits will be in a plus state, that is the superposition of $|0\rangle$ and $|1\rangle$ with equal probability of measurement. This is a convenient and neutral starting point for a computation. The construction of H_P is much less trivial than H_B, later on a possible choice will be proposed.

The time-dependent Hamiltonian $H(t)$ has to be a convex linear combination [20] of the form:

$$H(t) = (1-s(t))H_B + s(t)H_P \tag{2.8}$$

where $s(t)$ describes the evolution path or *schedule*.

If the schedule is $s:\ [0,1] \mapsto [0,1]$ and belongs to C^2, with first derivative $s' > 0$, and guaranteed that $s(0) = 0$ `and` $s(1) = 1$, then the function s is called the **adiabatic schedule** in the adiabatic algorithm .

The simplest function is a linear transition from 0 to 1, as

$$s(t) = \frac{t-t_i}{T} \ \texttt{where}\ T = t_f - t_i$$

.

The evolution time T must be chosen carefully.

According to the **adiabatic theorem**, if the the gap between the two lowest energy levels of the spectrum of the Hamiltonian, the difference $E_1(s) - E_0(s)$, is strictly greater than zero for all the duration of the schedule, then the nonzero gap will guarantee that $|\psi_S(t)\rangle$ obeying 2.4 will remain very close to the instantaneous ground state of $H(t)$ for t going from the initial time t_i to the final time t_f. A better definition of the minimal gap is *spectral gap* and it is mathematically defined as:

$$g_{min} = \min_{0\leq s\leq 1}(E_1(s) - E_0(s)) \tag{2.9}$$

The adiabatic theorem together with these considerations will translate in a more specific consideration. Here I write a simplified version that has similar meaning to the one you can find on [15]:

$$T \gg \frac{\left\| \frac{\partial}{\partial s} H(s) \right\|_{L2}}{g_{min}^2} \tag{2.10}$$

In the original formula, the numerator is a number of the order of a typical eigenvalue of H and it is not that big, so the formula is dominated by the denominator.

Thanks to Figure 2.3 we can see how the eigenvalues of the Hamiltonian $H(t)$ are changing depending on the progression of s(t), so it is clear that the total evolution time must be set according to the change rate of the Hamiltonian 2.8 so that the gap term loses relevance in the formula 2.10.

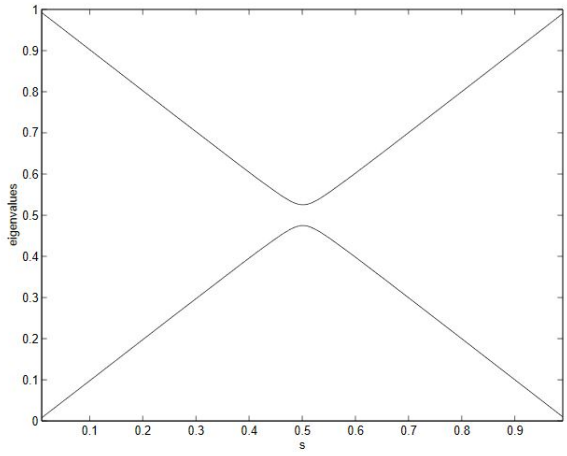

Figure 2.3: Examples of transitions of the eigenvalues over time, given the minimal gap at mid-transition.

Since in reality a condition of complete adiabaticity is impossible, given that the initial state is the ground state, the spectral gap is strictly greater than zero, and that the process evolves slowly enough, then with high probability the evolution will end with a final state equal to the ground state of the Hamiltonian fully developed.

2.1.5 Quantum Annealing

Quantum annealing (QA) was born as a classical algorithm, a heuristic for addressing challenging combinatorial problems. At first, it was just a variation of the *Simulated Annealing (SA)* [23] algorithm that was supposed to classically simulate more phenomena related to the quantum dynamics of a system rather than just the thermal fluctuations and the thermal annealing of the systems.

This algorithm is successfully filling up the gap between Adiabatic Quantum Computing (AQC) and classical simulation; there's a parallelism between QA and AQA but they have some differences.
Since quantum annealing can be described in terms of the few steps of the AQA, there is a common misconception that they are the same algorithm, but we could refer to quantum annealing as an implementation of the AQA restricted to a category of problems which is the NP-Hard, while the AQA is designed for tackling the whole spectrum of problems computable by a Turing machine; moreover since quantum annealing does not respect the adiabaticy condition, we can say that it is not dealing with closed systems anymore, hence the evolution of the system will be described more by a master equation rather than the Schrodinger one.

Quantum annealing can be referred as a "less ideal AQA" since it usually performs a probabilistic computation bounded by the limits of the adiabatic theorem for both runtime and quality of results, thus the interchangeability of the two concepts.

From now on we will keep this convention of terms.

Quantum annealing has been implemented on quantum computers with *superconductive qubits* by the company **D-WAVE**. Since quantum computers are *open systems* that cannot guarantee adiabaticity, then also the implementation will not respect the condition for the AQA, so the algorithm becomes an heuristic.

A comparison with SA

Let us describe how the QA and SA differ from each other, even though they both are heuristics. We will start by describing how the problem is described in this type of algorithms. Given a minimization problem and its relative function f, the domain composed by $x = \{0,1\}^N$ should be feasible $\forall x$.

The solution space is a lattice of solutions. The function then will represent something similar to a landscape, as in figure 2.4, where the peaks are its points of maximum and the valleys are the minimum points.

The search for the best solution is similar to a walk on the lattice; starting from an initial choice, we evaluate the value of the initial solution and we move to a neighbouring solution based on some rule (that is the main modus operandi for this type of algorithms), until we meet some requirements; in the case of a minimization problem, we move until reaching the lowest solution.

This is what is called a local search and the best landscapes are the "smoothest" because of the small number of local minima, but also huge "valleys" are not welcome since we would need powerful rules for distinguishing between neighbours with the same

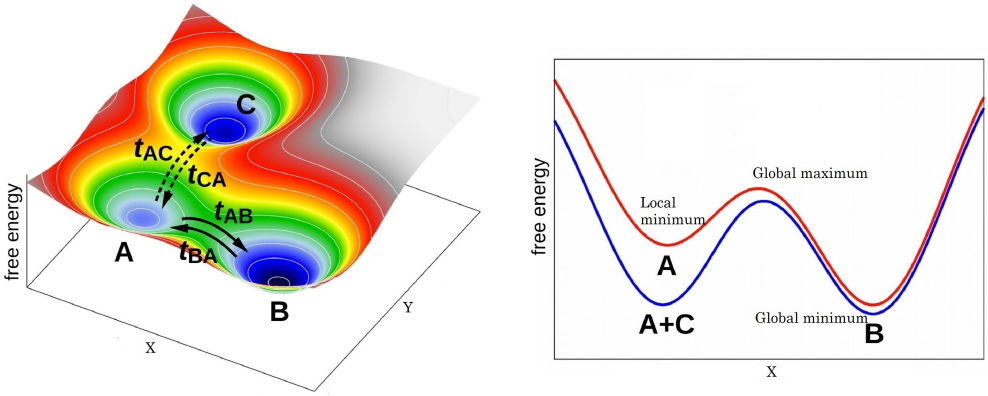

Figure 2.4: Example of Landscape. Image from [39]

evaluation. Simulated annealing starts with an initial guess $s_c = s_i$ with its evaluation $f(s_i)$, then the walk starts from there. If the neighbour has a lower evaluation, we take it as the current solution s_c.

If the neighbour has a higher evaluation, we accept it with a probability written as:

$$P_{accept} = \min(e^{-\frac{f(neigh.)-f(s_c)}{T}}, 1) \tag{2.11}$$

T is the temperature that starts from a high value, and it decrements in time based on the number of iterations, this is the "cooling schedule".

We talked about walking on the landscape since the mathematical model of S.A. is a random walk.

If we look at the formula if T is decreasing, when it will be close to zero, the probability density will increase around the global minimum, but the rate of convergence will become slow, this is happening because we want to quickly reach a steady state concentrated on good solutions.

The advantage of this algorithm is the adaptability to different implementations, the possibility of solving a problem with very little prior knowledge, almost like a black-box optimization, although domain-specific solvers can achieve better results. Instead of using temperature as a parameter for our optimization, in the classical original version of quantum annealing, it uses the quantum tunneling effect, which is the phenomenon where a particle instead of climbing up a potential barrier in the landscape, it just propagates through it, due to the dual wave/particle nature of quantum mechanics.

Quantum tunneling is controlled through *the Transverse field Hamiltonian* H_d [37], which is like a disordering operator.

$$H(t) = H_p + \Gamma(t)H_d \tag{2.12}$$

The gamma factor is the transverse field coefficient and it decreases during the simulation like the temperature before, this will slowly reduce the fluctuations in the solution space and bring the system closer to the problem's Hamiltonian.

H_d will be an Hamiltonian of just kinetic energy, and it will serve the purpose of escaping local minima.

The first connection point of quantum annealing (QA) with the AQC is that H_d is the initial Hamiltonian, and its high disordering property can be seen as a tentative of simulating classically the possibility of beginning the computation "from any state", logically we can fill this requirement in a quantum physical implementation by substituting H_d with the superposition Hamiltonian of the AQA. $\Gamma(t)$ will act as the schedule used in the AQA.

The problem's Hamiltonian that we haven't described yet can be the same for both the quantum physical version and the classical version, it just depends on what we want to simulate or compute.

Tunnelling

Although there is a proof [14] that the physical implementation of QA may outperform on some problems SA [3] and other classical implementations of QA, according to the latest discoveries, the physical implementation of quantum annealing on D-WAVE processors highlights the usefulness of the concept of quantum tunneling in this algorithm and how the physical use of quantum resources is relevant in this context. It is relevant to understand what is the computational value of finite range tunneling.

We can start from studying the exponential dependence of the annealing time on the size of the tunnelling domain D $T_{QA} = B_{QA}e^{\alpha D}$, where α is a constant representing a rescaled instanton action [10].

In SA the time is $T_{SA} = B_{SA}e^{\frac{\Delta E}{k_B T}}$, where ΔE is the energy difference between two points in the landscape, or distance between two minima (k_B is the Boltzmann constant).

$$\frac{\Delta E}{k_B T} > \alpha D$$

It is clear that for tall and narrow enough energy barriers, QA can terminate with exponential speed-up. This advantage can be found in the beginning of the algorithm when the initial state is more likely to be an excited state of the spectrum, and it can help in finding approximate solutions quickly, but it does not necessarily work steadily as before when the objective is finding the ground state.

2.1.6 The Problem's Hamiltonian, H_P

Here we describe the last ingredient for Quantum Annealing and the Adiabatic Quantum Algorithm, that can be anything of our choice but restricted from the possibilities of physical implementations.

The Ising model is a mathematical model designated to describe ferromagnetism in statistical mechanics. It is composed of discrete variables that represent the magnetic dipole moments of atomic spins, that can be +1 or -1. The model places the particles

[10] in simplier terms a classical solution in the equation of motion in quantum mechanics

in a lattice, where each spin interacts with its neighbours. The lattice can be a grid of N dimensions and the spin configuration of the system is defined by the assignment of the spin of each particle. The model is a simplified version of reality that takes into account an external force h_i applied to each individual particle, or our qubits, and the interaction between the particles, called forces J_{ij}. If we deal with a 2D Ising model, two nonadjacent qubits will have $J_{ij} = 0$.

D-WAVE's quantum annealers are actually a chip designed for solving this problem, but it will be discussed later on.

Ising's Hamiltonian is the operator that will define the energy of the overall system depending on these relations:

$$\mathcal{E}(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j \tag{2.13}$$

The values of h and J have a practical meaning, since the spins [11] configuration must reflect the lowest energy state possible, if h_i is positive, we can minimize the energy of that component by having spin +1, if the negative spin must be -1.

Another observation is that if J_{ij} is negative (ferromagnetic coupling), the spin of two neighbours must be equal to agree in sign, if positive, they will have to disagree (antiferromagnetic coupling). Both h_s and J_s can be imposed by the scientist in reality to artificially reproduce the configuration of spins.

Finding the optimal configuration for determining the ground state of this Hamiltonian in a multidimensional model is an NP-Hard problem, but the 2D version still can lead to results in decent time. It is important to highlight that quantum coupling is an effect in quantum mechanics where two or more quantum systems are connected in a way such that a change of state in one of the systems will cause an instantaneous change in all the others; it is different from the entanglement effect, since it can't take place over a long distance, but on the quantum annealers they are not mutually exclusive, as proved in [26].

Let us assume that there is an ensemble of N qubits, each of them capable of interacting with a subset of others, depending on their positioning and coupling. Given that the system in the adiabatic quantum algorithm starts from a state of superposition with Hamiltonian H_B, H_p will be:

$$H_p = \sum_i h_i \sigma_i^z + \sum_{(i,j)} J_{ij} \sigma_i^z \sigma_j^z \tag{2.14}$$

where the index i represents the single qubit, and the coupling i,j are the ferromagnetic interactions between two qubits. Pay attention that this time we are not dealing directly with spins anymore, but there are Pauli-z operators, it means that to minimize the energy of the system, the qubits will take spin according to the eigenvectors and eigenvalues of the operator. It is easy to convert the energy measurements to 0s and 1s, thanks to the trick of the Boolean Fourier transform :

[11] s in this case is the variable defining the spin and not the schedule

$$\sum_i h_i(-1)^{q_i} + \sum_{(i,j)} J_{ij}(-1)^{q_i}(-1)^{q_j} \tag{2.15}$$

We can derive the values of q_i and q_j that are the Boolean values that the i-th or j-th qubit is supposed to represent after its energy measurement.

2.1.7 Runtime analysis

The runtime of an AQA is very dependent on the type of problem that will influence the construction of $H(t)$. Various studies have been made in order to give some method for performing any type of analysis of the complexity of the algorithm and of the problem instance. For instance, in SAT problems [15] the evaluation is theoretical since the construction of the Hamiltonian for a 3-SAT problem is still feasible for a "pen-and-paper" proof. For more general CSP formulations, [44] is a quite exhaustive methodological discussion. Our project falls in the category of CSP problems.

Venturelli's [44] work defines a few steps that we can apply to the more practical implementation of the AQA, *Quantum Annealing* that runs on a system with related measuring equipment called *Quantum Annealer*, we will delve on its details in another section and from now on we will address the algorithm with the name of the device for simplicity.

The analysis goes from a first identification of what parameters span the solution space for our problem, to *pre-characterize* the families of instances with their minimal computational cost. This implies determining a distribution of execution times and a statistical distribution for the optimal results. Once the parameter space is defined, we can continue by planning an ensemble of queries $\boldsymbol{Q} = q$ to the quantum annealer.

Since the adiabatic algorithm in reality starts to become more of a theoretical model, every evolution does not really end in the ground state. We will need several executions to collect reasonable data. Every query is defined by a triple (t_A, R, T). R indicates the number of identical anneals, or evolutions, for the Hamiltonian forced on the system, with each annealing time as t_A.

Now we may be wondering how many queries are enough, but before this question we need formal assumptions. For each query, the number of anneals should be high enough to statistically draw conclusions at each run, and we need to create conditions for a *generalized adiabaticity* during the evolution, our previous T will be what we called t_A, so t_A will be optimized over the total maximal execution time $(t_A * R)$, to be long enough for performing anneals that will end up in the ground state.

The choices for these assumptions can be made through intensive tuning that requires resources in terms of implementation and time that are beyond the possibilities of the state-of-art devices.

What one can do is just a process of trial and error of these parameters until good enough results are found. If one still wishes to drive his search through a less pragmatic approach, we could set two variables r_0 and r_q, that represent respectively the target success probability for the queries and the rate of occurrence of the ground state per repetition for the following query.

Then we can define R and total time to solution T as:

$$R = \frac{\log\left[1 - r_0\right]}{\log\left[1 - r_q\right]} \tag{2.16}$$

$$T = \sum_{q \in Q} t_A \left(\frac{\log\left[1 - r_0\right]}{\log\left[1 - r_q\right]} \right) \tag{2.17}$$

Considering the large variability in the intrinsic nature of each problem, the number of variables, and their influence on the construction of the Hamiltonian, it becomes convenient sticking to the trial and error process. It is possible to moderately increase the total execution time for improving the probability of success.

An interesting index of quality for the executions of the quantum annealing and simulated annealing algorithms is the *Time-to-solution* [3] or TTS [22].

The time-to-solution equation is:

$$TTS = \frac{time\ per\ anneal}{ground\ state\ probability} \tag{2.18}$$

but it is equivalent to calculate it as:

$$TTS = \frac{total\ execution\ time}{occurrence\ of\ the\ best\ solution} \tag{2.19}$$

The total execution time can be an upperbound for the number of anneals for an execution, times the duration set for one anneal, while the occurrence of the best solution is the probability of finding the ground state times the number of anneals set for an execution. If we put together multiples runs we can calculate the total time-to-solution of an experiment, as the ratio between the cumulative sum of the runtimes and the sum of the occurrences of the best solution. This measure will be useful during the analysis of the results, for figuring out the differences, the pros and the cons, of the two types of annealing discussed so far. We can see some similarity between the TTS and the T in equation 2.17.

2.1.8 Quadratic Unconstrained Binary Optimization

From now, due to the duality between the Ising model and the Quadratic Unconstrained Binary Optimization problems (QUBO), it will be important to understand how to derive them the latter, and how to physically express it, or embed it, on a chip. The following paragraphs will explain how to mathematically model a problem and turn it in the problem Hamiltonian H_p that will be executed on the quantum annealer.

The Ising model represents two categories of problems: quadratic unconstrained binary optimizations (QUBO) and weighted maximum 2-SAT problems [8]. We will deal with the the first of the two.

The QUBO representation can be used for many types of optimization problems, mostly combinatorial, and it is a conceptual link between the physical experimentation and the realization of quantum annealing, today available and implemented in D-WAVE's chips, and the general problems of optimization.

The QUBO model is expressed by the optimization problem:

$$\min f(x) = x^t Q x \tag{2.20}$$

Or more formally:

$$\{0,1\} \mapsto \mathbb{R} \tag{2.21}$$

$$(x_0, \ldots, x_{n-1}) \mapsto \sum_{i,j=0}^{n-1} x_i Q_{ij} x_j \tag{2.22}$$

where x is a vector of binary variables and Q is a square matrix $n \times n$ of constants. The Q matrix can be symmetric or upper triangular.
If we want to use the symmetric form we should use:

$$\begin{cases} q_{ij} = (h_i), \forall i, \forall j, i = j \\ q_{ij} = \frac{(J_{ij}+J_{ji})}{2}, \forall i, \forall j, i \neq j \end{cases}$$

Meanwhile the upper triangular form:

$$\begin{cases} q_{ij} = (h_i), \forall i, \forall j, i = j \\ q_{ij} = (J_{ij} + J_{ji}), \forall i, \forall j, i > j \\ q_{ij} = 0, \forall i, \forall j, i < j \end{cases}$$

Notice that we used J_{ij} and J_{ji} with a little abuse of knowledge on the equivalence between QUBO and Ising models to distinguish the entry in the Q matrix from the coefficients that it represents.

Penalties

The definition that we just gave was that of an objective function minimization without any constraint on the variables. This is very unrealistic since most of the interesting problems impose constrains on solutions.

Many problems can be formulated in QUBO form, by introducing quadratic penalties in the objective function, alternatively to filtering the solutions.

The solver will adapt its search by avoiding invalid solutions for the penalties injected. A quadratic penalty is usually written as a function that is null for valid solutions and positive for invalid ones.

For a minimization problem, these penalties, as described in [19], are added to create an augmented objective function to be minimized. If the penalty term can be driven to zero, the augmented objective function becomes the original function to be minimized. When we add the penalty, since it is composed of just variables that are binary, this component of the objective function is multiplied by a scalar constant A that must be sufficiently large enough as it needs to be of the same order of the optimization component, or contribution.

Here are some of the most used examples:

Constraint	Quadratic Constraint
$x + y \leq 1$	$A(xy)$
$x + y \geq 1$	$A(1 - x - y + xy)$
$x + y = 1$	$A(1 - x - y + 2xy)$
$x \leq y$	$A(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$A(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$A(x + y - 2xy)$

The only limit for creating custom quadratic constraints is our ability in simplifying and solving the squared form of classical constraints, which is the most generic method up to today; another difficulty is also finding the right scalar constant A, such that should not be too high or too low, in order to be returned with valuable results.

Example of QUBO construction

An example can be the "Minimum Vertex Cover" problem.

Given an undirected graph with a vertex set V and an edge set E, a vertex cover is a subset of vertices such that each edge in the graph is incident to at least one vertex in the subset. The Minimum Vertex Cover problem seeks to find a cover with a minimum number of vertices in the subset.

Given that:

$$\begin{cases} x_j = 1, \ x_j \in Cover \\ x_j = 0, \ otherwise \end{cases}$$

Then the problem becomes:

$$\min \sum_{j \ inV} x_j$$

with constraint:

$$x + y \geq 1, \ \forall (i,j) \in E$$

By augmenting the objective function with the quadratic penalty component, we get the unconstrained objective function:

$$\min f(x) = \sum_{j \in V} x_j + A(\sum_{(i,j) \in E} 1 - x_i - x_j + x_i x_j \ ) \tag{2.23}$$

When constructing the matrix Q, the constant in the expression is ignored.

Here's an example of a graph with 5 vertices and 6 edges.

