UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Solving combinatorial optimization problems on D-Wave annealers

Relatore

Prof. Simone MONTANGERO

Laureando

Gianluca MORETTINI

Anno Accademico 2020/2021

Contents

1 Introduction

Building a quantum computer is a challenge that humanity has been dealing with for two decades. Some technologies based on quantum mechanics have been developed and, among them, there are the D-Wave annealers.
The goal of the thesis is to explore the whole process needed to solve an optimization problem on a D-Wave annealer, from the mathematical formulation to the final implementation. Furthermore, results obtained with quantum annealing (QA) will be compared to those provided by simulated annealing (SA). This will be useful to understand what is the state-of-art regarding the performances of these machines with respect to standard classical methods.
In the first chapter, an introduction to QA, its physical background and how it is implemented on D-Wave is presented.
In the second chapter, a description of how to solve a problem on the D-Wave annealers is provided. In particular, the focus will be on how to rewrite an optimization problem using Quadratic Unconstrained Binary Optimization (QUBO) or Ising problem formulation. In addition to that, there will be also a section concerning the embedding of the problem on the D-Wave hardware.
In the last chapter, an application to the graph colouring problem is presented.

2 Quantum annealing

2.1 Physical backgroud

Quantum annealing is a process that can be applied to find the global maximum or minimum of an objective function. It was proposed for the first time in 1998 by Tadashi Kadowaki and Hidetoshi Nishimori [1]. They developed the process starting from the classical analogue, where thermal fluctuations play the role of quantum fluctuation induced by quantum tunnelling.
The optimization goal is to find the ground state of a Hamiltonian that represents the problem to solve with high accuracy. The quality of the solutions is typically influenced by the annealing schedule, namely how quantum fluctuations are introduced in the system. The core of the algorithm lays in the adiabatic theorem [2]:

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

Optimization through quantum annealing begins by preparing the system with minimum energy. In these conditions, the landscape of energy is represented by a function simple to manage, depending on the technical features of the hardware. The annealing process consists of slow modifications of the simple function that gradually changes its shape into the objective function, a function that represents the problem to solve. The process finishes when the energy landscape of the system represents the objective function. If the annealing takes place slowly enough, the adiabatic theorem assures that in all the transformation phases of the function, the state of the system adapts itself to the shape of the function. Therefore, at the end of the annealing, the system has the minimum energy. The obtained state is a solution to the problem: may be present more than one state corresponding at the minimum energy.

2.2 The D-Wave annealer

The D-Wave machine is not a universal quantum computer, but rather, a more limited "quantum annealer" [8]: a classical model (SSSV model) can simulate the evolution of coherent single qubits without multi-qubit coherence, and in particular without any entanglement. Thus how much quantum is the D-Wave remains a controversial question [9]. However, its processor consists of a 2D array of quantum bits, or qubits, made of superconducting loops that carry electric currents (Fig. 2). The qubits act like tiny magnets that can point up, down, or, thanks to quantum weirdness, both up and down at the same time. Each qubit can interact with certain others through linkers that can be programmed by the user. Qubits can lower their energy by pointing either in the same direction or in opposite directions [8]. More in detail, the probability of falling in a particular state after a measurement is manipulated through external magnetic fields. The programmable quantity that controls the external field is called bias, and each qubit minimizes its energy in the presence of the bias. Qubits are not independent but they are connected through devices called couplers [3].
Both the number of qubits and couplers in the system depends on the version of the D-Wave annealer. In this work, two different machines are considered: DW_2000Q_6 and Advantage_system1.1.
The following Hamiltonian represents the QA in D-Wave annealer:

$$H = -\frac{A(s)}{2}\sum_i \sigma_x^i + \frac{B(s)}{2}\left(\sum_i h_i\sigma_z^i + \sum_{i>j} J_{ij}\sigma_z^i\sigma_z^j\right) \tag{1}$$

where $\sigma_{x,z}^i$ are Pauli matrices operating on the i^{th} qubit, h_i and J_{ij} are the qubit biases and coupling strengths and s=$\frac{t}{t_f}$ where t is time and t_f is the total time of the annealing process. J_{ij} is present when the two qubits are connected: in Eq. (1), the sum $\sum_{i>j}$ considers ideal connections where each qubit interacts with all the others. In general, this is difficult to have and building different topologies, schemes in which qubits are connected, are needed. Further details will be explained in Section 3.3.
The Hamiltonian is composed of two terms: the first one regards the initial state of the set of qubits before annealing starts, the second part of the Hamiltonian describes the problem to solve. A(s) and B(s) are time-dependent parameters that drive the annealing. As shown in Fig. 1, the former monotonically decreases during the QA from an initial value to 0, while the latter monotonically increases from 0 to a maximum value.
When the annealing begins, the system starts in the lowest energy state, which is well separated from any other energy level. As the problem Hamiltonian is introduced, other energy levels may get closer to the ground state. It might happen that the system jumps from the lowest energy state into one of the excited states during the annealing. The closer they get, the higher the probability of the jump is. The main causes related to this fact are annealing speed and the coupling of the quantum processing unit (QPU) with the external environment [3]. In the end, these factors might deteriorate the quality of the final solution.