The instance's expression to minimize, derived according to the generic rule 2.23, is

$$\begin{aligned} \min \ y = &(1 - 2A)x_1 + (1 - 2A)x_2 + (1 - 3A)x_3 + (1 - 3A)x_4 + \\ &(1 - 2A)x_5 + Ax_1x_2 + Ax_1x_3 + Ax_2x_4 + Ax_3x_4 + Ax_3x_5 \\ &+ Ax_4x_5 + 6A \end{aligned} \tag{2.24}$$

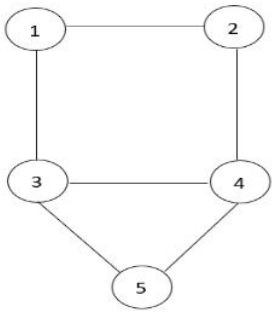

Figure 2.5: Example graph for MVC

By arbitrarily choosing A equal to 8, we can rewrite the expression as the QUBO matrix of the problem to solve.

$$Q = \begin{pmatrix} -15 & 4 & 4 & 0 & 0 \\ 4 & -15 & 0 & 4 & 0 \\ 4 & 0 & -23 & 4 & 4 \\ 0 & 4 & 4 & -23 & 4 \\ 0 & 0 & 4 & 4 & -15 \end{pmatrix} \tag{2.25}$$

The solver will take it from here and give us back the results. You can check yourself that y = 3 for x = (0,1,1,0,1) meaning that the minimum cover is given by nodes 2, 3, and 5.

2.1.9 Quantum Architectures

The D-WAVE quantum processing unit (QPU) is a lattice of interconnected qubits. Although some qubits connect to others via couplers, the D-WAVE QPU is not fully connected. In the earlier generations like D-WAVE 2000Q, qubits in the QPU are interconnected in a topology known as Chimera, while the latest quantum annealers, like Advantage QPU, incorporate the Pegasus topology. The topologies and the expression of the QUBOs on these architectures are the topic of the next paragraphs.

Minor-embedding

The process of "compiling" our problem to a QPU is called *embedding*. The process of embedding consists in a probabilistic algorithm that is trying to map our problem's graph to the one of a topology. This may lead to some new problems.

The first one may be that some small number of qubits and couplers in a QPU may not work or be manufactured as desired. These are therefore are removed from the programmable fabric that users can access.

Some other times, the graph representation of a QUBO may be too complex or present a very much different structure to the one of the topology of the QPU utilized, which may present a connectivity between the physical qubits, that does not match the

one of the logical variables, or at least it does with great difficulty. The solution found is chaining.

A chain is when a logical variable is represented during the run of an anneal by more than just one physical qubit. Chaining qubits is done by setting the strength of their connecting couplers negative enough to strongly correlate the states of the chained qubits; if at the end of most anneals these qubits are in the same classical state, representing the same binary value in the objective function, they are in effect acting as a single variable; most of the time this does not happen, moreover very long chain require setting the couplers to highly negative values that due to the limited the precision of the device, will affect the scaling of the other coefficients.

The minor embedding algorithm [10] aims at mapping the QUBO to a "minor" which is any graph that we can construct by contracting or removing an edge of the programmable fabric. It would be easy if the minor would be always the same, but since different devices have a unique QPUs, the algorithm must be probabilistic to achieve a certain degree of generality, that is why we should always reiterate several times the algorithm. It is of major importance to keep in mind the fact that logically equivalent embeddings have different energy spectra, thus different performances, therefore the need of performing this phase multiple times.

There are many versions of this algorithm, depending on the topology, but they try to analyze the QUBO and then decide how to tackle the task, for example two different approaches could be either performing a *globally constructed* or *locally constructed* embedding, similar to an incremental bottom-up construction.

Topologies

The qubits in a topology are grouped in unit cells and they are connected between each other by couplers that are categorized in internal and external just for starter.

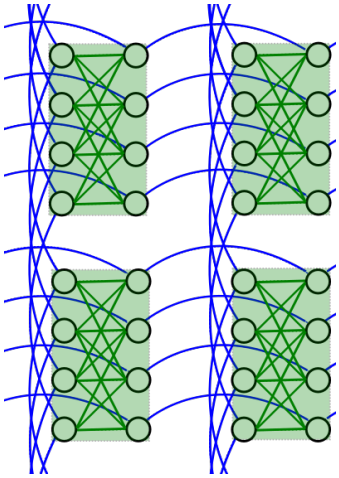

Figure 2.6: The K4,4 unit cell of the Chimera topology.

In the chimera topology, the unit cell is called $K_{4,4}$, it contains 8 qubits, and in figure 2.6 we can see that they are divided in two equal groups, where each qubit is

connected to 4 orthogonal qubits through the green internal couplers, and each one of them has two external couplers, the blue ones, that connect them to each particle with their counterpart in the neighbouring unit cells.

The qubits in the group on the left are connected with the neighbours along the vertical axis, the qubits in the group on the right are connected with the closest unit cells along the horizontal axis. The D-WAVE 2000Q has 2048 qubits divided in a grid of 16X16 unit cells. The notation for this topology will be be "C16".

The Pegasus topology [9] [13] is an evolution of Chimera, and it contains circa 5000 qubits. It has unit cells of twenty-four qubits ($K_{12,12}$), 12 qubits that are connected to 12 orthogonal qubits through internal couplers, and each qubit is coupled to 15 different qubits.

The connectivity between the unit cells has also been improved by introducing another type of coupler which can be called "odd". For the sake of simplicity, we can think of it as adding a third dimension to the topology, since they connect an element of a cell to cells that may not be direct neighbours; this allows stacking more instances of the same topology on one chip. This idea not only brings new connectivity, but it might be used in error-correction schemes that increase the energy scale for the embedding.

Differently from Chimera, the notation for this topology will be be "Pm", but the number of qubits is calculated as $24M(M-1)$, and for a more visual understanding of the complexity of this new architecture, in Figure 2.7 you can see an instance of a P4 Pegasus topology.

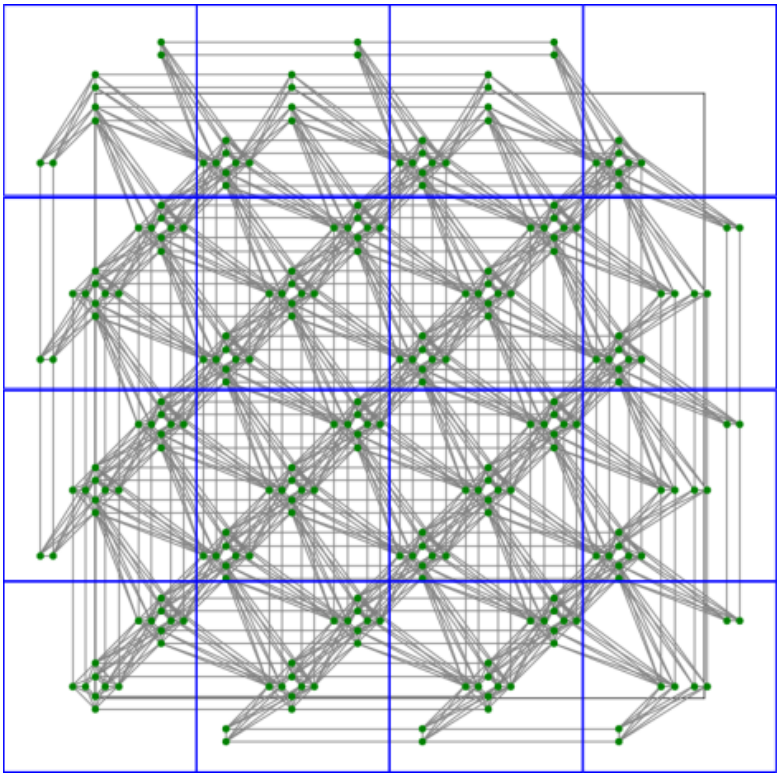

Figure 2.7: The P4 unit cells of the Pegasus topology.

2.2 Molecular Docking

Let us now introduce some basic concepts needed for understanding the target problem we faced by using quantum annealing. In section 2.2.1 We will understand the objectives of molecular docking, how the quality of the results is assessed, what types of docking exist and what are the main phases. In section 2.2.2 we can get to know more about the usage of quantum annealers in drug discovery and another example of quantum molecular docking.

2.2.1 Basics

In-silico approach to drug discovery [31] [28] can be viewed as a multitiered process or pipeline that encompasses several sequential computational techniques with the aim of screening virtual libraries of the order of billions of compounds for the most suitable molecules to forward to later experiments. This can be seen as an effort with the goal of cutting down the research timeline, which can be years and cost, by reducing wet-lab experimentation with computer modelling.

This virtual screening is driven by the structural characteristics of a target receptor, which can be a protein, and of each molecule taken individually and independently from each other from the database. After filtering part of the library and preparing the target binding site, a subprocess is involved, the so-called molecular docking.

Molecular docking [34] is an approach used to simulate the interactions on the atomic level of a compound, called *ligand*, inside a protein binding site, to highlight the possible biochemical reactions between them and predict whether they can form a stable complex.

This process is divided in two main tasks:

- the first is to detect the valid conformations of the ligand inside the active site of the protein, as well as its position and orientation, all together these will define what is called a three-dimensional *pose* of the ligand.

  The aim of this process is to achieve an optimized conformation for both receptor and ligand and their relative orientation such that the *free energy* of the overall system is minimized.

  A famous metaphor used for describing the conformation search is that of the "Lock and Key", where the binding site is the lock's hole and the search is mimicking the movement of the key. Due to the dimension of the binding site, the active region will be called *pocket*.

- The second task is to score and rank the poses found. The process is successful if the solutions are retrieved effectively and we used a scoring function that correctly ranks them. Usually the more negative is the value of the docking score, the better will be the *binding affinity* of the result. Clearly, high energy systems will be very unstable. Unfortunately, scoring functions involve estimating, rather than calculating the binding affinity between the protein and ligand, by adopting various assumptions and simplifications.

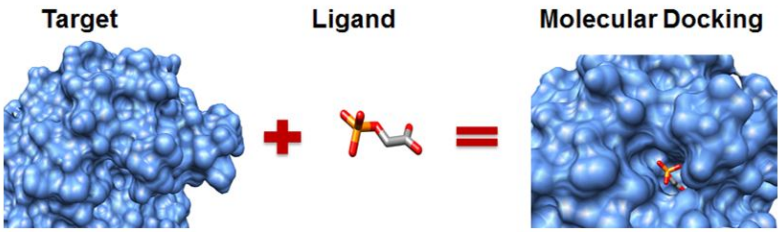

Figure 2.8: Insertion of the lingand in pocket, similar to the relation between locks and keys.

Scoring functions can be divided in macro areas, based on the fact whether they use physical features of the ligand and of the pocket, chemical features, or statistical knowledge of the interactions between classes of chemical compounds and some typical patterns in the shape of the pockets. There are geometrical scoring functions [25] [18] that rely on a measurement of the steric effect of an interaction.

Steric effects are phenomena observed when two atoms are spatially located close to each other, and this requires an energy cost. Since the electrons of the atoms are repulsive to each other, this interaction will change the shape of the molecule and its reactivity. Therefore, it is logical that if we can control the shape of the molecule, we can predict its reactivity to a protein's pocket and the energy cost of the shape. There are emerging scoring functions based on a sort of consensus [24] algorithm, that combines the outcomes of more scoring functions for more precise predictions.

The types of docking [34] are determined by some other particular assumptions. We can define three main types:

- **Rigid ligand and rigid receptor docking.** Its results are not really fascinating since the search space is very limited, considering only three translational and three rotational degrees of freedom.

- **Flexible ligand and rigid receptor docking.** Almost all docking programs adopt this type. Introducing a source of flexibility introduces a type of authenticity to the method while still pursuing a trade-off between accuracy and computational time.

- **Flexible ligand and flexible receptor docking.** This one is the most challenging type because of its high computational expense, which prevents this method from being used in the screening of large chemical databases.

In our approach, the docking considers the pocket a rigid structure, while the ligand is a flexible set of atoms. Furthermore, from a strictly geometrical interpretation, the ligand can also be seen as a set of chemical bonds (or edges) with a fixed length, where a pair of consecutive bonds is fixed at a determined angle and a subset of edges will be defined as "rotatable".

The rotatable bonds (or torsionals) are bonds that split the molecule in two nonempty disjointed fragments when virtually removed, consequently when the two portions of the compound are detected, they can rotate independently from each other around the axis of the rotatable. This first definition is graphically reported in Figure 2.9, where the rightmost rotatable bond is splitting the molecule in left and right fragment.

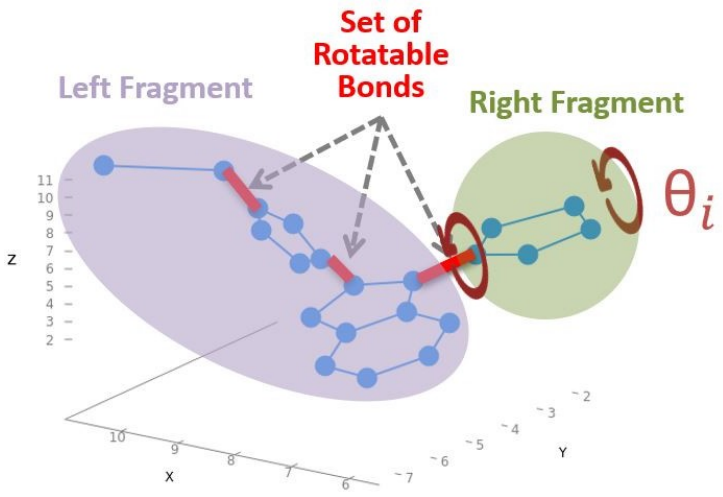

Figure 2.9: Fragments and rotatable bonds

A great number of degrees of freedom determine the solution space for the process as well as the conformation space of each molecule, in addition because the evaluation of each molecule based on pharmacophoric features is computationally expensive, a geometric approach becomes a more viable method for sampling a set of likely ligands that could fit the pocket with success and that later will validated for their chemical and physical usefulness.

The main difference relies on the evaluation of the pose: on the one hand, the geometric docking is scored as a function of the shape and volume of the molecule, and on the other hand the pharmacophoric docking scores a pose depending on its physicochemical properties.

The docking method related to our work is GeoDock [18]. We can point out three main phases in the search for the most fit compound, and we can call them: *Ligand expansion*, or better unfolding of the ligand molecule, *Intial Placement* and *Shape Refinement* inside the pocket.

The sampling of solutions through geometrical methods are susceptible to biases induced by various factors. One of the crucial biases is introduced before the ligand expansion, our mission becomes mitigating its effect.

Expansion for improving docking

The ligand expansion cannot be strictly associated with the docking process, since it solves a problem induced just by the preprocessing tool "SMILE-to-3D". The tool is

used for the translation of the structural information of a molecule, retrieved from a mol2 file into a two-dimensional smile, and then from the smile to the three-dimensional representation. The problem arises because the initial pose of the substance is chosen by the tool apriori, meaning it is reproducible but disconnected from any criteria, thus not useful. This can be defined as an example of shape bias that requires as much time as the docking itself for being removed and its fix is unskippable.

2.2.2 Combining Quantum Computing and Medicine?

There are examples of attempts to apply quantum computation to medicine. One example that we could relate to because of the usage of quantum annealers is the the work of Babej et al. on coarse-grained lattice protein folding [4]. The second is the study of a molecular docking implementation on a photonic quantum computer, which is completely different in every sense from our quantum annealers, but it still takes advantage of quantum effects such as entanglement and superposition.

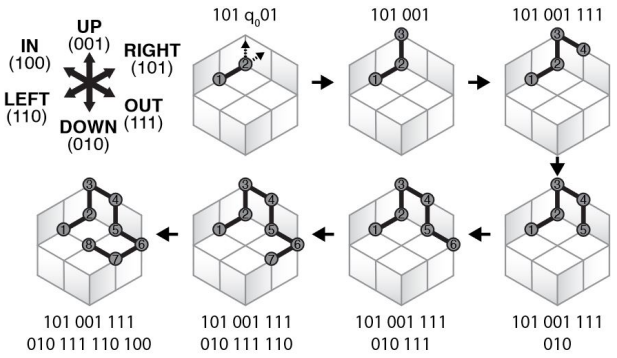

Figure 2.10: Example of the work from Babej et al., on proteing folding.

In the problem of lattice protein folding, we can view the protein as a sequence of amino acids connected by peptide bonds, that all together form a chain. The final objective is to embed the chain in a 2D or 3D lattice, with each amino acid at a vertex and the bonds on the edges connecting them. Two of the constraints are that the chain cannot overlap with itself, it means that vertices and edges can be occupied just by one element of the chain, and it cannot fold back on itself.

The quality of the embedding is scored by an energy function calculated on the interactions between neighbouring lattices. The lower the energy the better; this minimization problem is still is a difficult task since it is an NP-Hard problem. The embedding of the chain is represented by an encoding, that is a sequence of binary variables, that are indications of the sequence of moves that have been performed when laying the chain on the lattice.

The downside to the type of encoding is the massive number of qubits required for a moderate encoding, this is caused by the linear growth of the encodings length with the number of atoms; in many instances the QUBO formulation exceeds the number of physical quibits. For small instances the annealer does not return optimal solutions, this

may be due to the high degree of precision required in the scaling of the coefficients. In Figure 2.10 is reported an example of how the encoding is used and how a protein is folded on the grid.

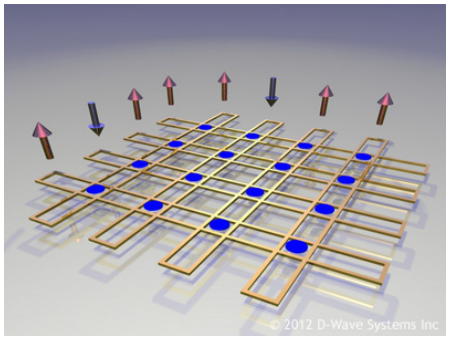

Figure 2.11: A representation D-WAVE's Chimera QPU.

There are not many works that we could compare ourselves with, while talking about quantum molecular docking, in fact there is only one case. Gaussian Boson Sampling (GBS) is a special model of photonic quantum computing. In this type of architecture, the computation is realized via the interference of identical photons that are passing through a circuit or a network of beam splitters and phase shifters, and that are measured at the end of it, at the output ports. The power of this architecture is the exponential speed up, achieved by physically generating the samples of the output instead of classically simulating the output photon distribution.

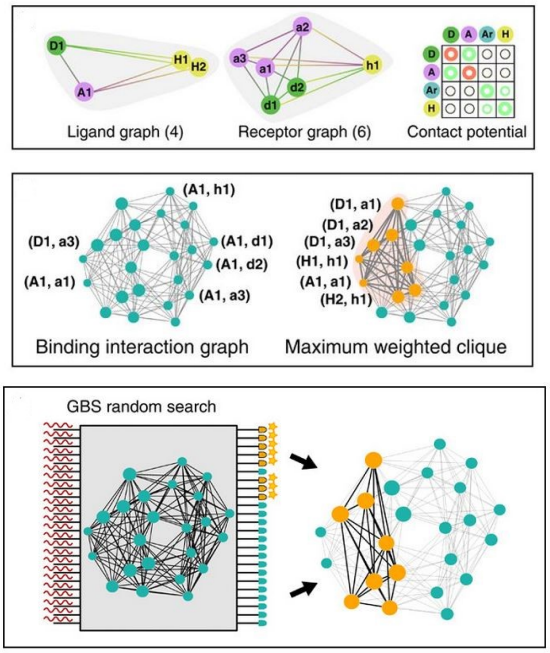

Figure 2.12: Representation of the process of molecular docking on GBS.

In the work of Banchi et al. [5], GBS can be used to find docking configurations between ligands and receptors. The method used extends the binding interaction graph

approach, where the problem of identifying docking configurations can be reduced to finding large clusters in weighted graphs. The work shows how GBS devices can be programmed to sample from distributions that assign large probabilities to these clusters, thus helping in their identification. The best docking configurations are selected based on the weight of the corresponding clique.

The method is an improvement of the binding interaction graph that associates the retrieval of docking configurations with the task of identifying large clusters in a weighted graph. In this case, the quantum computer is used for the scope of sampling from the distribution of clusters in the graph; GBS will assign large probabilities to the most dense subgraphs. The beauty of the method is that while GBS is just looking for the greatest cliques, the scoring function is implicit in the weights of the original graph. This optical quantum computer doesn't bother mitigating the difficulties associated with scoring functions, which represent the major hurdle in accurate docking, while still improving the sampling of docking configurations. Various classical techniques can be applied during the execution or in postprocessing of the results. The variant can be defined as a hybrid-quantum algorithm. In Figure 2.12 we can better visualize the three steps for the solution of the problem of molecular docking on the GBS, where first we construct the binding interaction graph from the graphs of the molecule and of the pocket, then we convert it in a weighted graph and then we input it to the photonic quantum computer. In Figures 2.13 and 2.11 we can see the difference in the type of connections and components in the two quantum circuits, where the first is made of optical waveguides and optical devices, while the second is a set of superconductive resonators connected as a grid.

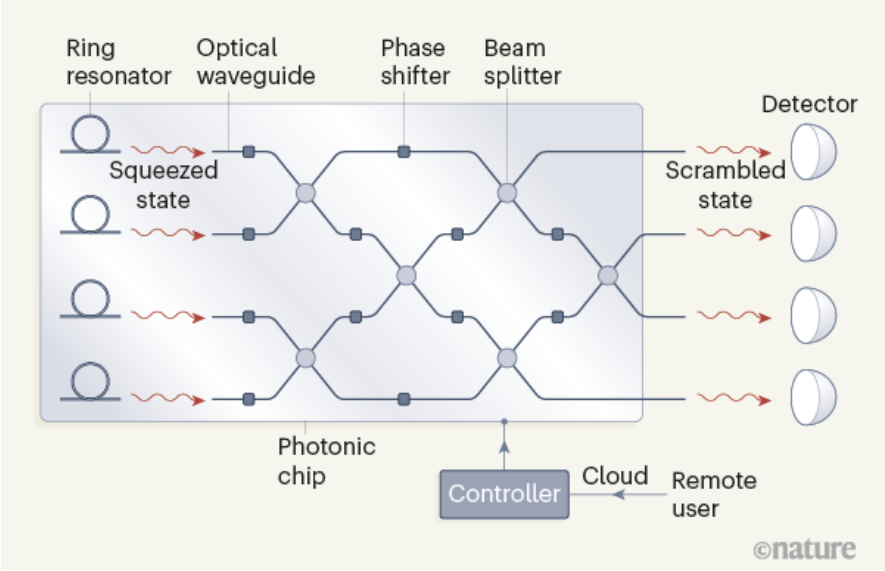

Figure 2.13: A representation of Xanadu's Photonic chip that performs the GBS.

Chapter 3

Problem Formulation

In this chapter is described the construction and the intuition behind the model used for performing the ligand expansion. In the problem definition we are explaining how the total distance is calculated, what conditions are involved in order to consider the contribution of a couple of atoms and how the contribution of a couple of atoms is dependent on the rotational matrices of the torsional bonds that are in the path that connect them. In the binary optimization formulation we are exposing what steps are needed in order to the translate the totality of the contributions in binary problem, the constraints on the binary variables and the final construction of the binary optimization function.

3.1 Problem Definition

Since the initial construction of each molecule is reproducible but it does not follow any criteria, we can derive that this methodology is introducing a shape bias that is distorting the sampling of the solution space for the docking. We need an initial shape for the molecules to be neutral for the sampling, like an unfolded shape. The objective of the project is to find this unfolded shape of the chemical substance, where by unfolded we intend to discover the shape where the volume that the molecule is occupying is maximised. Since we are not studying the spatial positioning of this structure, we are going to measure, in substitution to the volume, the total sum of the internal distances between pairs of atoms in the virtual graph of the compound. The starting point for the algorithm is the biased molecule that was previously defined as formed by fixed bonds and atoms, except for the rotatable bonds.

We can assign to each of the rotatable bonds T_i a variable corresponding to the angle θ_i with which one of the fragments connected to the segment will rotate. As a convention, we will define the ordered set of rotations by a vector of angles dictated by the bonds as:

$$t = \begin{bmatrix} \theta_1 & \theta_2 & \dots & \theta_n \end{bmatrix}$$

We assume that each torsion θ_i around the bond's axis can assume a values $[0, 2\pi)$.

Given a molecule, it is possible to construct the associated graph as shown in Figure 3.1, where on the left we reported the 3D rendering of the chemical substance and on

its right the graph of the compound. The edges of the graph or bonds cannot contract and the torsional bonds are depicted in red in the graph.

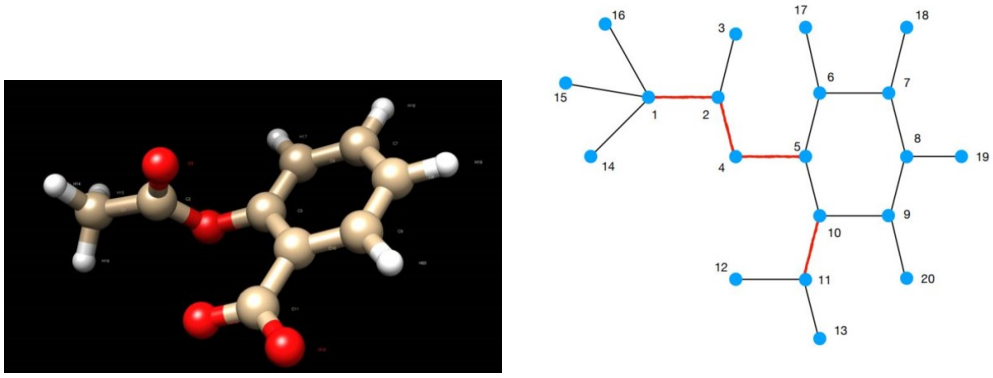

Figure 3.1: Illustration of a simple organic molecule on the left. The molecular structure, with 4 rotatable bonds highlighted in the graph representation on the right.

The scoring function for the operation of expansion, which was chosen as the total sum of the internal distances of the molecule, is what we aim at maximizing. More formally, the objective is the following: given a molecule, we want to find the "unfolded" torsional configuration that maximises the molecular volume, or equivalently, that *maximises the distances between atoms*, i.e., find

$$t^{unfold} = [\theta_1^{unfold}, \ldots, \theta_M^{unfold}] \tag{3.1}$$

such that the following quantity is maximised

$$D(\mathbf{t}) = \sum_{\substack{a,b \in \mathcal{M} \\ a \neq b}} D_{ab}(\Theta)^2 \tag{3.2}$$

The function $D_{ab}(\Theta)$ denotes the distance between an atom a and an atom b of the molecule M, different from each other, and the quantity is then squared as it represents a distance that if negative will be causing the cancellation of the contribution of other couples of atoms, while $D(\ \mathbf{t}\ )$ will be the sum of all atomic distances squared. It is useful to label atoms from 1 to N as shown in Figure 3.1.

The rationale behind this choice is that the objective function is simple and relies just on the geometry of the molecule, since each distance between pairs of atoms $D_{ab}(\Theta)$ depends directly on the angles assigned to the torsion induced by the rotatable bonds, that appear in the shortest path that connects atom a to b in the graph.

It may seem redundant to make all these measurements, but actually it is not necessary to calculate all the pairwise distances that are expressed in equation 3.2. In fact, we can make some simplifications that will just rescale the measurement, and optimize the resources needed in order to solve the expression, while still preserving the same amount of information. For this reason, we are introducing two conditions for selecting the couples of atoms for which to calculate the relative $D_{ab}(\Theta)$. The conditoins are the following two:

Condition 1. The shortest path between two atoms must contain at least one torsional, otherwise the distance is not counted in 3.2. This is due to the fact that their relative position will never change because they will always belong to the same rigid fragment.

Condition 2. The shortest path connecting two atoms must have at least three edges, otherwise the distance is not counted in 3.2. This condition is caused by the fact that in a chemical structure two consecutive bonds have always the same determined angle, so if we pretend to have two atoms connected by only two bonds, their distance is fixed as the sine of the angle between the two bonds, moreover if we rotated one of the bonds around its axis the distance is unchanged.

A useful mathematical representation for rotations are rotation matrices; however, it will be a function itself, $R(\theta_i)$, of the angle of the torsional T_i it is associated to. If two atoms are connected via multiple torsional, the rotation matrix characterizing their relationship will be $R(\Theta) = R(\theta_i, \theta_j, \ldots, \theta_k)$ that is traduced by the ordered multiplication of single torsion rotation matrices

$$R(\Theta) = R(\theta_i, \theta_j, \ldots, \theta_k) = R(\theta_i) \times R(\theta_j) \times ... \times R(\theta_k) \tag{3.3}$$

Each rotation matrix $R(\theta_i)$ recalls the structure of the homogeneous transformation matrices that are rotations about the coordinate axes, typically used in robotics. $R(\theta_i)$ is a 4×4 matrix of the form:

$$\begin{bmatrix} \frac{\left(u2\ +(v2+w2)*cost\right)}{l2} & \frac{\left(u*v*(1-cost)-w*l*sint\right)}{l2} & \frac{\left(u*w*(1-cost)+v*l*sint\right)}{l2} & \frac{\left(\left(x\_orig*(v2+w2)-u*\left(y\_orig*v+z\_orig*w\right)\right)*(1-cost)+\left(y\_orig*w-z\_orig*v\right)*l*sint\right)}{l2} \\ \frac{\left(u*v*(1-cost)+w*l*sint\right)}{l2} & \frac{\left(v2+(u2+w2)*cost\right)}{l2} & \frac{\left(v*w*(1-cost)-u*l*sint\right)}{l2} & \frac{((y\_orig*(u2+w2)-v*(x\_orig*u+z\_orig*w))*(1-cost)+(z\_orig*u-x\_orig*w)*l*sint)}{l2} \\ \frac{\left(u*w*(1-cost)-v*l*sint\right)}{l2} & \frac{\left(v*w*(1-cost)+u*l*sint\right)}{l2} & \frac{\left(w2+(u2+v2)*cost\right)}{l2} & \frac{((z\_orig*(u2+v2)-w*((x\_orig*u+y\_orig*v))*(1-cost)+(x\_orig*v-y\_orig*u)*l*sint)}{l2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The extremes of a bond are identified by the two atoms that they connect, and they can be described by the coordinates of the atoms themselves, (x_vector, y_vector, z_vector) and (x_orig, y_orig, z_orig), this will imply a sort of directionality, thus the molecule will not be any longer just an undirected graph.

The other parameters are defined as follows:

$$\text{u} = \text{x\_vector - x\_orig;} \quad \text{v} = \text{y\_vector - y\_orig;} \quad \text{w} = \text{z\_vector - z\_orig;}$$
$$\text{u2} = \text{u} * \text{u;} \quad \text{v2} = \text{v} * \text{v;} \quad \text{w2} = \text{w} * \text{w;}$$
$$\text{l2} = \text{u} * \text{u} + \text{v} * \text{v} + \text{w} * \text{w;}$$
$$\text{l} = \sqrt{(l2)};$$
$$\text{cost} = \cos(\theta) \ ; \quad \text{sint} = \sin(\theta) \ ;$$
$$\text{one\_minus\_cost} = \text{1- } \cos(\theta).$$

The components of the matrix are almost self-explanatory; the notation is kept as in the code. Consider the distance $D_{ab}(\Theta)$. The representation of the coordinate system for the atoms now has to adapt to the shape of the matrix, so the initial position of atom a is identified with $\vec{a_0} = (x_{a_0}, y_{a_0}, z_{a_0}, 1)$ while the position of atom b is $\vec{b_0} = (x_{b_0}, y_{b_0}, z_{b_0}, 1)$.

Relative positions are obtained by fixing one of the two atoms and by rotating the other:

$$\vec{a} = \vec{a_0} \qquad \vec{b} = R(\Theta)\vec{b_0} \tag{3.4}$$

Hence, we can rewrite the single contributes to equation 3.2 as the Euclidean distance.

$$D_{ab}(\Theta)^2 = \|\vec{a_0} - R(\Theta)\vec{b_0}\|^2 \tag{3.5}$$

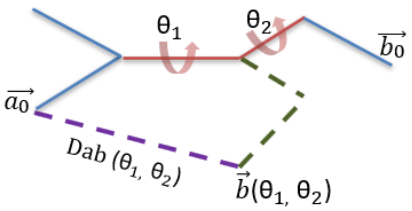

Figure 3.2: The a visual explanation of equation 3.5

Figure 3.2 is the result of the combination of the two conditions explained before with the expression of the relative position as function of the angles assumed by the rotatable bonds. We can count four bonds between the atom a and the atom b, there is at least one torsional and the new position of b is found after the simultaneous rotation of θ_1 and θ_2.

3.2 Binary Optimization Formulation

Here we formalize how the binary formulation is constructed from the distances made of sines and cosines derived from the multiplication of rotation matrices, functions of

the angles assumed by the torsionals. This step is required in order to retrieve the mathematical formulation, the QUBO or also known as Ising problem , that permits the usage of the D-WAVE quantum annealers, therefore we will obtain a formula of this form:

$$O(x) = \sum_i h_i x_i + \sum_{i>j} J_{i,j} x_i x_j \tag{3.6}$$

with $x_i \in \{0, 1\}$ binary variables and h_i and J_{ij} parameters whose values encode the optimization task to solve in such a way that the minimum of $O(x)$ represents the solution to our optimization problem.

Note that, besides being a QUBO problem difficult to solve, it can be extremely difficult as well constructing a formulation that could effectively lead to solutions of high quality.

One of the main hurdles is, that is it is very likely ending up with formulations that make use of terms of order higher rather than just quadratic, for example, terms like $x_i x_j x_k$ or $x_i x_j x_k x_l$ and so on.

In this case we refer to the binary problem as a high-order quadratic unconstrained binary optimization (HUBO) problem. Fortunately, it is always possible to convert a HUBO into QUBO, and the trick used is to add new ancillary binary variables and to substitute the high-order terms with a sum of quadratic expressions, made from the product of the original binary variables with the new ones, that preserve the local minima described by the original set of variables in the high-order term.

As an example, a cubic term like $\pm x_1 \cdot x_2 \cdot x_3$ can be divided using the ancillary binary variable a_1 as :

$$\pm x_1 \cdot x_2 \cdot x_3 \rightarrow \pm a_1 x_3 + 2(x_1 x_2 - 2x_1 a_1 - 2x_2 a_1 + 3a_1) \tag{3.7}$$

Equation 3.7 is a possible choice already tested in other works [35], but other substitutions are possible [32].

After we understood that we there are little obstacles to the actual formulation, we can continue with the rest. Let us consider a discretization of the angle θ_i associated to a rotatable bond T_i into d possible values,

$$\theta_i = [\theta_i^1, \theta_i^2, \theta_i^3, ..., \theta_i^d] \tag{3.8}$$

as a consequence, all continuous functions applied to the angles are discretized, therefore also the sines and cosines of the rotation matrix of the rotatable bond T_i become d possible values

$$\sin(\theta_i) = [\sin(\theta_i^1), \sin(\theta_i^2), \sin(\theta_i^3), ..., \sin(\theta_i^d)] \tag{3.9}$$

$$\cos(\theta_i) = [\cos(\theta_i^1), \cos(\theta_i^2), \cos(\theta_i^3), ..., \cos(\theta_i^d)] \tag{3.10}$$

Since a torsional can take only one value at a time, it will be equivalent to force θ_i being associated with only one value among all the possible θ_i^k. What we are formally describing can be rewritten as a ***One-Hot Encoding***; we can use a set of binary variables to acknowledge the torsion of a rotatable bond according to one of these finite

values. That is, for each torsional T_i, we assign a binary variable x_{ik}, with $1 \leq k \leq d$, such that

$$x_{ik} = \begin{cases} 1 & \text{if } \theta_i = \theta_i^k; \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Since the system 3.11 tells us that only one of the binary variable can be assigned a one or truth value, out of all the binary variables representing the state of a torsional at each time, then the sum of all the binary values must be one at all times, we can derive the constraint on the binary variables as :

$$\sum_{k=1}^{d} x_{ik} = 1 \tag{3.12}$$

The problem of assigning only one value among all the possible, is the same constraint of the graph coloring problem exemplified in [30] .

Then more in general, the value selection on $\sin(\theta_i)$ and $\cos(\theta_i)$ can be expressed as

$$\sin(\theta_i) = \sum_{k=1}^{d} \sin(\theta_i^k)\ x_{ik} \tag{3.13}$$

$$\cos(\theta_i) = \sum_{k=1}^{d} \cos(\theta_i^k)\ x_{ik} \tag{3.14}$$

A simple way to convert an algebraic constraint is to square it and so make it a quadratic expression; then we proceed in the same way with constraint 3.12, but for all the rotatables as:

$$\sum_{i} \left(\sum_{k=1}^{d} x_{ik} - 1 \right)^2 \tag{3.15}$$

The first summations is iterating the constraint over all the rotatables. Once the binomial square is expanded, the term that is of the shape

$$\left(\sum_{k=1}^{n} x_{ik} \right)^2 = \sum_{k=1}^{n} x_{ik}^2 + 2 \sum_{k=1}^{n-1} \sum_{h=k+1}^{n} x_{ik} x_{ih}$$

will introduce all the quadratic terms of the constraint; the two in front of the double sum will be simplified with the rest of the sqaured binomial. Constraint of eq. 3.13 is what will be called a *hard constraint* in QUBO terms.

With such encoding, the rotation matrix $R(\theta_i)$ associated to torsion T_i becomes a function of all the binary variables x_{ik} needed to represent the angle θ_i

$$R(\theta_i) = R(x_{i1}, x_{i2}, ..., x_{id}) \tag{3.16}$$

For a generic rotation we have

$$\begin{aligned} R(\Theta) &= R(\theta_i) \times R(\theta_j) \times ... \times R(\theta_k) \\ &= R(x_{i1}, x_{i2}, ..., x_{id}) \times R(x_{j1}, x_{j2}, ..., x_{jd}) \times ... \times R(x_{k1}, x_{k2}, ..., x_{kd}) \end{aligned} \tag{3.17}$$

A note regarding the granularity of the rotations, to obtain a precision of $\Delta\theta_i = \theta_i^{k+1} - \theta_i^k = 0.02\pi$ in the representation of angle θ_i we need $d = 100$ variables x_{ik}, in general the number of variables needed for each torsion will be given by

$$d = \frac{2\pi}{\Delta\theta_i} = \frac{2\pi}{\theta_i^{k+1} - \theta_i^k} \tag{3.18}$$

Given a number M of torsional bonds, the total number of binary variables is

$$n = d \times M = \frac{2\pi}{\Delta\theta_i} \times M \tag{3.19}$$

The general form of the HUBO optimization function is the following

$$O(x_{ik}) = A\_const \sum_i \left(\sum_{k=1}^{d} x_{ik} - 1 \right)^2 - \sum_{a,b} D_{ab}(\Theta)^2 \tag{3.20}$$

Here the second term is *optimization constraint*, and it has a minus sign in front because the quantity we must maximized (while the whole expression is minimized). The first term is the *hard constraint* where the parameter A is the penalty scalar that will modulate its strength and modify accordingly the problem landscape.

Chapter 4

From the Representation to the QUBO Formulation

The chapter has the objective of introducing the end-user to the application and the mechanisms used for reconstructing the QUBO model starting from the data of the molecules chosen. The application relies on two main features: computer algebra, also called symbolic computation, and quantum annealers. Symbolic computation was exploited as a flexible and reliable instrument for manipulating mathematical expressions, used as a vector for the translation of the position of atoms in space, which is a function of one of the angles θ_i that the rolling segment T_i can take, into a score composed of only Boolean variables. The chapter begins with section 4.1 that is an analysis of the database with which the project interacts, in fact it is useful to understand how powerful is our method and how many molecules we can actually solve.

Algorithm 1 Molecular Unfolder Routine

Require: $molecule\_descriptor$, $mode\_selection$, $granularity$
Ensure: $\Theta_{max} =$ *angles assignment maximising the volume.*
1: M, angles = SetUp($molecule\_descriptor$, $granularity$)
2: M = MoveAtoms(M)
3: $internal\_distances = get\_contributes\_poly(M)$
4: **if** $mode\_selection$ **then**
5: $\quad \Theta_{max} = classical\_inspect(internal\_distances,\ angles)$
6: **else**
7: $\quad HUBO = huboization(internal\_distances,\ angles)$
8: $\quad \Theta_{max} = quadratization(HUBO)$
9: **end if**

The main workflow used by the application is shown in algorithm 1; this is a simplification made for understanding how the information flows in the project. The line 1 corresponds to section 4.2 where, taken the descriptor file of a molecule, we are returned with the graph of the molecule (M), already split in fragments that follow the requirement that the atoms in the same fragment will have the same *rotatables influece set*, plus

a list of the possible values an angles of a torsional can assume. Line 3 is still described in section 4.2, and it's where the expression of the total sum of the internal distances, in terms of sines and cosines of the angles of the rotatable bonds, is derived from the graph of the molecule. Line 5 tells us that in the workflow of the project there is a checkpoint where we can decide if to solve the total contributions not yet manipulated, with one of the classical algorithms in section 5.3, besides simulated annealing. Line 7 corresponds to section 4.3, where we show what kind of substitutions and approximations are applied in order to convert the original expression of the distances into a HUBO. The line 8 corresponds to section 4.4, where we take as input the HUBO expression of the problem, and we show how the reduction by substitution works and enables the transformation from HUBO to QUBO expression; the section ends with an attentive analysis on how the embeddings are performed and on how the runs are executed on the quantum annealers and SA.

4.1 Domain Analysis

Before discussing the methods, we should prove that the ligand expansion phase is somehow useful. As proof that the initial conformation of the molecule cannot be considered optimal, a sample of 118 substances ordered by increasing volume or by a proxy of it, i.e. by the number of atoms and or by approximately the number of fragments divided by two, was expanded with the use of the expansion algorithm inside *GeoDock* [18].

In Figure 4.2, it is easy to understand that the application of an expanding procedure benefits the molecule with an increased volume determined by the height of the blue bar. On average our classical reference algorithm improved the volume of each atom by 17% starting from the initial pose.

4.2 Set up - Modelling the molecules

The first task performed is the set-up in line 1 of the workflow, where the main data structures are created and manipulated. Line 1 hides a process that starts with the parsing of the pharmacophoric information of the chosen molecule to analyze from a MOL2 file. A graph is constructed from it and it contains the three-dimensional description of the substance, which includes the positional coordinates of the atoms, the bonds that bind them and their type, other information are disregarded.