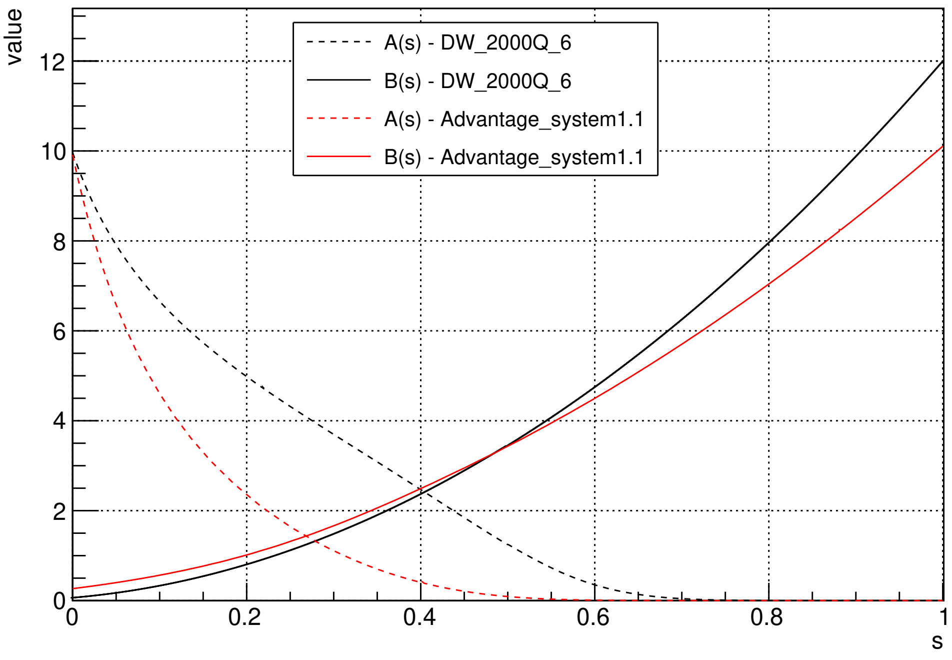

Figure 1: Parameters A(s) and B(s) versus s of DW_2000Q_6 and Advantage_system1.1 annealers [4,5]; they will be used in the last chapter.

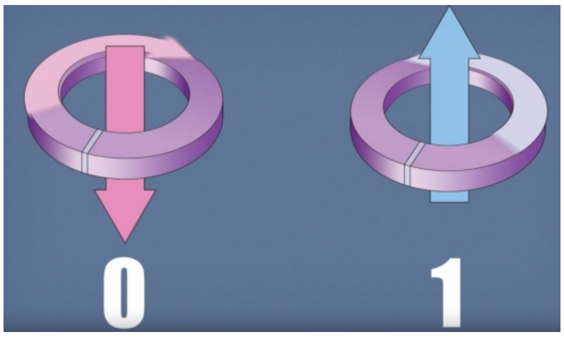

Figure 2: Representation of the superconducting loops that determine qubit states [3].

3 Combinatorial optimization on D-Wave annealers

3.1 The Ising/QUBO formulation

A combinatorial optimization problem is a discrete optimization problem whose solution is found among a finite number of possible solutions. An objective function is necessary to express a general problem in a form that allows its resolution through a D-Wave annealer. It is a mathematical

expression of the energy of a system and represents the problem to solve. The problem is expressed, and it is always possible, in the form of a minimization problem.
The objective functions accepted by D-Wave annealers are of two equivalent ones: Ising Hamiltonians and QUBO formulations [3].

- The Ising model is a well-known model in statistical mechanics. It is a quadratic and binary model describing a lattice of N sites each one endowed with a degree of freedom called spin. Each spin can be in two different configurations: spin up (+1) and spin down (-1). The Ising Hamiltonian has the following expression:

$$H_{Ising} = \sum_i h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij} s_i s_j \tag{2}$$

  where h_i is the magnetic field and J_{ij} the two-spin interaction, as already been explained in Eq. (1).