Bonds can be seen as a relationship that binds an origin atom and a target, and its nature is identified by a type chosen from 1 (= single), (2 = double), (3 = triple), am = amide) , ar (= aromatic), dummy, unknown, not connected.

In this pre-processing phase, two simplifications related to chemical-physical reasons are applied: the first is that only single and aromatic bonds can be valid candidates to be rotated, and the second requires the removal of terminal hydrogens in the molecule.

The reason for the latter is that the hydrogens' distance from all internal atoms does not affect the total sum of internal distances, and since the single bonds of the hydrogens should be considered rotatable, their presence creates fragments of an element

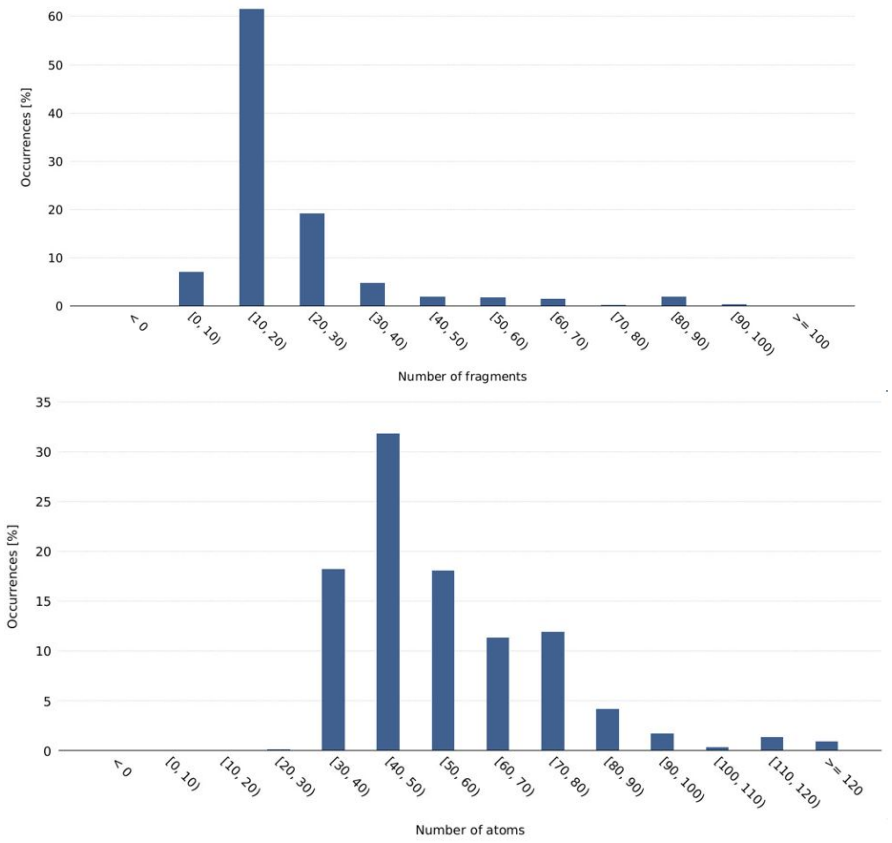

Figure 4.1: Analysis of the distribution of the two features selected for the molecules, number of atoms and number of fragments.

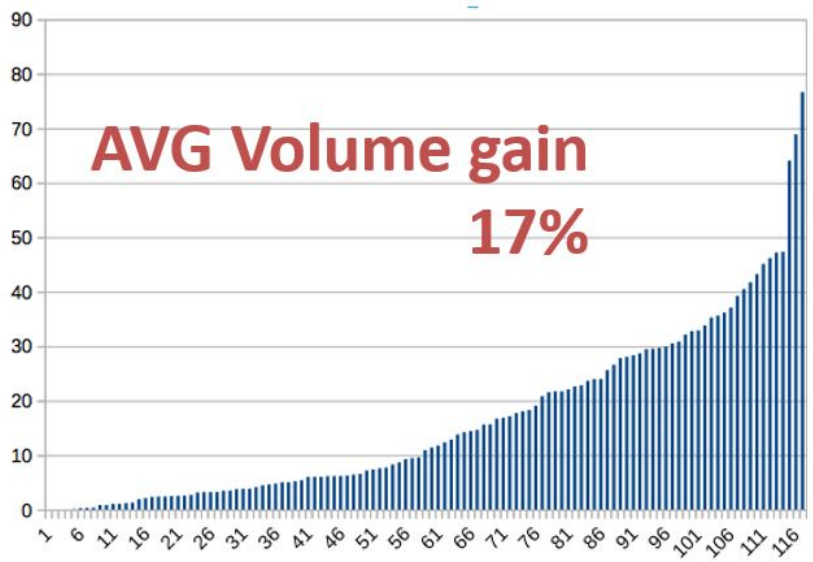

Figure 4.2: Results of the expansion with the classical greedy algorithm for 118 compounds ordered by volume.

only, whose rotation has no effect, therefore a computational waste.

The removal of the terminal hydrogens triggers the generation of conformations that would present what is called "internal bumping", that is when the final positions of two virtual atoms are less than the sum of the van der Waals radii. These invalid shapes will be removed in the post-processing phase among the available ones. In this setup phase also the set of feasible discrete angles is identified and returned, as according to 3.18 and the granularity desired.

Once it is established that the initial conformation in the common case is to be discarded, in line 2 we proceed by virtually dividing the graph of the substance into fragments like in Figure 4.3.

The code here performs the identification of the fragments, which have a slightly different definition from the one previously given as one of the two halves in which the molecule is divided by the removal of a bond, from now it will have a new meaning. The code starts by the calculation of the betweennes centrality [17], equation 4.1, of each atom. This attribute allows us to define an ordering of the atoms by centrality within the graph of the chemical substance.

$$\text{betweeness centrality of atom v} : \sum_{v \neq s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4.1}$$

Formula 4.1 is calculated for each atom v, as the sum of the ratios between the number of possible shortest paths between all the possible pairs of atoms s and t, different from v, that cross it, and the number of all possible shortest paths between s and t that exist with no restriction. Afterwards, the atom with the greatest centrality is chosen as the centre of the structure and the origin for any ordering of the bonds; the subsequent action performed is the construction of the routes in the graph starting from the central vertex to all the vertices. The set of torsional bonds is constructed by removing iteratively one edge at a time from the graph, and then by checking if the graph is still connected, if the outcome is negative, it will be following the definition of torsional.

Each fragment is identified as a set of atoms influenced by the same set of rotatable bonds, which we called *rotatables influence set*. The definition of rotatables influence set is depicted below.

$$\text{rotables influece set} : \quad \mathrm{I}_s = E_{C_{a,a_k}} \cap E_R \tag{4.2}$$

$C_a = atom\ with\ greatest\ betweeness\ centrality\ \ or\ central\ atom$
$E_R = Rotatable\ bonds$
$E_{C_a,a_k} = Bonds\ on\ the\ shortest\ path\ \sigma_{C_a,a_k}$

All atoms belonging to the same fragment are mapped to a unique shortest path composed of the edges in their influence set, and a rotational matrix is created for each bond. Eventually, a composite rotational matrix will be created as a product of the single matrices, in order from the outermost bond to the innermost. The position

vectors stored in the vertices of the graph as attributes are then multiplied with their corresponding composite matrices.

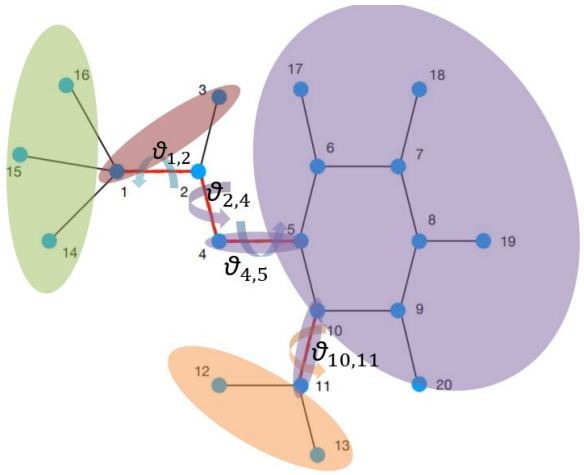

Figure 4.3: An example of the principle of the rotatables influece set.

Here comes into play the usage of computer algebra, since the expressions, born from the matrix products that appear in the rotation matrix 3.1 are symbolic expressions constructed in Python with the help of the Sympy library [36]. Some atoms will have their initial position, while others will have a position vector made of symbolic coefficients as non-linear trigonometric functions, that will be turned into real numbers by their evaluation at the angles assumed by the torsional.

In line 3 the contributions are computed as in equation 3.5 for each fragment, with special care to the fact that the distances between a couple of atoms, one from a fragment 'A' and one from a fragment 'B', are calculated only once and especially that they respect the conditions 1 and 2. Distances inside the same fragment are not calculated since they do not change.

After recovering the partial contributions of a fragment, we sum all them up to get the total of the internal distances. The total is also a non-linear symbolic expression where the symbolic variables representing the turning angles are wrapped in trigonometric functions. Now the formulation obtained is computed once and for all, and can be saved for a variety of future applications. The same can be done for the graph with its attributes, since the symbolic positions of the atoms are a function of the turning angles, they could be exploited in the future for new applications that modify the shape of the substance, like the successive phases of docking.

4.3 Huboization

We defined the procedure *huboization* where in simple words we substitute every trigonometric function into the expression of the total contributions as according to relation 3.13, and the result is then combined with the hard constraint 3.15.

The process of derivation is made of several steps, where after identifying the ordering

of the bonds, we first we substitute the single cosines and sines representing a torsional, with a placeholder a_i, with i going from 0 to $2 * number\ of\ torsionals$.

The placeholders that are of the form a_i^2, are replaced by a symbol that keeps track of the power but that mathematically will keep the power low in a term. As a third step, instead of the placeholder it is put the one-hot encoding as $sin/cos(\theta_i) * X_0 + sin/cos(\theta_d) * X_d$; each of the theta angles is one of the possible discrete angles that the rotatable bond can take. Depending on the placeholder they replace the real valued sines or cosines will be raised to a power. The expression is then expanded, and we end up with owning a $HUBO$, since the terms will be for example $\cdots + X_0X_2X_4X_5 + \ldots$ which is a product of two or more binary variables; the letters X are still symbolic binary variables. Since binary variables with powers greater than one will be just 0 or 1 again, as a major mathematical simplification, we set all the binary variables' powers to 1 and sum up the real part of the terms made up by the same literal part, this will decrease the degree of power of each term. This phase of expansion that we just mentioned is one of the most computationally intensive, at least when using a symbolic framework. The challenges that arise during this part of the whole process are strictly related to the enormous amount of memory used and the unparalleled implementation of the symbolic compiler.

Proceeding with the derivation of the QUBO is unrealistic, several simplifications allow us to reach the final phase in which we can feed the annealer with the data.

4.3.1 Approximations

The distance simplification

The greatest simplification introduced in the routine for calculating the respective contributions is the simplification in the number of pairs considered; instead of measuring the Euclidean distance of each atom a from a fragment A to each of the atoms b in a fragment B, we consider only the median atom in each fragment, and since the atoms inside the fragments are again ordered by centrality, these atoms will be the central components of these molecule segments,a sort of barycenter. This is similar to an extreme form of loop perforation, or an edge contraction on the graph of the molecule[33].

In almost the whole number of cases without this technique, the intermediate substitutions did not even terminate with the current framework used, hence it was impossible to compare the number of terms present in a HUBO where this simplification is triggered and where it is not. For a small number of rotatables, e.g. 1, 2 , the number of terms is almost identical, but for more difficult problems where the workflow deals with 3, 4 or more torsionals, it becomes already unfeasible since the number of terms grows exponentially with the number of torsionals. The formulation will have much less terms and the preprocessing times are much better performant.

Coarse-grained rotations

As the dimension and complexity of a molecule is increasing, we have to face several problems connected with our limited computational resources. We have again to employ

trade-offs that take us away from real optimal solutions but still of very much acceptable quality.

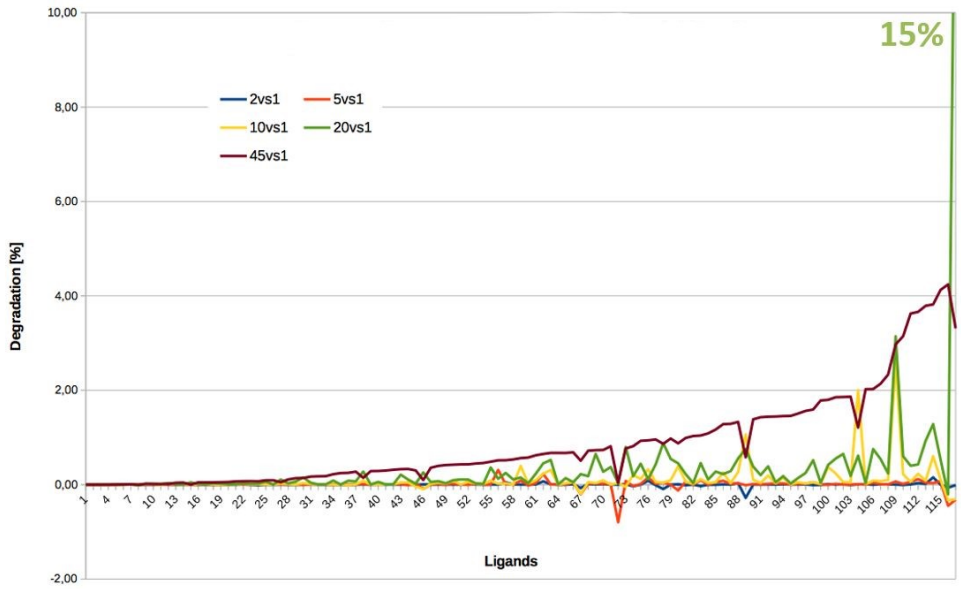

Figure 4.4: Analysis on the results degradation by transitioning to a coarse-grained rotation.

A first precaution that we have been able to verify concerns the granularity with which the rotations of the fragments can be performed.

In figure 4.4 we have again the 118 molecules shown before, always in order of increasing volume. The vertical axis shows the percentage of degragation of the final volume for each molecule when their expansion is made with an angle greater than 1 degree. The lines drawn in the plot concern rotations at multiples of 45, 20, 10, 5 and 2 degrees. The surprising result is that the degradation is lower in all cases below 2% except for the rotations at 45 degrees which reach a peak of 4% but with average around 1.7%. There is an outlier at the last molecule which for rotations at 20 degrees explodes at 15%. Even though we introduced the possibility of using different granularities (as the granularity input), we believed in using multiples of 45 degrees, because the construction of the instance would require only 8 logical qubits for each rotatable bond to identify its position in the make of the HUBO, and the final potential quality loss is acceptable. On the other hand, the other extreme would be using a granularity of 1 degree that would require 360 logical qubits, a total waste of resources.

Landscape alterations

This last simplification proposed implies that if we wanted to use an exact mathematical formulation for our application, we would have to use more qubits and use very complex logical embeddings of the qubits. Eventually, we poured all our efforts in finding ways that relax the problem instance, so that embedding less complex structures and using fewer logical qubits would lead to near-to-optimal results more frequently as proved in [41]. From the beginning of the experimental phase we noticed that we were running

out of qubits as the number of rotatables was increasing, so in order to fit the problem on the device two other remarkable simplifications are made. The first is the rounding of the HUBO and QUBO coefficients due to the physical sensitivity[1] of the quantum devices and for faster formulations, the second is the elimination of HUBO and QUBO terms from the whole expression, based on their multiplicative coefficients, or simply called "chop".

The chopping consists in eliminating the terms from the optimization constraints and not related to the hard constraints, that present a coefficient strictly less than a threshold value in HUBOs and QUBOs. The fine art of chopping will be the trade-off between less exact specifications and better embedding (thus improved quality). We should have made a premise about the chop. As the elements in the QUBO are getting chopped, there is the risk of eliminating all the terms that present a common variable between each other. The missing variable is backtracked throughout the transformations and it will result as "don't-care" assignment to one of the logical variables assigned to the rotatable bond, therefore the chances of getting worse solutions or even invalid ones will increase dramatically.

The decision of when and where to apply this "chopping" phase can dramatically change the landscape of the problem. In the context of this project, it was performed after the application of the placeholders, together with the rounding of the coefficients, and after the creation of the symbolic expression of the optimization constraint of the objective function. In the code, this parameter is a float number that represents the exponent of $\frac{1}{10^{chop}}$, if chop is equal to 1, then any term with coefficient less than 0.1 in absolute value will be deleted. Depending on the pre-existing landscape, this operation can be seen as *smoothing* the surface by removing local minima close to each other and that are not very deep; pay attention that smoothing can create valleys from which the algorithm can't escape anymore.

4.4 Quadratization

Once the HUBO is calculated as the summation of the hard constraints expression and the optimization constraints, in order to perform the annealing, we need to derive the QUBO form from it. Firstly, we apply recursively the substitution expressed in 3.7. One can decide to build a custom function or exploit a wide range of functions present in DWAVE's software offer, as "make_quadratic". In our case, we developed a more precise version of **dimod.make_quadratic**.

As a first step, the function must create new "slack variables", binary variables that will keep track and split ternary products into quadratic ones; this can be applied recursively. The creation of the slack variables can be done by either performing a *reduction by minimum selection* or a *reduction by substitution* [21], from which are derived more advanced works as [32].

We sticked to the method by substitution which introduces a new variable z in order to express the product of two binary variables $x_1\, x_2$ inside a function multiplied by a

[1] energy scaling of the couplers

penalty term as:

$$P(x_1, x_2, z) = x_1x_2 - 2(x_1 + x_2)z + 3z \ , \ z \iff x_1x_2 \tag{4.3}$$

This shows that the term expanded adds a strictly positive energy cost to all the solutions that do not reflect the equivalence between z and x_1x_2, so if the identity is injected in a bigger formulation, its results will be:

$$x_1, x_2x_3 = \min_z\{zx_3 + MP(x_1, x_2, z)\} \ , \ M_{penalty} > 1 \tag{4.4}$$

The difference with the original function is that in our implementation P is equal to the product of all the coefficients in the HUBO that are related to z, times an amplification factor that is tuned depending on the factor. Meanwhile, the DWAVE implementation sets P to a constant unrelated to the energy of the variable but sets it equal to the maximum of the function rescaled by a factor; this is completely deranging the walk of the algorithm.

The QUBO now can be chopped again and tranformed in a python dictionary that will be encoded in order to be launched via DWAVE's sapi library on the device.

4.4.1 Physical embedding and post-processing

Finally we procede with the runs with attention to a slight tuning of the procedure.

The anneal is performed multiple times by trying different values for the 'A_const' constant in the penalty function which we chose to be the maximum of the optimization constraints, multiplied by an increasing amplifying scalar. Once we get the adjacency matrix of the topology of the device available, we try several times the embedding and we take the one that uses the least physical qubits, it's implicit that this embedding will also be the one with the shortest chains. The QUBO problem is transformed in an instance of ideal Ising model, as in 3.6. The Ising model is then physically embedded from which we fetch the the parameters h, J_q, J_c, that are respectively the linear terms, the quadratic and the chain energy levels.

We performed runs only by forward annealing, where the state of the system is modified from the initial state to the final state without any stops or any reversion of the state to any intermediate ones, called "reverse annealing".

After the run we collect the results of the procedure that is a set of solutions or set of assignments to the binary variables $\vec{x}_{sol}$, that represent an energy level of the quantum system. Unfortunately, even though a related energy level is optimal, the assignments could defy our initial conditions and constraints, due to the side effects of the procedure; for example, we could have *broken chains* or chains that do not have the same value on each physical qubits, to solve this problem we used majority voting to recover the possible value of the whole chain. We finish by post-processing the result, where we convert the Ising spins -1 and +1, in 0 and 1, and then we eliminate the solutions that did not logically respect the hard constraint. It is often likely that even if the annealer found many results, none of them could be valid.

Chapter 5

Experimental results

In this chapter we will describe in detail all the results and experiments of the project. After outlining the machines and the software libraries used in section 5.1, in section 5.2 we will give a short introduction on the data selected for the experiments, and on the criteria adopted for the selection. In section 5.3 all the classical algorithms used as comparisons with the quantum annealers are described in detail.

Experimental results in terms of optimizations and comparative results are shown in the following sections:

- Complexity of the creation of the HUBO is shown in Section 5.4.
- The effects of chopping is Shown in Section 5.5.
- Algorithms Comparisons, also including the effect of the distances simplification, is shown in Section 5.6.

5.1 Machines and libraries

The platform used for running the classical algorithms is based on NUMA nodes, featuring 2 Intel(R) Xeon(R) CPU E5-2630 v3 CPUs (@2.40GHz), Virtualisation VT-x, caches L1d, L1i cache of 32K. L2 and L3 caches are, respectively, 256K and 20480K. RAM memory of 125 Gb and 114 Gb of SWAP memory. Operative system used is Ubuntu 18.04.5 LTS Bionic. The code was written in python 3.9.1, compiled with gcc 9.2.0. The parallelization was actuated with the usage of mpi4py 3.0.3, compiled with MPICH 3.3.2.

Since the classical probabilistic algorithms, grid search and random search, could be run in parallel, mpi4py [11] [12] was used for a broadcast operation of the input parameters at the beginning of the execution, and then for a successive MPI collective "Gather", from all the processes to the root process, where the results were filtered for the maximum value of volume and the value of the angles that maximise the volume of the molecules. For each problem instance, the number of processes used for each run was 32.

Other libraries of support are: Pandas 1.2.1, Numpy 1.19.5, Biopandas [40], 0.2.7; symbolic calculations were performed thanks to Sympy 1.7.1 [36]. Please be aware of the version of Sympy, which has undergone through noticeable changes in the latest versions.

5.2 The Dataset

Since in Figure 4.1 we can see that 60% of the molecules of the database of compounds at our disposal present on average 10 rotatable bonds, we made the choice of analyzing molecules with the same features' set.

The molecules were selected with the objective of representing compounds with very different numbers of atoms, but we decided to concentrate our research only on molecules that had at most only 12 rotatable bonds. Because of the statistical relevance of the number 10 for the number of rotatables, we decided to study only the effect on the complexity of the resolution for the first 10 central torsionals for each problem; the last 2 bonds have almost zero effect on the variation of the scoring function since they rotate with high probability fragments of small cardinality, or they are terminal bonds that disappear after the removal of the terminal hydrogens.

We started from seven mol2 files containing each of them respectively: 42, 40, 33 , 35 , 30 , 28 and 25 chemical substances. Every mol2 file contains only molecules with the same number of atoms; declared in the same order as before, with 20, 25, 30, 35, 40, 45 and 50 atoms. From each file, we randomly selected only two molecules from the set of those which had the smallest initial volume. Once isolated in separate mol2 files, we ran our experiments by considering only a subset of torsionals at a time, out of the 10 in total. The results are statistics drawn from all the molecules selected, while considering only the same number of degrees of freedom every time.

Our study aimed at highlighting the change of complexity in the problem's instance by augmenting the number of torsionals each time.

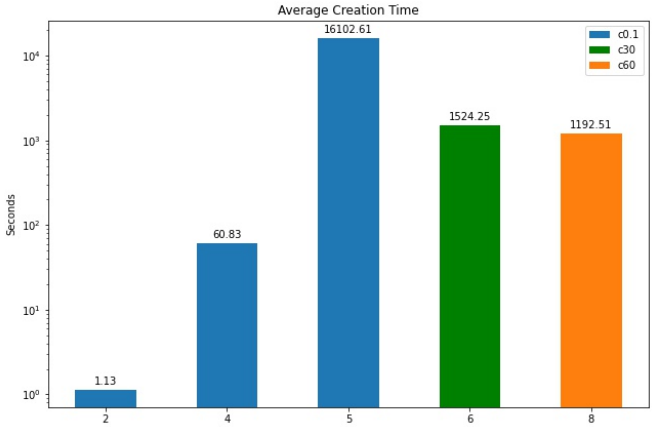

Figure 5.1: Average creation time for the unapproximated HUBO.

5.3 Altenative Search Strategies

This part of the study is a clarification of the optimization strategies used later to compare the classical methods against the quantum annealing device, in order to discover the set of assignments that can maximize the evaluation of the contribution's expression.

Since it is unfeasible to satisfactorily explore with runtimes comparable to real docking applications our solutions' space, and given the impossibility of possessing the extreme computational resources necessary to perform an exhaustive research, there is a need for algorithms less expensive, primarly in terms of memory.

Grid Search

This algorithm is usually employed in the field of machine learning, where the models need an hyperparameter optimization to perform optimally. The choice of the parameters are driven by a *loss function* on the given data, in our case the parameters to optimize are the assignments of the angles that optimize the volume of the compound.

Grid search [27] is an exhaustive search on a subsample of parameters, usually picked manually and repeated many times, in our case it will be randomic since we do not currently have a measure that could tell us which angles could pilot a faster and more conclusive search for the optimum. Grid search will still be bounded by a memory limit, so the number of parameters selectable is quite low. One of the positive aspects of this method is the chance of running it in an embarrassingly parallel mode, and that it performs well on discrete spaces like ours.

Practically, selecting a subset of dimensions means rotating the corresponding bonds and fixing to the initial value the other torsionals.

Algorithm 2 Grid Search

Require: $A = Angles,\ Poly = D_{ab}(\Theta)^2,\ T = Torsional\ Bonds\ Set$
Ensure: $\gamma_{max} : \mathbf{max}\ D_{ab}(\Lambda = \gamma_{max})^2$
1: $\Lambda = \{\lambda | \lambda \in T\},\ |\Lambda| = s$ ▷ Random uniform selection of s torsional.
2: $\Gamma = A_1 \times A_2 \times ... \times A_s$ ▷ Cartesian product of the set of angles - s times.
3: **for** $\gamma_i\ in\ \Gamma$ **do**
4: $Evaluate\ Poly(\Lambda = \gamma_i)$ ▷ Evaluate assignments and record the optimum.
5: **end for**
6: return $[\ \gamma_{max},\ max\ Poly]$

Random Search

Random Search [7] belongs to the same family of probabilistic algorithms related to hyperparameter optimization. It substitutes the exhaustive enumeration of all combinations by picking up randomly from a uniform distribution and testing the candidate solution. The pro of this method is the adaptability to a discrete setting, but also other types of space can be searched, albeit in combination with techniques for improving the outcome.

It can outperform grid search when the number of dimensions is small. A big strength of this method is again the high parallelization and the extendability to the use of Bayesian techniques or the chance of using probability distributions that are more suitable for the problem.

In practice, this algorithm can be applied on smaller subdomains of the same number of dimensions of the original solution space, where for a fixed time, a set of random points will be picked out and the torsional bonds will rotate according to the discrete angles or coordinates.

Algorithm 3 Random Search

Require: $A = Angles,\ Poly = D_{ab}(\Theta)^2,\ T = Torsional\ Bonds\ Set$
Ensure: $\gamma_{max} : \mathbf{max}\ D_{ab}(\gamma_{max})^2$
1: $|T| = m$
2: $\Gamma = A_1 \times A_2 \times ... \times A_m$ ▷ Cartesian product of the set of angles - m times.
3: **while** $t < MaxTime$ **do**
4: $\gamma = pick\_rand\_uniform(\ \Gamma\ )$
5: $Evaluate\ Poly(\Theta = \gamma)$ ▷ Evaluate assignments and record the optimum.
6: **end while**
7: return $[\ \gamma_{max},\ max\ Poly]$

The difference between random and grid search is crucial. Random search represents the class of Monte Carlo algorithms, while grid search is a Las Vegas algorithm. Even though both categories are randomized algorithms, the Las Vegas type always returns the correct result, otherwise it signals the failure. The classical definition expects the computational time to be a finite variable depending on the instance, and requires a sort of bet on the resources. On the contrary, Monte Carlo algorithms are based on repeated random sampling of results, while the runtime is fixed, but the answer will be incorrect for a defined probability. The definitions reflect the modus operandi of the first two methods.

GeoDock-inspired

The algorithm in this case is based on the mathematical method called *dynamic programming* [6], that was coined by Richard Bellman, and it means tackling a problem by dividing the decision process in intermediate steps over time, in a recursive manner. In fact, the word dynamic denotes the time-varying aspect of the model and programming is used for defying an ordering. This model cannot always be applied, but if it is, then we can tell that the optimum solution to an instance of our problem, contains the optimal solutions to the subproblems.

The GeoDock-inspired search is a greedy algorithm, which makes a decision locally at each rotatable bond, it does not lead to an optimal solution for the majority of the time, but it can give a good approximation in a reasonable time.

This algorithm is the only one that is not utilizing symbolic computation in our experiments. Starting from the most central and internal bond, and going outwards by

Algorithm 4 GeoDock-inspired Search

Require: $A = Angles,\ Poly = D_{ab}(\Theta)^2,\ T = Torsional\ Bonds\ Set$
Ensure: $a_{max} : \mathbf{max}\ D_{ab}(a_{max})^2$
1: $|T| = m$
2: **for** $t_i, \theta_i\ in\ T_{ordered}$ **do** ▷ Ordered by betweenness centrality.
3: **for** $a\ in\ A$ **do**
4: $Evaluate\ Poly(\theta_i = a)$ ▷ Record intermediate optimums.
5: **end for**
6: **end for**
7: return [$\underline{a}_{max},\ max\ Poly$]

following betweenness centrality, each bond is rotated for each angle it could assume, and only the conformation that improves the volume is recorded in a vector. The whole process is repeated a number of times, or until a stalemate in the improvement is reached.

Simulated Annealing (SA)

Simulated annealing is a probabilistic algorithm which runs for approximating the global optimum of the problem, in particular this is a metaheuristic useful in the search of the optimum in large solution spaces.

Algorithm 5 Simulated Annealing Search

Require: $A = Angles,\ Poly = D_{ab}(\Theta)^2,\ T = Torsional\ Bonds\ Set,\ N_{max} = Total\ Iterations$
Ensure: $a_{max} : \mathbf{max}\ D_{ab}(a_{max})^2$
1: $T \leftarrow T_{max}$
2: $a_{max} = \text{NULL}$
3: **while** $T > T_{min}\ and\ N < N_{max}$ **do**
4: $next = pickNeighbour(T, best)$
5: $\Delta E = Poly(next) - Poly(a_{max})$
6: $r = randomNumberUniform(0, 1)$
7: **if** $\Delta E < 0$ **then**
8: $a_{max} = next$ ▷ The neighbour is accepted as the new optimum.
9: **else if** $r < e^{\frac{-\Delta E}{k_b * T}}$ **then**
10: $a_{max} = next$ ▷ Moving to a neighbour.
11: **end if**
12: $T = geometricDecrease(T)$
13: **end while**
14: return [$\underline{a}_{max},\ max\ Poly$]

We largely described it in background section and we used this algorithm as a the main comparison against the QPUs. Since they both ran the QUBO formulations of our problems, SA will be the benchmark for the resolutions based on the annealing

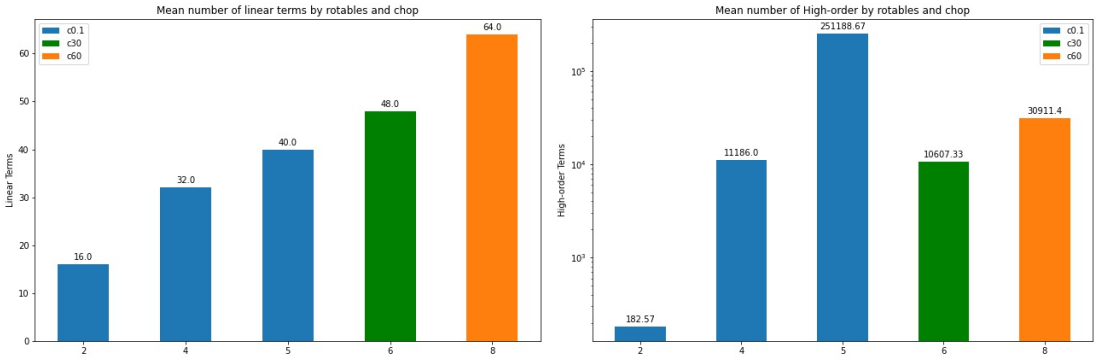

Figure 5.2: Mean number of linear and non-linear terms in the unapproximated HUBOs.

phenomenon. In algorithm 5 you can find the pseudocode for a much clear understanding of how the walk on the landscape of solutions is performed by SA.

5.4 Complexity on the HUBO Creation

In Figure 5.1 we can see the average creation time in seconds for an increasing number of rotatable bonds. The numbers 2,4,5,6,8 were chosen with the intention of showing a particular behaviour for the process of creation of the full HUBOs, constructed with the use of the distance simplification. Our study stops at 8 rotatable bonds, since the construction of a HUBO at 10 torsionals did not terminate with the current setup and software.
The plots show three thresholds of chopping, $0.1, 30, 100$. The value 0.1 represents an almost null value of chopping, for which almost all the terms are preserved, meanwhile 30 and 100, are the minimal chop threshold for which the construction of a problem with 6 and 8 rotatable bonds was possible, without exceeding the platform's memory and with a decent runtime. A chop at 0.1 can be considered a pretty good achievement, although the time increases almost 239 times when going from 4 to 5 torsionals.
By increasing the chop in this first phase we managed to keep the construction, when going from 4 to 6, and 4 to 8, only to approximately 25 times and 20 times, instead of a lower bound of 57121 times regarding the scaling from 4 to 6.
This effect is due to the usage of a python symbolic library, but it can still be representative of the complexity of the problem. We tried using C++ symbolic libraries, but they can only moderately improve these runtimes, the solution could be introducing parallelization which would give us runtimes with almost linear acceleration.

5.5 The Effects of Chopping and Embedding Exploration

Two other metrics for evaluating the complexity of the problem's instances are the number of linear and high-order terms in Figure 5.2. The number of linear terms follows the relationship: $Number\ of\ rotables \times\ granularity\ of\ rotation$, so for 2 bonds considered we will have 16 terms, for 4, 32 and so on... It is curious to see that the number of

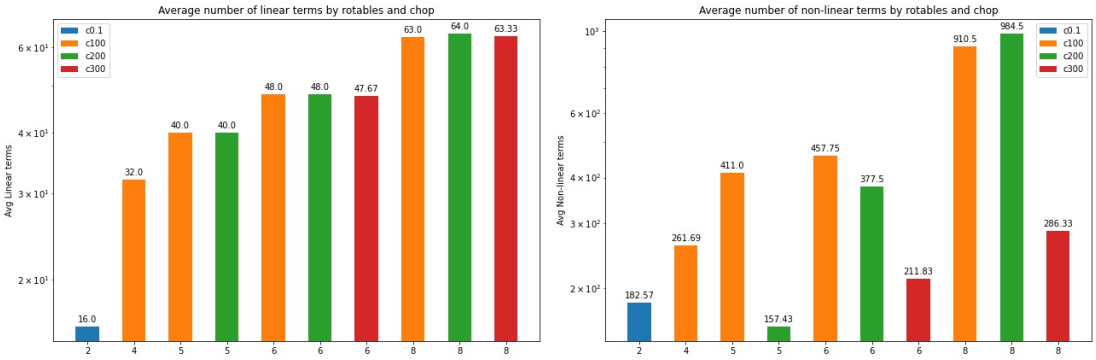

Figure 5.3: Average number of linear and non-linear terms in the approximated HUBOs.

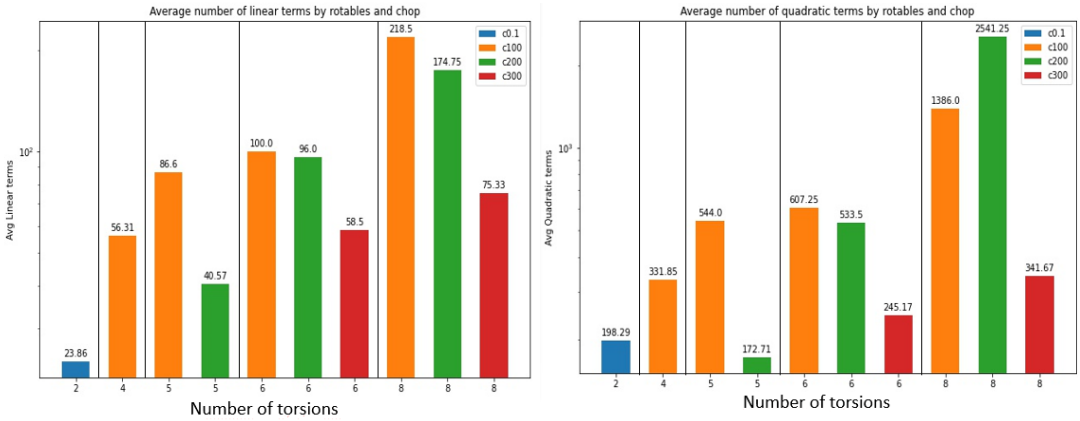

Figure 5.4: Average number of linear and quadratic terms in the approximated QUBOs.

non-linear terms is increasing steadily but not exponentially. In fact, the number of non-linear terms when going from 2 to 4, the size increases of 61.2 times, if we had to average it would be almost 30.6 times per rotatable bond, and when from 4 to 5, the expression becomes 21.23 times larger, the ratio for each bond actually decreases. We are able to get this type of growth in the non-linear terms thanks the initial chop, that is allowing us to avoid the exponential growth that these numbers would otherwise follow. Although these numbers can give a positive feeling about the the complexity of the general problem, we should keep in mind that this first part of the discussion was only dealing with the construction of the expression to solve.

Since the final objective is to be able to embed our expressions into a QPU, we are obliged to chop even more the HUBOs before applying the quadratization to the expression. The choice of the amount of chop is done by a fine tuning of this cancelling threshold, that is lowered until the embedding cannot be performed anymore. We apply it twice, once on the HUBO and then on the QUBOs derived from them.

This way of proceeding slowly acts as a classifier on the set of molecules, and categorizes them in classes of "difficulty". Since there are three points in the workflow where we are applying the chop, we can view this process as a decision tree of three levels where

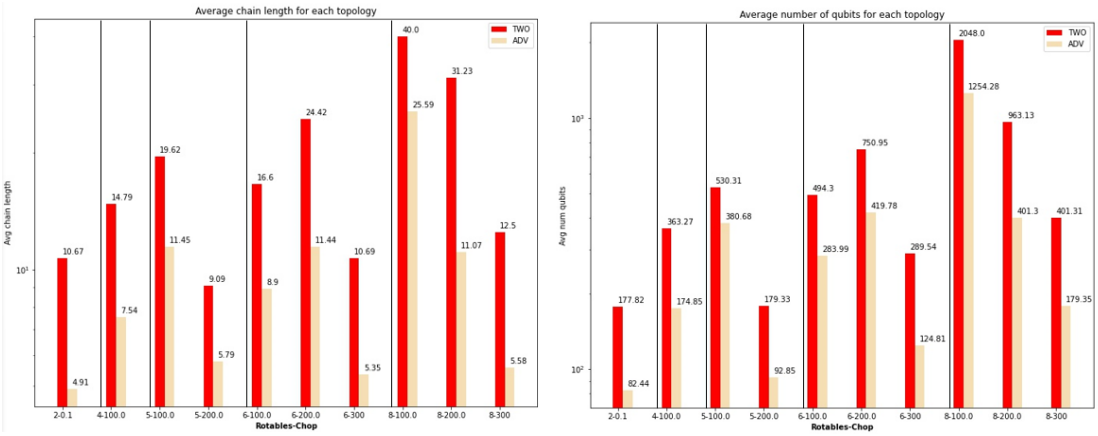

Figure 5.5: Plots depicting the average chain length and the average number of qubits in the embeddings.

each node is a chopping phase, with leftmost branch the smallest chop and rightmost branch the highest branch possible. The root is the initial expression, and the leaves record the history of chop thresholds, required to allow us to perform the embedding of a particular molecule, and the final QUBO. Each leaf is a sort of category that implies the complexity of a molecule, and depending on the path in the tree, we can tell more or less if the QUBO will be of high quality or not; the leftmost leaf will contain only the easiest molecules that need only chops at 0.1, while the rightmost will be the most difficult ones. If the molecule can end up in the more categories the leftmost will be the one with the highest quality of QUBO.

The plots in Figure 5.3 and Figure 5.4 may not have a direct connection, but we can glimpse an increasing trend in the columns representative of the same chops for an increasing number of rotatables. There is an increase in the number of linear terms because of the quadratization as expected, and the number of linear terms is decreasing for an increasing chop for molecules with the same number of torsionals. It is likely that these molecules come from HUBOs that have already been stripped of the highest order terms, which would have induced the introduction of new linear substitutive variables. There is an odd fact concerning these data, and it's the oscillation in the case of 8 rotatable bonds. This is due to the skewness of the difficulty's distribution of the molecules, that is concentrated around what we would call, the "8-200.0 class". This is why the plots should be read by following the strong colors, then combined with the use of the x-axis, and not the opposite.

The plots of the embeddings at Figure 5.5 are highly representative of what are the capabilities of each topology. There is an important detail regarding these two plots, that the columns of 2000Q at $8 - 100.0$, both for the average chain length and the average number of qubits are saturated to the maximum value for the that topology. This is because the topology couldn't embed all the possible molecules for that category, therefore when averaging the values radically increased.

Now we can see a clear difference in connectivity of the topologies, since the Chimera QPU has visibly chains almost twice longer than those of Pegasus in average. To be precise, for Chimera, the maximal value is 2.82 times the chain of Pegasus and corresponds to the columns at $8 - 200.0$, while the minimal value is 1.56 and it is the ration between the columns at $8 - 100.0$. On average the chains of Chimera are 2.09 times longer than those of Pegasus, therefore Pegasus returns chains 52% shorter.

Regarding the number of qubits, Pegasus is again the winner in terms of performance, it presents embedding with 1.39 up to 2.4 times less qubits than Chimera. On average Advantage is 1.96 times more efficient than 2000Q, or it means 51% less expensive in terms of qubits; this will be reflected on the quality of results. There are only two results slightly diverging from our expectations, that are the columns regarding the class $6-100$, we can see that they are shorter than they are supposed to be, this is due to the extreme complexity of the expressions, therefore since only the easiest expressions can be solved, the average is biased towards a lower value.