- A QUBO (Quadratic Unconstrained Binary Optimization) problem is defined by an $N \times N$ upper real triangular matrix Q and a vector of binary variables $\mathbf{x}$. Each binary variable can have either value 0 or +1. A generic QUBO Hamiltonian is expressed as follows:

$$H_{QUBO} = \sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j \tag{3}$$

  where the diagonal terms of the matrix Q play the role of linear terms while the other non-zero elements are the quadratic terms. In matrix form:

$$\min_{\mathbf{x} \in [0,1]^N} \mathbf{x}^{\mathsf{T}} \mathbf{Q} \mathbf{x} \tag{4}$$

It is possible to map the Ising to the QUBO formulation with the transformation $x_i \mapsto s_i := 2x_i - 1$.

3.2 Constrained problems

In general, the solution of a combinatorial optimization problem has to satisfy some constraints. This implies the presence of penalty functions in QUBO or Ising formulation to make unwanted solutions less probable. In particular, if a given solution does not satisfy some constraints, the penalty function adds a positive quantity to the value of the function, increasing it. Thus, it brings the state to a higher energy value moving it away from the global minimum and making it less probable.
Two examples of constraints that appear very frequently in combinatorial optimization and will be used in the last chapter in this work are:

1. The constraint $\sum_{i=0}^{N} x_i = C$, where C is a constant and x_i is a QUBO variable, can be imposed via the minimization:

$$\min_{x_i} \left(\sum_{i=0}^{N} x_i - C \right)^2 \tag{5}$$

   where:

$$\left(\sum_{i=0}^{N} x_i - C \right)^2 = \left(\sum_{i=0}^{N} x_i \right)^2 - 2C \sum_{i=0}^{N} x_i + C^2 \Longrightarrow \left(\sum_{i=0}^{N} x_i \right)^2 - 2C \sum_{i=0}^{N} x_i \tag{6}$$

where we neglect the term C^2 being an offset in a minimization problem. Finally,

$$\left(\sum_{i=0}^{N} x_i\right)^2 - 2C\sum_{i=0}^{N} x_i = \sum_{i=0}^{N} x_i^2 + 2\sum_{i<j} x_i x_j - 2C\sum_{i=0}^{N} x_i \overset{(2.)}{=} \tag{7}$$

$$= \sum_{i=0}^{N} x_i + 2\sum_{i<j} x_i x_j - 2C\sum_{i=0}^{N} x_i = \sum_{i=0}^{N}(1-2C)x_i + 2\sum_{i<j} x_i x_j \tag{8}$$

where passage (2.) is valid as $x_i^2 = x_i$ for $x_i = 0, 1$.

2. $\sum_{i=0}^{N} x_i \leq C$. A way to introduce it is to reduce the inequality in a equality. C additional binary variables are needed, indeed:

$$\sum_{i=0}^{N} x_i \leq C \Longrightarrow \sum_{i=0}^{N} x_i = C - \sum_{k=0}^{C} x_K \Longrightarrow \sum_{i=0}^{N+C} x_i = C. \tag{9}$$

In the end, both constraints are introduced minimizing:

$$\min_{x_i}\left(P\left(\sum_{i=0}^{L} x_i - C\right)^2\right). \tag{10}$$

As already seen, the scalar P, L and C depend on the considered constraint. In summary, in Table 1 [6] the principal penalty functions are:

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

Table 1: Constraints and relative penalty functions. P is a real number decided by the user, depending on the problem to solve and on the solution; x and y are QUBO variables.

3.3 Minor embedding

The next step after the formulation of the QUBO or Ising problem is to map it into the D-Wave qubits. At the time of the present study, two different topologies were available: Chimera graph and Pegasus graph. More details about them will be provided at the end of this section.
An undirected graph is defined as a set of vertices and a set of edges. Each node and each edge is weighted with an arbitrary value. Vertices are the qubits, and edges are the couplers among them. The weights of the vertex are the linear terms that compose H_{QUBO} or H_{Ising}, while the weights of the edge are the quadratic ones.
In general, the graph used to represents the QUBO problem does not have enough vertices or edges. If the number of vertices is not enough, a new reformulation of the problem or a new type of graph is needed.