5.6 Search Strategies Comparisons

A short remark should be made on the way these algorithms run. Since the execution of the quantum annealers and simulated annealing are repeated many times in the attempt of finding the best solution, also the probabilistic and greedy techniques try to do the same.
Random search divides the solution space in N parallel processes and as long as it does not run out of time, it executes and records the maximal value found, up to that moment. In a similar way, grid search is completely exploring the solution space determined by the torsionals picked at random, with N parallel processes. According to [7] we expect grid search to perform poorly compared to random search even if they run with the same amount of resources allocated, when the size of the problem is still moderate. Since grid search will have to do a repeated execution, it will continue picking out random rotatable bonds of the molecule and it will solve the related problems as long as the time limit permits; in this case the single instance will be an intermediate solution on the total execution time span. Grid search creates instances with a number of at most 5 torsionals, since the exhaustive solution of problems with 6 rotatable bonds could exceed the total available time; the algorithm will be used for tackling only molecules of 10 rotatable bonds in order to highlight the limits of this method on moderately large spaces.
The Geodock-inspired search is trying an incremental approach, which consists in performing the rotation of an incremental number of bonds at a time, for example, if it is dealing with a problem on 4 rotatable bonds, it will search for the optimal value by rotating at first one bond at a time, until it has rotated each of the bond, for N times [1], then when it is done, it will record the solution found. It will start again looking for the best solution but this time by rotating two bonds at a time, or also said "in a batch of two" rotatables at a time, then again, but three plus one, and in the end all of the four bonds at the same time. The last iteration will be the most expensive since it is an

[1] parameter of the experiment

exhaustive search. In the first annex it is possible to see in the plot for 2 rotatable bonds the incremental improvement given by this strategy, with the distance simplification.

The plots in the Figures 5.6 are the average trends for all the molecules solved by each of the algorithms discussed so far, for 8 and 10 rotatable bonds on a window of 100 seconds, that is the X-axis.
For this type of plots on the Y-axis it is possible to find values from 0 to 1, as they are the percentages of the maximum volume reached by each algorithm, at each second. The values are the absolute values reached by an algorithm, averaged and normalized by the maximal average reached by any of the algorithms during a total available execution time cap of 1 hour. The plots demonstrate the algorithmic behaviour up until 100 seconds, since the greatest changes are in the first two minutes of the total time cap; it is not shown for the sake of clarity, but it is good to be aware that the maximum value is reached by the random search algorithm in all cases, this may be due to the large amount of resources used for a very long time. This is not the most efficient way of solving this type of problems, since in this work the key factor for the goodness of an algorithm is achieving high throughput for a low price, but we can consider it our "best benchmark", because past 5 rotatable bonds exhaustive solutions are almost impossible to achieve. We must point out that the probabilistic algorithms are running on the expression of the contributions made just from sines and cosines, while the annealing algorithms are running the QUBOs derived from them. The greedy algorithm is not using any expression, although all of them are working starting from the same modelling of the same problem.
For a better visual inspection, one can visit the first annex where it is possible to see that the increasing trends, by augmenting the number of degrees of freedom, are worsening in terms of the maximum value reached and scaling.

5.6.1 Runtime parameters

For the random search the only parameter is the time allocated; since the problem grows exponentially we could not assign exponential time for each rotatable bond added, after a tuning phase we decided to add 100 seconds for every bond considered, which is a good trade-off between quality of search and runtime, since even if for 10 bonds the theoretical maximum runtime is 1000 seconds, we had to take into account all the overheads of the python libraries. The Geodock-inspired algorithm has one parameter only, the number of times it should reiterate, or refine, his search on an instance with a certain batch of rotatables, which is three times; the angles' assignments are the sum of the discrete angles returned from each each iteration of the algorithm, the value of volume recorded is the best found over the three iterations. Grid search does not have any explicit parameter, it only has to pick an increasing number of bonds from 1 to 5, out of the 10 availables, and solve the related expression.

Regarding the set-ups of Advantage and 2000Q: since the QUBO has the constant A_const that must be tuned; we performed 10 runs for each value assigned to the constant. A_const is evaluated as the maximum coefficient in the optimization contribution multiplied by 5, then 10 and 20 for a total of 30 runs. Each run is performing ten thousands forward anneals with a linear schedule of one microsecond. Simulated annealing performs the same type of procedure on the QUBOs, but the anneals are controlled by the number of steps the algorithm can perform on the landscape, and there is the possibility of chosing the function with which to decrease the temperature of the simulation. In our case we used 500 steps and a geometrical decrease function.

5.6.2 Comparison between Advantage, Random Search and Grid Search

After just a graphical inspection, we can observe that Advantage is the best between the two DWAVE devices; if we scroll through the images in the annex I, for an increasing number of rotatable bonds, the level of volume achieved by 2000Q quickly dies off, while the results of Advantage remain competitive for a good number of torsionals.

A side note on the comparison between Advantage and the other algorithms concerns the results on the instances with 10 degrees of freedom. Since the creation of the HUBOs with maximum number of bonds rarely terminated and when it was possible, the embedding failed. In order to make up for the loss, and since the objective is improving molecules up to 10 torsionals, for the algorithms SA, Advantage and 2000Q we projected the results for the same molecules considered with only the 8 most internal bonds, together with the results of the other classical algorithms that solved the molecules with actually 10 bonds.

In all the instances random search, given the amount of resources allocated, performed with the best results, for both the versions with and without distance simplifications; Advantage and random search have the same null results at 0.1 seconds, which is null, but the parallel algorithm presents a disruptive improvement which comes always with a delay. It is mainly due to the overheads in the memory access to the expressions. On average at 1 second it doesn't provide any results, which leads Advantage to present volumes 28% closer to the benchmark of the instance, than those of the classical routine. At 10 seconds, the two get to a point where their difference is reduced to just 10.3% from the optimum; this amount of time gives to the QPU the time to start converging to its optimum. At 50 seconds, the random search beats Advantage for a 17% volume closer to the optimum. It is noticeable that 50 seconds is the time of activation for this algorithm. At a 100 seconds, we can consider the two of them in regime, and we can see that the random algorithm is beating the quantum annealer on average over all the problems with a volume on average, 16% closer to the maximum. With a simple average we could say that the volumes generated by the random search are just 0.96 times those of Advantage, but it is true only for the first 100 seconds, in fact if we treat the first 50 seconds as outliers, the volume of random search turns out to be 1.46 times the volume of Advantage.

At the bottom of Figure 5.6, you can see reported the trends of simulated annealing, Advantage and 2000Q for 8 rotatable bonds, normalized by the maximum achievable at 10. The peculiarity of grid search is the fast convergence to a stable value, which we can't tell immediately if it is the maximal value it can find. Although it is sampling from a totality of 10 bonds, the algorithm ends up with a very much similar scaling to Advantage, the value reached at 100 seconds is also similar, in fact grid search is on average only 1% better than Advantage. The fact that for both the algorithms the maximum value stops at around 30% at 50 seconds, makes us wonder if for certain classes of difficulty these two algorithms are computationally equivalent. It is to exclude the possibility that the last two bonds don't have an influence, since the volume of random search is on average actually 1.403 times greater than the one of grid.
The scalings for Advantage and grid search are slightly different, since grid search starts improving its results pretty early in the total execution, his growth is smoother, while Advantage ramps up in a later stage.

Table 5.1: Comparison on % volume gained for Grid and Advantage, with ratio and scaling.

seconds	GRID	ADV	GRID/ADV	ADV SCALE	GRID SCALE
0.1	0.0	0.0	-	-	-
1	0.01	0.0	-	-	-
10	0.16	0.09	1.77	-	16
50	0.31	0.27	1.14	3	1.93
100	0.31	0.3	1.03	1.11	1

5.6.3 Comparison between Advantage and GeoDock

In the plots in Figure 5.7 it is possible to start drawing some conclusions on the limits of the greedy algorithm.
On top we can see the black dotted line of the greedy algorithm, that has improvement of the percentage of volume logarithmic with time; this is due to the fact that first we always try to execute the greedy algorithm by considering the subproblems related to only one bond at a time, then two at a time, then three at a time and then 4, so as the time passes the solution of the general problem is attempted by solving subproblems of increasing complexity, which require more time for a volume gain that is not growing linearly with it. Small plateaus appear more often and in a longer measure than the volume improvements reported by the steps. Instead of having a linear scaling as we would wish from the dynamic programming method, we can see a curve usually starting between 20 and 30 seconds of execution. While for 2 torsionals the problem looks an easy task, with 4 torsionals we can foresee a deviation of the trends, the yellow line of 2000Q gets quickly closer to our greedy routine, until when dealing with 8 rotatable bonds, where Advantage takes the place of GeoDock-inspired and 2000Q becomes very similar in performance to this classical algorithm. The GeoDock-inspired algorithm is more than halving its maximal value while the number of rotable bonds is increasing. On average,

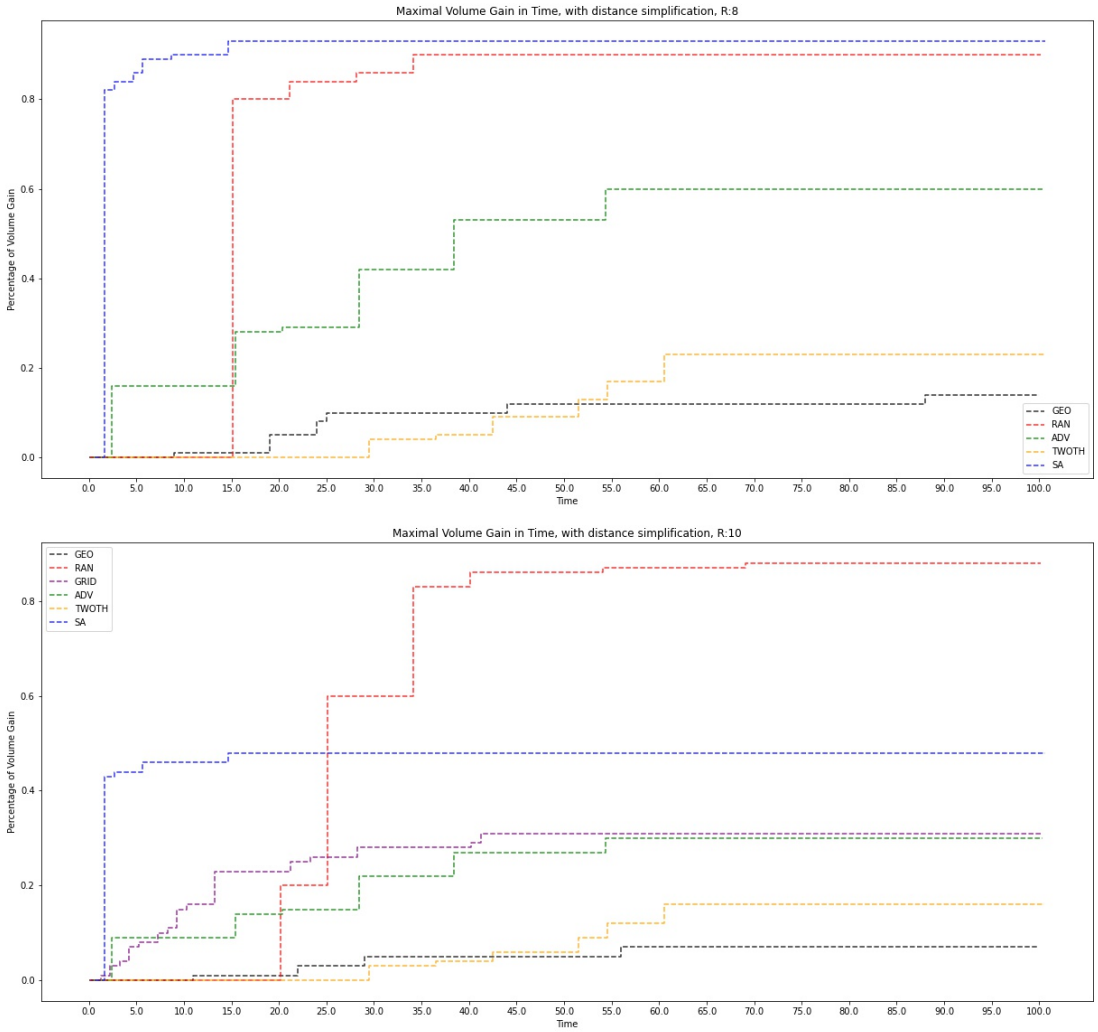

Figure 5.6: Comparison on a time window on 100 seconds, with distance simplification, for 8 and 10 rotatable bonds considered.

every time we increase the complexity of the problem, GeoDock loses 62% of its efficacy, its capability of reaching the maximum, in simple words of achieving volume 100% of the benchmark at the end of the time window. It is curious to point out the affinity between 2000Q and our version of the GeoDock expansion which, for an instance of 6 bonds, with and without simplification, they have close values of volume, although at 8 torsionals 2000Q completely fails the expansion, when the result retrieved with distance simplification is compared in the context of the solutions without this approximation.

Let us analyze the usage of the distance simplification.

Since many of the calculations have been done with the assumption that the solution with distance simplification could work also as a surrogate for the solutions without distance simplification, we need to verify how much in percentage are the trend losing when trying to reach the maximum value for a problem with and without this approxi-

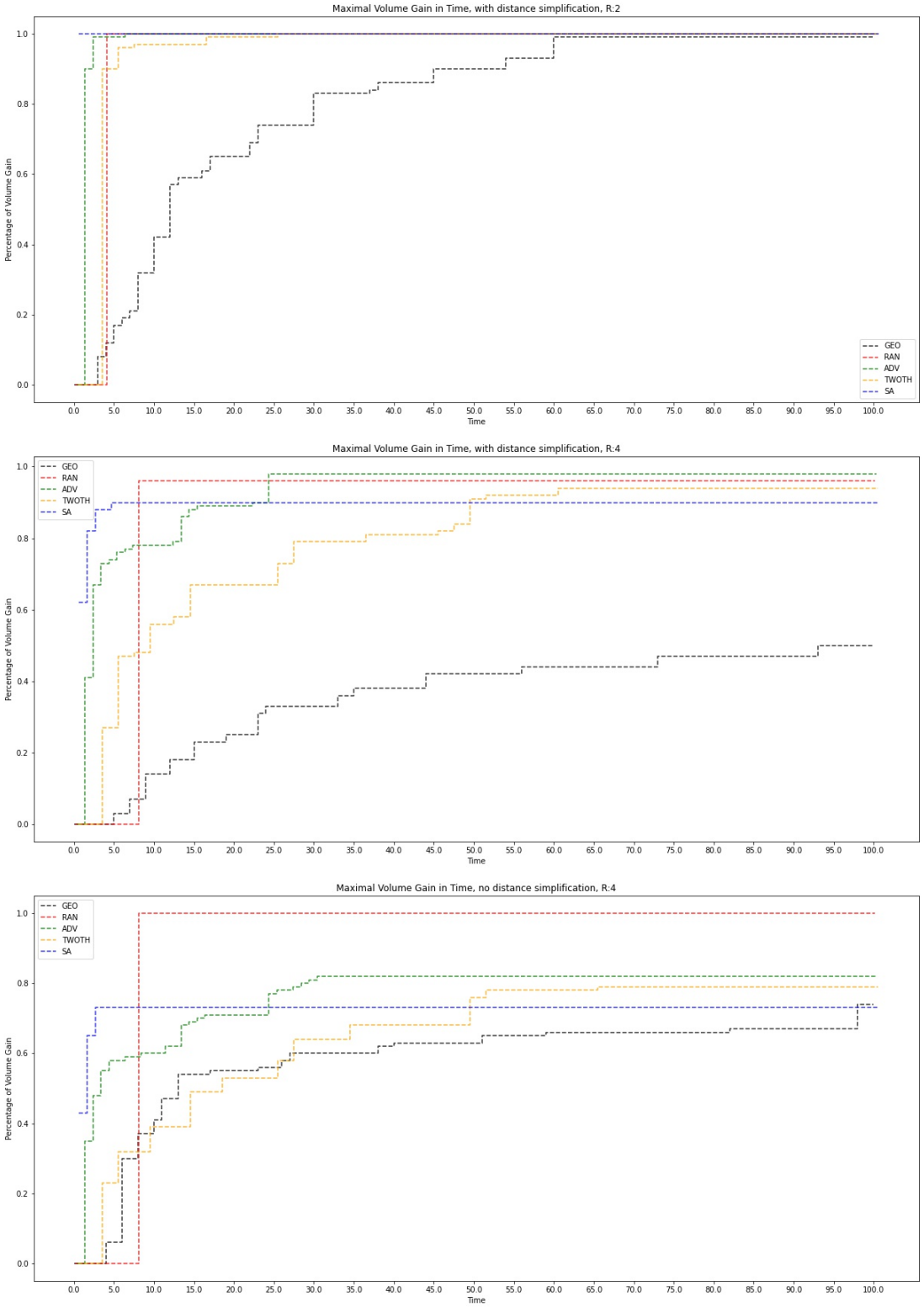

Figure 5.7: Comparison on a time window on 100 seconds, for 2 and 4 rotatable bonds considered, with and without the distance simplification for the second type.

mation. Therefore, we evaluated the solution of the problem simplified into the original problem and we traced the trends for all the annealing algorithms at 0.1, 1, 10 and 100 seconds, for all the problems of increasing difficulty. The average degradation of the distance from the 100% is of the 4% on average for Advantage while for SA it's of the 12%, this is a positive result in relation to the strong assumptions made when applying the simplification.
Meanwhile for algorithms like GeoDock and grid search the approximation was a worsening factor, it is possible to see that when their computation is driven by a more precise model the improvement in the maximum level of volume achieved happens to be 3.27 times better on average over all the instances, while for the grid search the final volume is 2,47 times greater.

5.6.4 Comparison between SA, Advantage and 2000Q

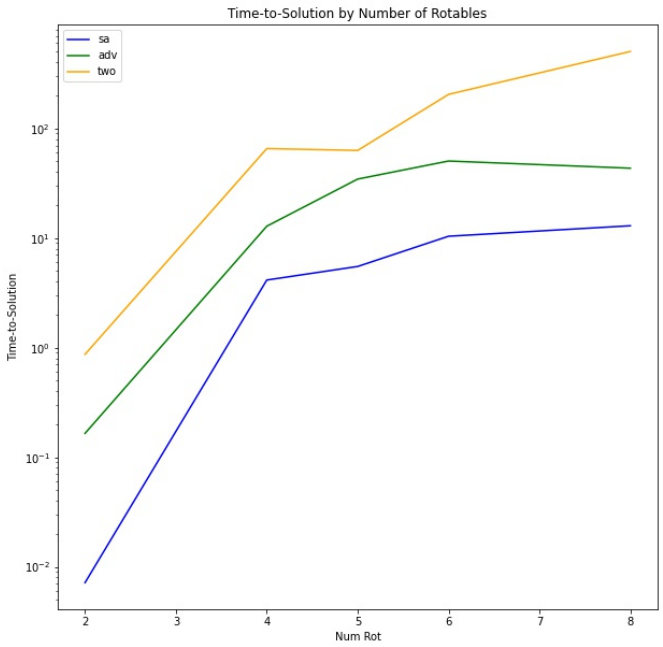

Figure 5.8: A comparison between Simulated Annealing, Advantage and 2000Q on TTSs, for increasing number of torsionals.

Comparing classical algorithms with quantum annealers in terms of absolute time isn't fair, since the quantum devices have runtimes, that are comprising of a programming time, a readout time and a delay time, besides the time spent in actually performing the annealing, all of this is called **total access time**; we need more suitable metrics for comparing the quantum annealers with at least other annealing implementations, which we will do with comparisons based on the TTS, the time to solution, which was described in the background section. The TTS is very high for 2000Q, then the second in order

is Advantage, while the best TTS is of simulated annealing. Simulated annealing has a TTS several times smaller than Advantage, and it means its convergence to the optimum is much faster than the quantum annealing method. Since the annealing schedules imposed to the three algorithms are almost identical, even if expressed in different terms, we can consider the numerator in the ratio of the TTS constant. The trend on the top of plot has the significance of being the one with the least number of appearances for the maximum value of the optimization problem, hence the most ineffective. The lowest trend is the most effective.

Table 5.2: Comparison on the TTS, for increasing number of torsionals.

torsionals	sa	adv	two
2	0.007171	0.165081	0.868552
4	4.138503	12.818871	65.482955
5	5.502274	34.508321	62.996198
6	10.384806	50.430784	204.457250
8	12.942243	43.363135	503.938826

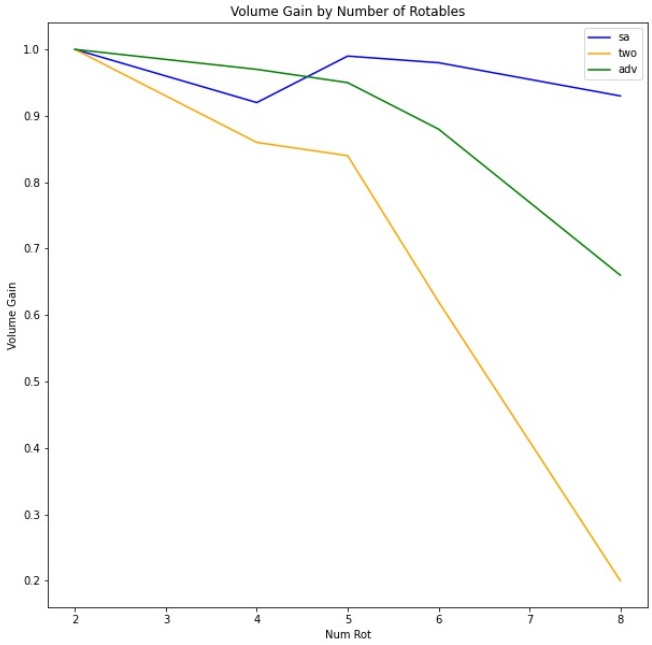

Figure 5.9: A comparison between Simulated Annealing, Advantage and 2000Q on volumes.