If the number of edges is insufficient, mathematical and physical operations are feasible: let the Hamiltonian $H_{example} = 3a + b + 4c - ab - 2bc + 2ac$, it can be translated with the graph in Fig. 3a; supposing to have an annealer with a topology of the shape in Fig. 3b, mapping the problem to the target graph is challenging. The solution is to create a chain between qubits 0 and 5 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, they behave as a single variable [3].
The result is in Fig. 3c.: qubits 0 and 5 compose a chain of length 2. The new coupling composing the chain is not described in the objective function, thus it must be added. This process requires a few steps: firstly, to split the bias of $+1$ from variable b, between qubits 0 and 5 obtaining a bias value for both qubits of 0.5; then it is necessary to choose the strong negative coupling strength for the chain between qubits 0 and 5. In this example is -4, stronger than those around it. To compensate for the new strength, needed is to add $-\frac{-4}{2} = 2$ to each bias of qubits 0 and 5. Now the biases for these qubits are 2.5 [3]. Table 2 summarizes the results.

Variable	Linear coefficient	Qubits	Bias
a	3	4	3
b	1	0, 5	2.5, 2.5
c	4	1	4
Edges	Quadratic coefficient	Couplers	Strength
(a, b)	-1	(0, 4)	-1
(a, c)	2	(1, 4)	2
(b, c)	-2	(1, 5)	-2
		(0, 5)	-4

Table 2: Embedding results for $H_{example}$ and the graph in Fig. 3c.

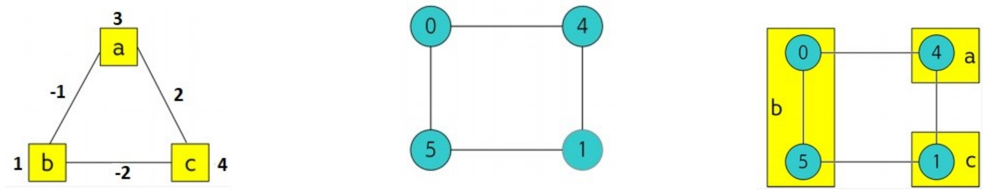

Figure 3: (a) Embedding of the Hamiltonian. (b) Example of a graph. (c) Final embed of the problem [3].

In case of more sophisticated problems, the embedding is managed by Python libraries automatically.
During the annealing, the chain could break and qubit 0 and 5, which must have the same value at the end of the process, will be of different values. This phenomenon is called chain break. This brings difficulties to the solution to the problem. This fact is solved by the majority vote method [7].

The two annealers that are used in this work have two different topologies. Since the topology is the scheme in which qubits are connected, different topologies determine different embeddings:

- DW_2000Q_6 has a Chimera topology: qubits connectivity degree is equal to 6 (each qubit is coupled to 6 different qubits through external couplers). It has 2041 qubits and 5974 couplers [4];

- Advantage_system1.1 has a Pegasus topology: qubits connectivity degree is equal to 15. It has 5436 qubits and 37440 couplers [5].

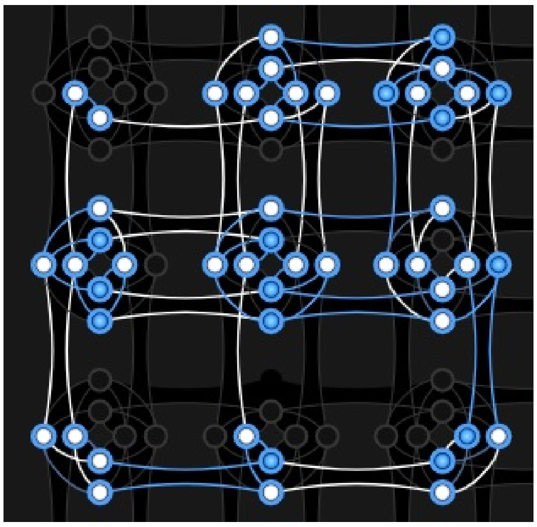
Figure 4: Example of a Chimera graph.

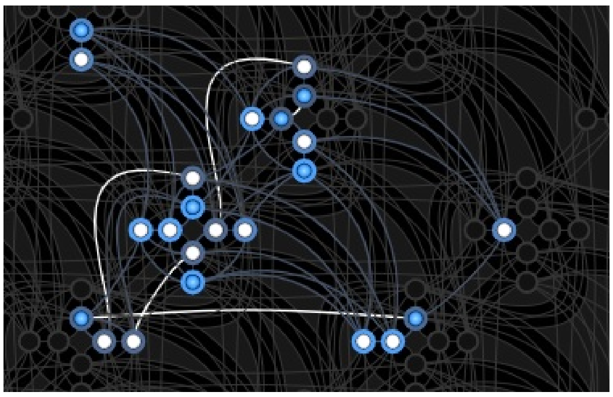
Figure 5: Example of a Pegasus graph.