Regarding the volumes achieved, plots in Figure 5.9 represents the degradation of the final volume as the problem complexity arises. The volumes are the three decreasing trends that may seem normalized, but they have a common starting point since all the

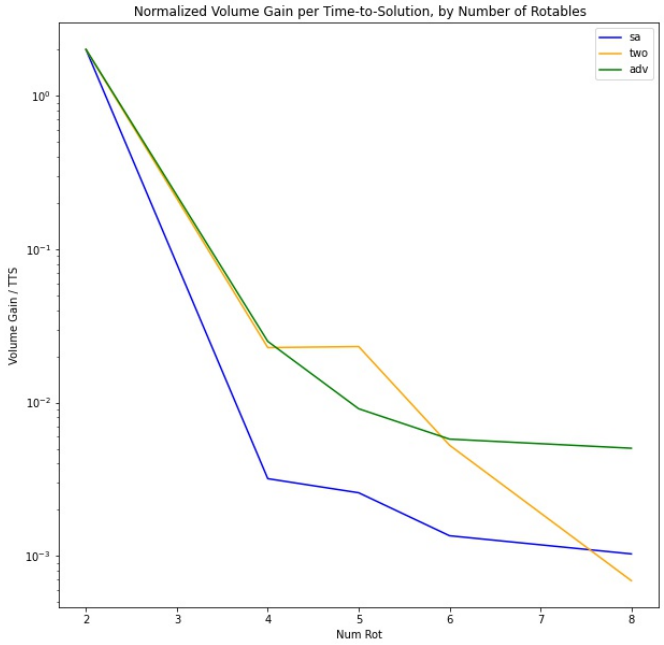

Figure 5.10: A comparison between Simulated Annealing, Advantage and 2000Q on the decreasing quality of the volumes, per time to solution.

three algorithm actually reach the maximum steadily and completely for problems on 2 rotatable bonds. SA has an inflection on 4 bonds but then recovers and finishes as the best in absolute terms; it has a volume which is 4.65 times better than the one of 2000Q at 8 bonds, and 1.4 times better than the result of Advantage.

Table 5.3: Comparison on the Volumes, for increasing number of torsionals.

torsionals	sa	two	adv
2	1.00	1.00	1.00
4	0.92	0.86	0.97
5	0.99	0.84	0.95
6	0.98	0.62	0.88
8	0.93	0.20	0.66

The last concept worth mentioning is what we would call velocity of the algorithm. It is the measure that tells us how much volume in percentage we can gain per one unit of time to solution; practically, it is computed as the ratio between the trends of the volumes and of the TTS of each algorithm at each number of variables considered.

The plot is composed of the segments that connect the values of velocity for each number of torsionals. Encountering a segment with a negative slope can be interpreted

as the diminishing amount of percentage of volume reachable by a certain algorithm when increasing the number of torsionals, or as the increasing time-to-solution required for achieving the same percentage of volume from scaling up to a more difficult problem. A higher negative slope means a faster worsening of the algorithm with the increasing difficulty, while a smaller slope is a synonym of stability.

It is clear that the trends that are running towards the bottom of the plot are belonging to the algorithms which are defective in one of the two measures, or more likely in both of them compared to the others.

In order to highlight the change in behaviour of the phenomenon when adding one torsional at a time, we normalized the y-axis by the range of each trend. What we end up with are some progressions which start from the same point, again not because of the normalizing factor, but because of the simplicity of problems with 2 rotatables.

The trend of SA falls down very fast, from 2 to 4 torsionals, and then starts becoming stable again. The same happens to Advantage which slows down this negative acceleration before. 2000Q never stops decreasing its quality, if not for a special case between 4 and 5 torsionals.

If we check the ratio on the slopes when increasing the number of torsionals, we can understand by how many factors the algorithms are worsening for each degree of freedom. The smaller the ratios, the higher is the probability for the algorithm to be in the upper part of the plot, hence being the best choice. On average the smallest growth in scaling of the slopes belongs to Advantage, with 1.36, whereas SA presents a higher 1.609, meanwhile 2000Q has an average of 6.01.

Although the absolute values of the results of Advantage may not be the winners, the fact that its velocity is much better than the one of SA, is a very promising result that reflects how much QA is immature, but also how much the improvements in the next hardware generations in the QPU access time would lower the time-to-solution, or it could tell us how much an improvement in the sampling factor of the QPU could improve the levels of volume achieved. The margin of growth for QA is very much far from saturation.

Table 5.4: Average normalized velocity's slopes scaling for the three annealing algorithms.

sa	two	adv
1.609948	6.0189816	1.36206524

Table 5.5: Normalized Velocity for each algorithm, at different number of rotatable bonds.

sa	two	adv	torsionals
2.001031	2.000690	2.005038	2
0.003190	0.022822	0.025046	4
0.002582	0.023171	0.009112	5
0.001354	0.005269	0.005776	6
0.001031	0.000690	0.005038	8

Chapter 6

Conclusions

In this thesis, we explored the possibility to use a quantum annealer to support the drug discovery process within the in-silico virtual screening phase. In particular, we studied a new method regarding the process of molecular unfolding, which is the first step in geometric molecular docking techniques. This phase aims at finding the molecule's configuration that maximizes its volume, or equivalently, that maximizes the internal distances of the atoms that compose the molecule. We proposed our Quantum Molecular Unfolding model with the aim of executing it on DWAVE's latest hardware, Advantage and 2000Q. The usage of the quantum annealers has the objective of understanding the capabilities of these quantum annealing devices and discovering if it is possible to improve the quality and the throughput of the ligand expansion in the state-of-art molecular docking methods.

The model is constructed starting from the identification of the rotatable bonds which are the parameters of the problem, and once their discrete rotations are rewritten with a one-hot encoding thanks to the introduction of binary variables, the total sum of the internal interatomic distances expressed in function of these variables will be the starting HUBO. The HUBO is then transformed into a QUBO, thanks to these three approximations: (i) coarse-grained rotations, (ii) landscape alternations by coefficients chopping, and (iii) fragments contraction, also called the distances simplification. The three approximations gave us the opportunity of reducing the complexity of the model to embed and run all our problems' instances on the two QPUs (Quantum Processing Units). The possibility of tuning the threshold for chopping and applying it multiple times, in different moments of the implementation, led us to the exploration of the capabilities of this method. We compared the performances of Advantage and 2000Q, with four other optimization methods, which are parallel random optimization, parallel grid search, simulated annealing, and a new greedy version of the original algorithm used in GeoDock.

The outcomes of the experiments show how classical techniques are still better than the quantum version, however provided an interesting inside on the evolution of the quantum annealer. Interestingly, the results obtained by the quantum annealer, both while running on Advantage and 2000Q, are better than the GeoDock-inspired algorithm, that is the currently adopted method. In terms of the evolution of the quantum

annealers, embedding our problems on Advantage, compared to 2000Q, was on average 51% less expensive in terms of qubits and with chains 52% shorter. Advantage significantly outperforms 2000Q in terms of time-to-solution and volume gain when increasing the number of torsionals. Advantage also presents the best scaling of the normalized velocity of the quality (volume gain over time-to-solution) when compared to both 2000Q and simulated annealing; this last result is one of the most promising as DWAVE devices are only bound to get better.

Completely unexpected results are related to the usage of the fragments contractions, or distance simplifications, thanks to which the average degradation of the distance from the 100% is of the 4% on average for Advantage while for SA it's of the 12% when we use the solutions retrieved with the approximation evaluated on the original problem. Meanwhile, for algorithms like GeoDock and grid search the approximation was a worsening factor, in fact, the maximum level of volume achieved happens to be 3.27 times better on average overall instances for the greedy routine, while for the grid search the final volumes are 2,47 times greater than the simplified problems.

Future work can be the introduction of new dynamic thresholds of chop, which may lead to smaller embeddings but still of high quality, and then the introduction of greedy approaches as a chance for testing the capabilities of the new frontier of hybrid quantum algorithms. A more difficult but also feasible and interesting task could be the study made of an extension of the model to the whole docking process, as other works have shown. We are confident that this study already sheds further light on the applications of quantum computing, thus enriching knowledge on topics that stand becoming central to the computational sciences.

Appendix A

Additional Results

A.I Additional Plots

In the next pages are shown the plots that represent the performance of the algorithms in a time window of 100 seconds.

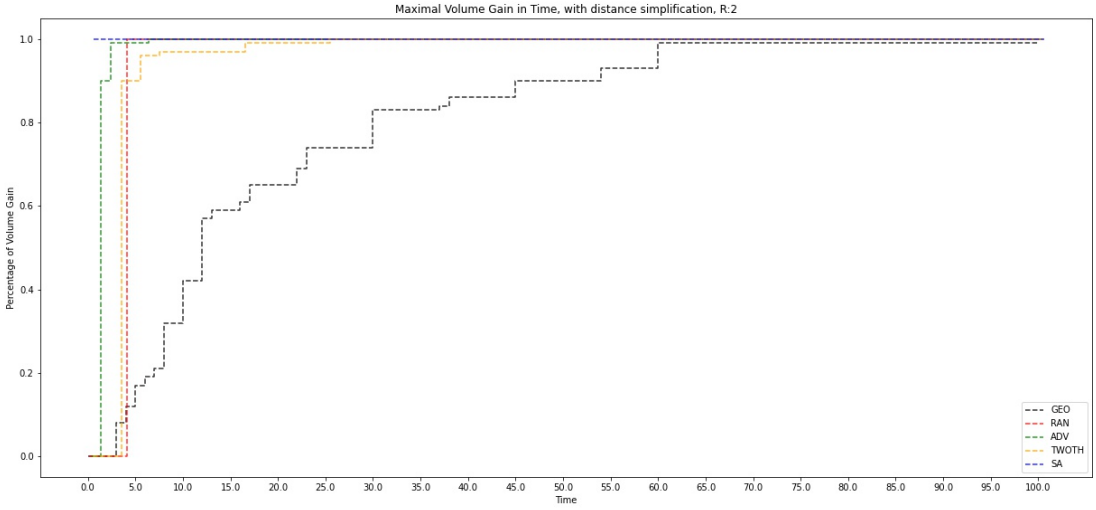

Figure A.1: Maximal Volume Gain in Time, with distance simplification. R:2

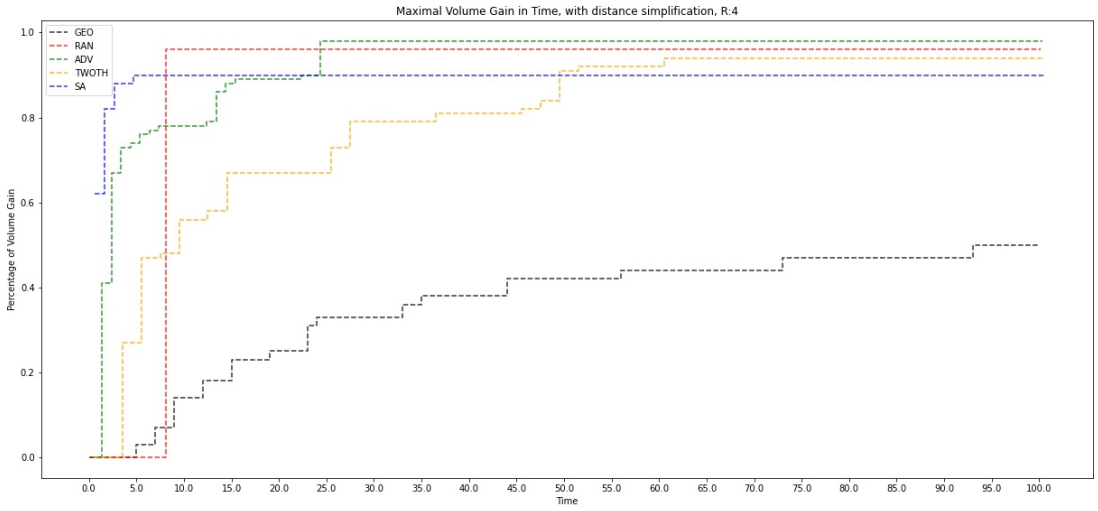

Figure A.2: Maximal Volume Gain in Time, with distance simplification, R:4

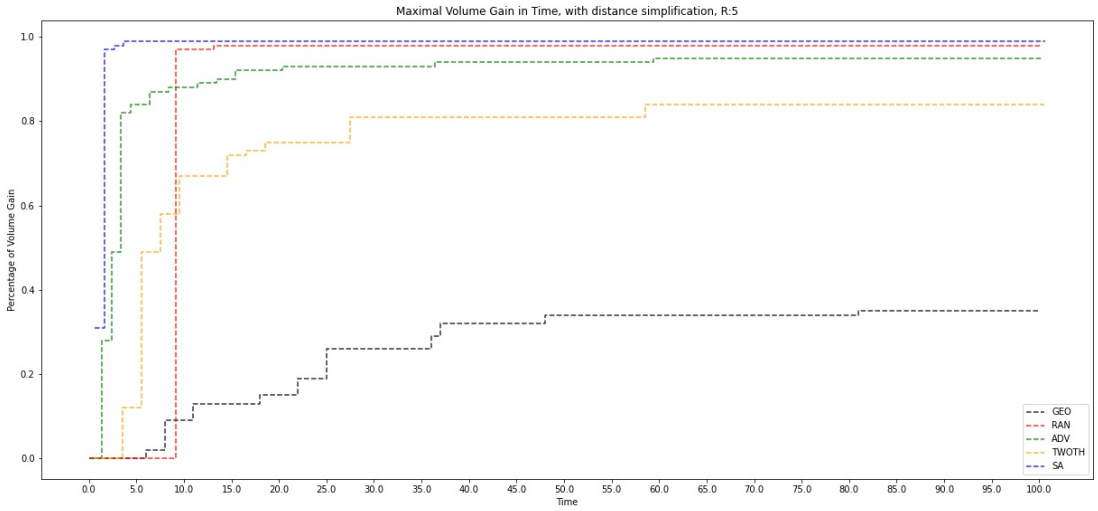

Figure A.3: Maximal Volume Gain in Time, with distance simplification, R:5

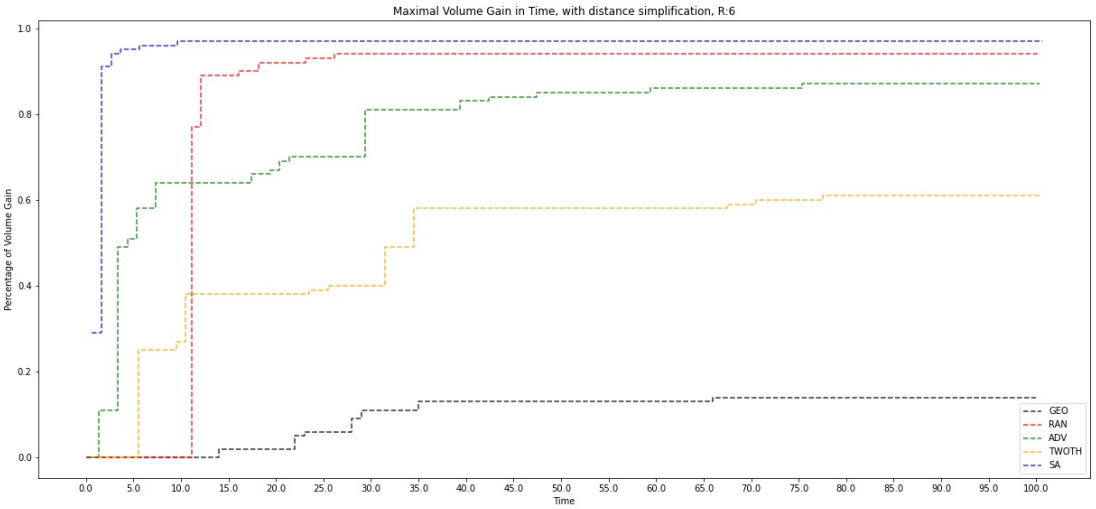

Figure A.4: Maximal Volume Gain in Time, with distance simplification, R:6

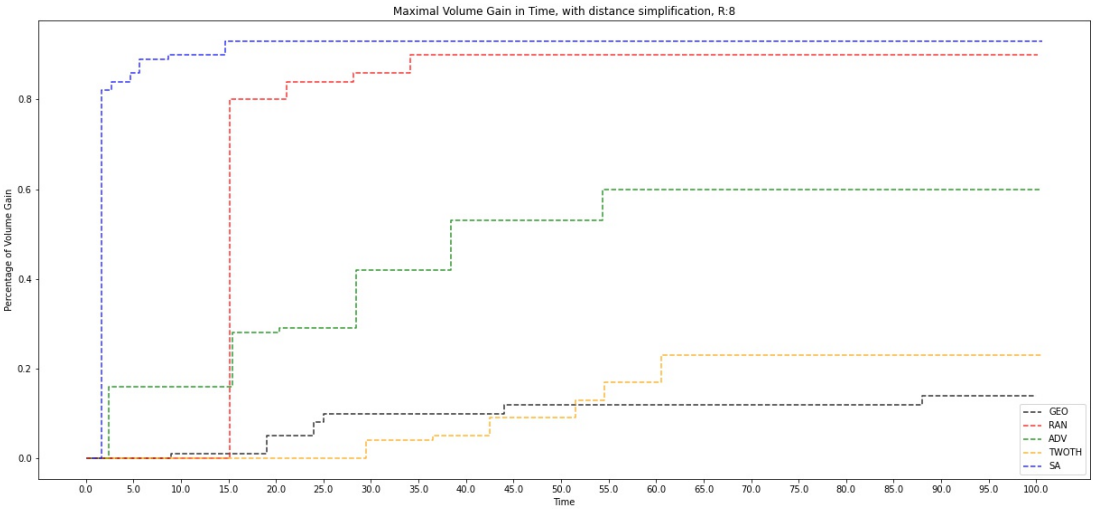

Figure A.5: Maximal Volume Gain in Time, with distance simplification, R:8

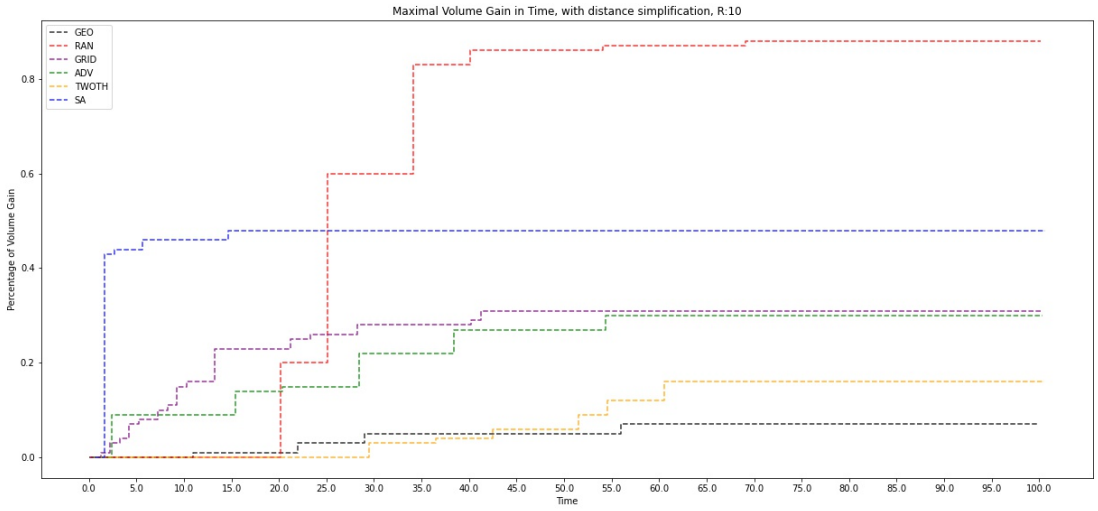

Figure A.6: Maximal Volume Gain in Time, with distance simplification, R:10

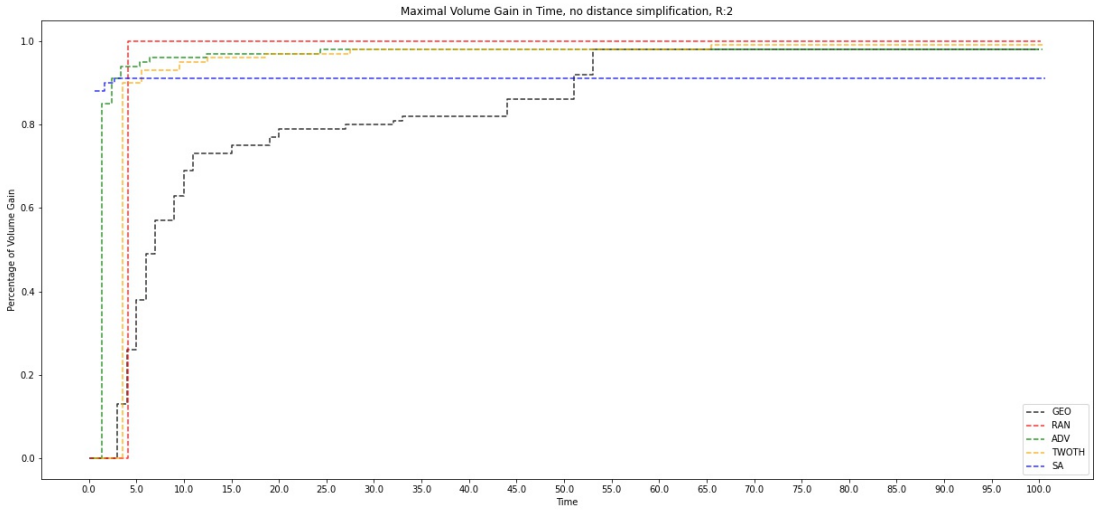

Figure A.7: Maximal Volume Gain in Time, no distance simplification, R:2

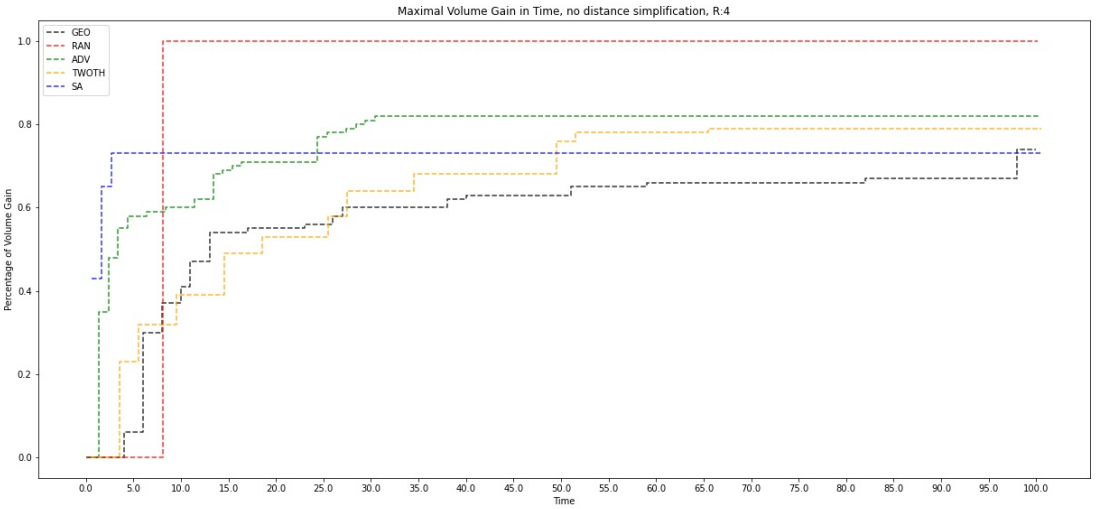

Figure A.8: Maximal Volume Gain in Time, no distance simplification, R:4

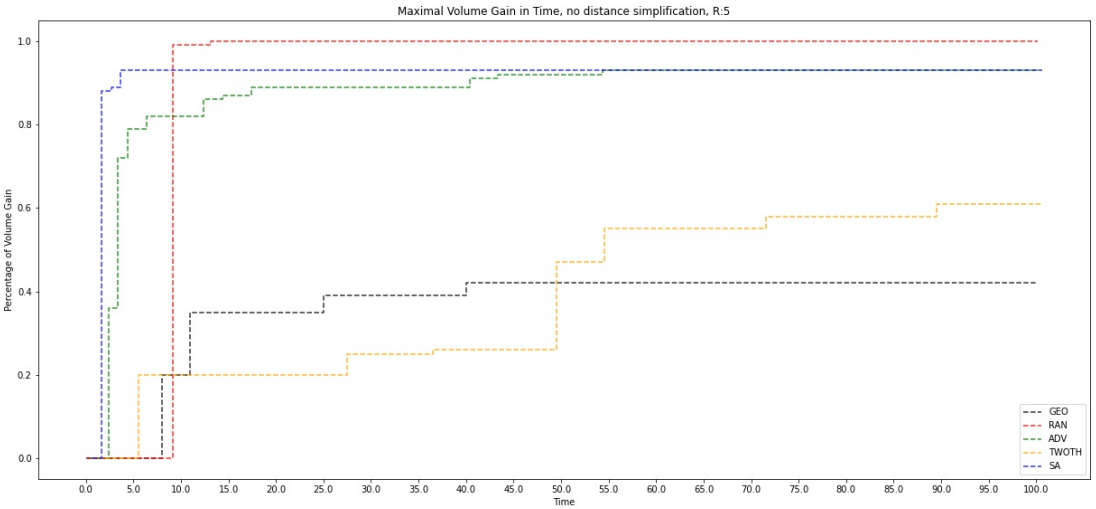

Figure A.9: Maximal Volume Gain in Time, no distance simplification, R:5

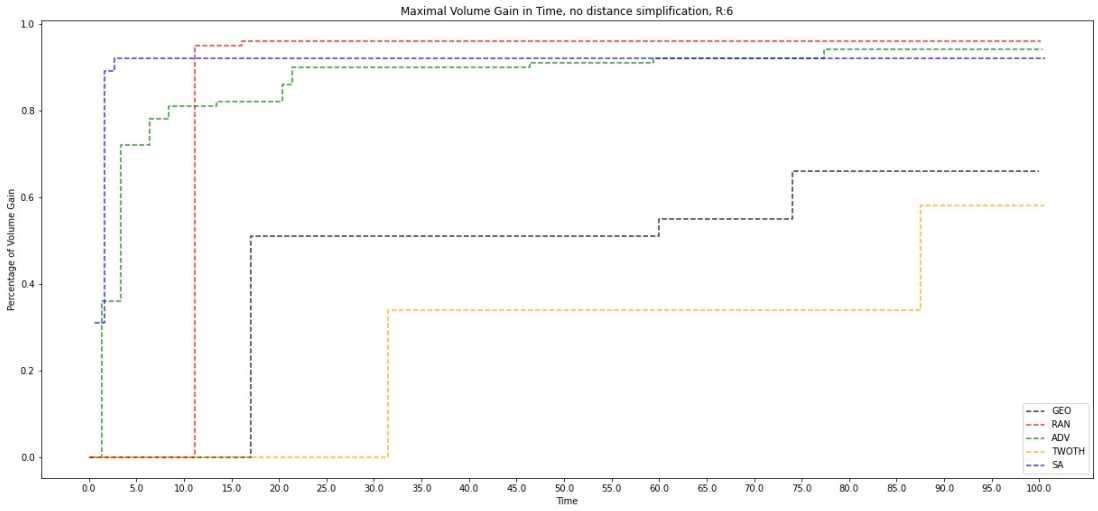

Figure A.10: Maximal Volume Gain in Time, no distance simplification, R:6

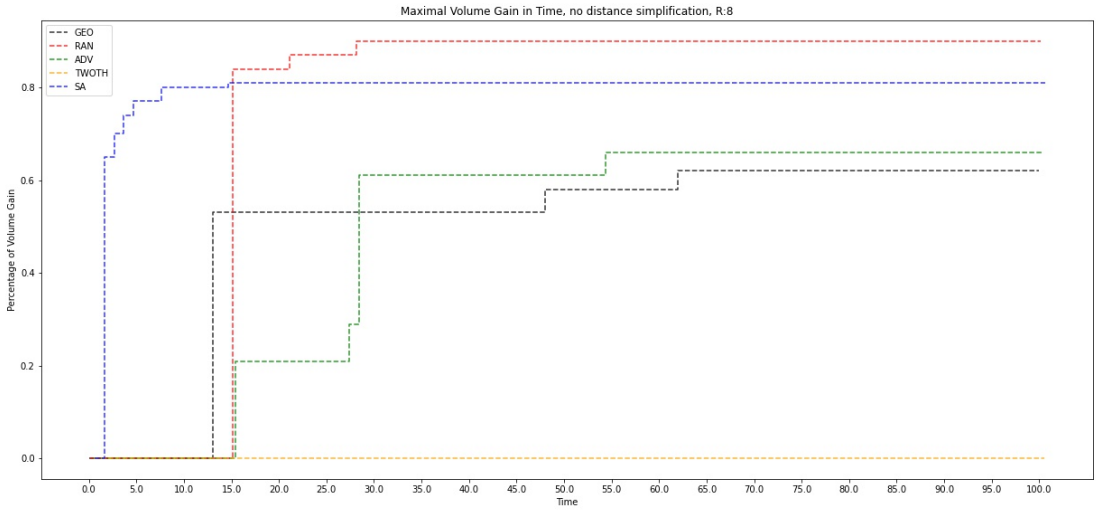

Figure A.11: Maximal Volume Gain in Time, no distance simplification, R:8

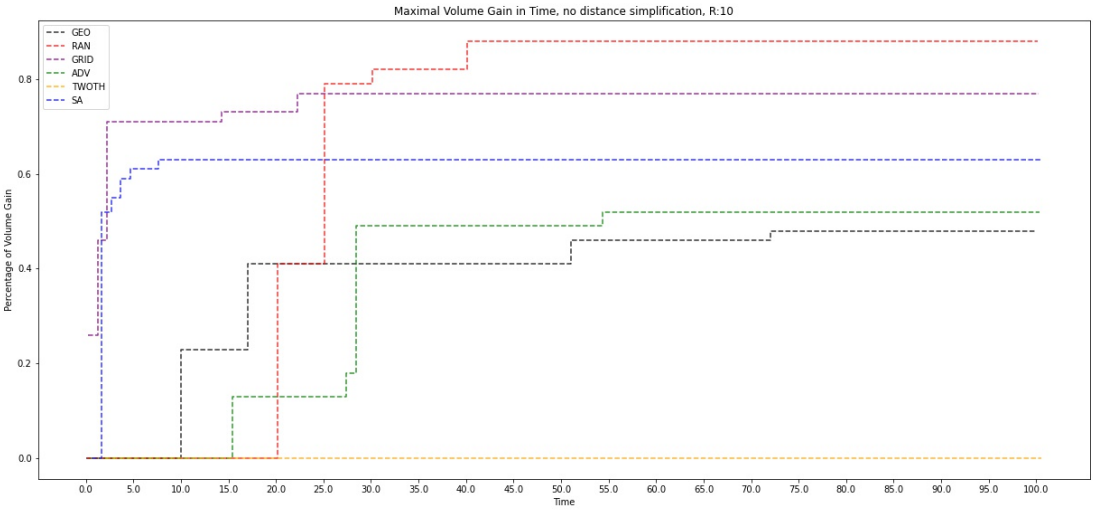

Figure A.12: Maximal Volume Gain in Time, no distance simplification, R:10

A.II Additional Tables

The tables in this second annex are additional data that represent the measured volume gained in percentage achieved by each algorithm, at defined time checkpoints, for problems with a certain number of torsionals. For each number of rotatable bonds we reported the same information regarding the runs made with, and without the distances simplification, with the objective of giving a deeper insight on the results.

Table A.1: % volume gained, no distances simplification, for 2 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.88	0.00	0.0	0.0	0.00	0.1
0.85	0.90	0.00	0.0	0.0	0.00	1.0
0.96	0.91	0.69	0.0	1.0	0.95	10.0
0.98	0.91	0.86	0.0	1.0	0.98	50.0
0.98	0.91	0.98	0.0	1.0	0.99	100.0

Table A.2: % volume gained, with distances simplification, for 2 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.0	1.0	0.00	0.0	0.0	0.00	0.1
0.9	1.0	0.00	0.0	0.0	0.00	1.0
1.0	1.0	0.42	0.0	1.0	0.97	10.0
1.0	1.0	0.90	0.0	1.0	1.00	50.0
1.0	1.0	0.99	0.0	1.0	1.00	100.0

Table A.3: % volume gained, no distances simplification, for 4 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.43	0.00	0.0	0.0	0.00	0.1
0.35	0.65	0.00	0.0	0.0	0.00	1.0
0.60	0.73	0.41	0.0	1.0	0.39	10.0
0.82	0.73	0.63	0.0	1.0	0.76	50.0
0.82	0.73	0.74	0.0	1.0	0.79	100.0

Table A.4: % volume gained, with distances simplification, for 4 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.62	0.00	0.0	0.00	0.00	0.1
0.41	0.82	0.00	0.0	0.00	0.00	1.0
0.78	0.90	0.14	0.0	0.96	0.56	10.0
0.98	0.90	0.42	0.0	0.96	0.91	50.0
0.98	0.90	0.50	0.0	0.96	0.94	100.0

Table A.5: % volume gained, no distances simplification, for 5 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.00	0.00	0.0	0.00	0.00	0.1
0.00	0.88	0.00	0.0	0.00	0.00	1.0
0.82	0.93	0.20	0.0	0.99	0.20	10.0
0.92	0.93	0.42	0.0	1.00	0.47	50.0
0.93	0.93	0.42	0.0	1.00	0.61	100.0

Table A.6: % volume gained, with distances simplification, for 5 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.31	0.00	0.0	0.00	0.00	0.1
0.28	0.97	0.00	0.0	0.00	0.00	1.0
0.88	0.99	0.09	0.0	0.97	0.67	10.0
0.94	0.99	0.34	0.0	0.98	0.81	50.0
0.95	0.99	0.35	0.0	0.98	0.84	100.0

Table A.7: % volume gained, no distances simplification, for 6 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.31	0.00	0.0	0.00	0.00	0.1
0.36	0.89	0.00	0.0	0.00	0.00	1.0
0.81	0.92	0.00	0.0	0.00	0.00	10.0
0.91	0.92	0.51	0.0	0.96	0.34	50.0
0.94	0.92	0.66	0.0	0.96	0.58	100.0

Table A.8: % volume gained, with distances simplification, for 6 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.29	0.00	0.0	0.00	0.00	0.1
0.11	0.91	0.00	0.0	0.00	0.00	1.0
0.64	0.97	0.00	0.0	0.00	0.38	10.0
0.85	0.97	0.13	0.0	0.94	0.58	50.0
0.87	0.97	0.14	0.0	0.94	0.61	100.0

Table A.9: % volume gained, no distances simplification, for 8 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.00	0.00	0.0	0.0	0.0	0.1
0.00	0.65	0.00	0.0	0.0	0.0	1.0
0.00	0.80	0.00	0.0	0.0	0.0	10.0
0.61	0.81	0.58	0.0	0.9	0.0	50.0
0.66	0.81	0.62	0.0	0.9	0.0	100.0

Table A.10: % volume gained, with distances simplification, for 8 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.00	0.00	0.0	0.0	0.00	0.1
0.00	0.82	0.00	0.0	0.0	0.00	1.0
0.16	0.90	0.01	0.0	0.0	0.00	10.0
0.53	0.93	0.12	0.0	0.9	0.09	50.0
0.60	0.93	0.14	0.0	0.9	0.23	100.0

Table A.11: % volume gained, no distances simplification, for 10 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.00	0.00	0.26	0.00	0.0	0.1
0.00	0.52	0.00	0.46	0.00	0.0	1.0
0.00	0.63	0.23	0.71	0.00	0.0	10.0
0.49	0.63	0.41	0.77	0.88	0.0	50.0
0.52	0.63	0.48	0.77	0.88	0.0	100.0

Table A.12: % volume gained, with distances simplification, for 10 rotatables.

ADV	SA	GEO	GRID	RAN	TWO	seconds
0.00	0.00	0.00	0.00	0.00	0.00	0.1
0.00	0.43	0.00	0.01	0.00	0.00	1.0
0.09	0.46	0.00	0.16	0.00	0.00	10.0
0.27	0.48	0.05	0.31	0.86	0.06	50.0
0.30	0.48	0.07	0.31	0.88	0.16	100.0

List of Figures

List of Tables

Bibliography

[1] Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, July 1985.

[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 Journal on Computing*, 37(1):166–194, January 2007.

[3] Tameem Albash and Daniel A Lidar. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Physical Review X*, 8(3):031016, 2018.

[4] Tomás Babej, Christopher Ing, and Mark Fingerhuth. Coarse-grained lattice protein folding on a quantum annealer. *arXiv: Quantum Physics*, 2018.

[5] Leonardo Banchi, Mark Fingerhuth, Tomas Babej, Christopher Ing, and Juan Miguel Arrazola. Molecular docking with gaussian boson sampling. *Science Advances*, 6(23):eaax1950, June 2020.

[6] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, November 1954.

[7] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13, 2012.

[8] Zhengbing Bian, Fabián A. Chudak, W. Macready, and G. Rose. The ising model : teaching an old problem new tricks. 2010.

[9] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors. *arXiv preprint arXiv:2003.00133*, 2020.

[10] Jun Cai, William G Macready, and Aidan Roy. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741*, 2014.

[11] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. MPI for python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, September 2005.

[12] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, September 2011.

[13] Nike Dattani, Szilard Szalay, and Nick Chancellor. Pegasus: The second connectivity graph for large-scale quantum annealing hardware, 2019.

[14] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3), August 2016.

[15] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[16] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[17] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35, March 1977.

[18] Davide Gadioli, Gianluca Palermo, Stefano Cherubin, Emanuele Vitali, Giovanni Agosta, Candida Manelfi, Andrea R. Beccari, Carlo Cavazzoni, Nico Sanna, and Cristina Silvano. Tunable approximations to control time-to-solution in an HPC molecular docking mini-app. *The Journal of Supercomputing*, 77(1):841–869, April 2020.

[19] Fred W. Glover, G. Kochenberger, and Yu Du. Quantum bridge analytics i: a tutorial on formulating and using qubo models. *4OR*, 17:335–371, 2019.

[20] Itay Hen and A. P. Young. Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems. *Physical Review E*, 84(6), December 2011.

[21] H. Ishikawa. Transformation of general binary mrf minimization to the first-order case. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1234–1249, 2011.

[22] James King, Sheir Yarkoni, Jack Raymond, Isil Ozfidan, Andrew D King, Mayssam Mohammadi Nevisi, Jeremy P Hilton, and Catherine C McGeoch. Quantum annealing amid local ruggedness and global frustration. *Journal of the Physical Society of Japan*, 88(6):061007, 2019.

[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[24] Douglas B. Kitchen, Hélène Decornez, John R. Furr, and Jürgen Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature Reviews Drug Discovery*, 3(11):935–949, November 2004.

[25] Irwin D. Kuntz, Jeffrey M. Blaney, Stuart J. Oatley, Robert Langridge, and Thomas E. Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology*, 161(2):269–288, October 1982.

[26] T. Lanting, A. J. Przybysz, A. Yu. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose. Entanglement in a quantum annealing processor. *Physical Review X*, 4(2), May 2014.

[27] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning - ICML '07*. ACM Press, 2007.

[28] Evanthia Lionta, George Spyrou, Demetrios Vassilatis, and Zoe Cournia. Structure-based virtual screening for drug discovery: Principles, applications and recent advances. *Current Topics in Medicinal Chemistry*, 14(16):1923–1938, October 2014.

[29] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, July 2019.

[30] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[31] Paul D Lyne. Structure-based virtual screening: an overview. *Drug Discovery Today*, 7(20):1047–1055, October 2002.

[32] Avradip Mandal, Arnab Roy, Sarvagya Upadhyay, and Hayato Ushijima-Mwesigwa. Compressed quadratization of higher order binary optimization problems. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*. ACM, May 2020.

[33] D. J. J. Marchand, M. Noori, A. Roberts, G. Rosenberg, B. Woods, U. Yildiz, M. Coons, D. Devore, and P. Margl. A variable neighbourhood descent heuristic for conformational search using a quantum annealer. *Scientific Reports*, 9(1), September 2019.

[34] Xuan-Yu Meng, Hong-Xing Zhang, Mihaly Mezei, and Meng Cui. Molecular docking: A powerful approach for structure-based drug discovery. *Current Computer Aided-Drug Design*, 7(2):146–157, June 2011.

[35] Riccardo Mengoni, Daniele Ottaviani, and Paolino Iorio. Breaking rsa security with a low noise d-wave 2000q quantum annealer: Computational times, limitations and prospects. *arXiv preprint arXiv:2005.02268*, 2020.

[36] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[37] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12):125210, 2008.

[38] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2009.

[39] Fabio Pietrucci. Strategies for the exploration of free energy landscapes: Unity in diversity and challenges ahead. *Reviews in Physics*, 2:32–45, November 2017.

[40] Sebastian Raschka. Biopandas: Working with molecular structures in pandas dataframes. *The Journal of Open Source Software*, 2(14), jun 2017.

[41] Irmi Sax, Sebastian Feld, Sebastian Zielinski, Thomas Gabor, Claudia Linnhoff-Popien, and Wolfgang Mauerer. Approximate approximation on a quantum annealer. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*. ACM, May 2020.

[42] Wolfgang Scherer. *Mathematics of Quantum Computing*. Springer International Publishing, 2019.

[43] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. Simulating physical phenomena by quantum networks. *Physical Review A*, 65(4), April 2002.

[44] Davide Venturelli, Dominic JJ Marchand, and Galo Rojo. Quantum annealing implementation of job-shop scheduling. *arXiv preprint arXiv:1506.08479*, 2015.

[45] Emanuele Vitali, Davide Gadioli, Gianluca Palermo, Andrea Beccari, Carlo Cavazzoni, and Cristina Silvano. Exploiting openmp and openacc to accelerate a geometric approach to molecular docking in heterogeneous hpc nodes. *J. Supercomput.*, 75(7):3374–3396, 2019.