In Fig. 4 and 5, it is possible to see the embedding of the same problem with 18 logical variables, in the two topologies. Blue points are the qubits, that at the end of the annealing, have the value +1 while white points have the value -1 (Ising formulation). Blue lines represent the chain necessary to embed the problem and white lines are the connection between logical variables.
Using the Chimera graph, 49 qubits are necessary to solve the problem that will be presented in Section 4.1 and in Fig. 6, while in the Pegasus graph only 23 qubits. An ideal embedding is formed from the same quantity of qubits and logical variables. An important thing is also the chain length: the longer it is, the higher the probability to have errors during the annealing. Further details are provided in Section 4.2.

4 A case study: the graph colouring problem

4.1 Materials and methods

The problem to solve is to colour a given graph, with connections between its vertices, imposing that, given a set of colours, connected vertices have different colours. It is an interesting optimization problem to study the D-Wave machines. Moreover, it is easy to extend it at lots of variables to find potential limits of the QA. In the following, the solution to the graph colouring problem computed using the DW_2000Q_6, Advantage_system1.1 and classical SA is discussed.
Starting with the simple case shown in Fig. 6,
let $x_{i,p}$ a logical variable where the index i is the number of the variable (in this case $i = 1, ..., 6$), while the index p is associated at the colour of the i^{th} variable (having 3 colours, for example, $p = 1$ represents yellow, $p = 2$ red and $p = 3$ green). When $x_{i,p} = 1$, the i^{th} variable assumes

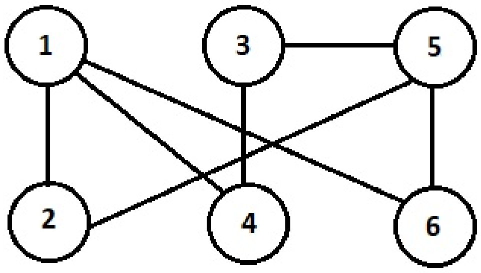

Figure 6: The case study scheme.

the p^{th} colour in the end of annealing, vice-versa, when $x_{i,p} = 0$ the variable do not assume the p^{th} colour. The two constraints to satisfy:

1. Each variable is associated to one colour:

$$\sum_{p=1}^{3} x_{i,p} = 1 \tag{11}$$

   where i=1,...,6. It is transformed in a QUBO formulation through the following penalty function:

$$P\left(\sum_{p=1}^{3} x_{i,p} - 1\right)^2, \qquad P \in \mathbb{R}. \tag{12}$$

   In case of $i = 1$, Eq. (12) becomes:

$$\begin{aligned} &P(x_{1,1} + x_{1,2} + x_{1,3} - 1)^2 = \\ &= P(x_{1,1}^2 + x_{1,2}^2 + x_{1,3}^2 + 1 - 2x_{1,1} - 2x_{1,2} - 2x_{1,3} + 2x_{1,1}x_{1,3} \\ &\qquad\qquad + 2x_{1,2}x_{1,3} + 2x_{1,1}x_{1,2}) = \\ &= P(-x_{1,1} - x_{1,2} - x_{1,3} + 2x_{1,1}x_{1,3} + 2x_{1,2}x_{1,3} + 2x_{1,1}x_{1,2}) + P. \end{aligned} \tag{13}$$

   Now the constant terms is irrelevant, thus it is neglected. By making the substitution $x_{1,1} \equiv x_1$; $x_{1,2} \equiv x_2$; $x_{1,3} \equiv x_3$; Eq. (13) becomes:

$$P(-x_1 - x_2 - x_3 + 2x_1x_3 + 2x_2x_3 + 2x_1x_2) \tag{14}$$

2. Two connected variables have different colours:

$$x_{i,p} + x_{j,p} \leq 1 \tag{15}$$

   therefore, following Table 1, the penalty function is:

$$Px_ix_j, \qquad P \in \mathbb{R} \tag{16}$$

   where i and j are two connected variables. P is a real constant and in general, different to the P introduced in the other constraint. This penalty function is introduced for all possible combinations of the same colour between variables i and j. Viewing Fig. 6, for connection between variables 1 and 2, the penalty function is:

$$P(x_1x_4 + x_2x_5 + x_3x_6). \tag{17}$$

The study of the graph colouring problem is also considered for a periodic graph of 100 logical variables coloured using 3 to 6 colours in Advantage_system1.1 and SA and the same colours for a periodic graph of 30 variables in Advantage_system1.1, DW_2000Q_6 and SA to compare the two annealers.
After 50 repetitions of QA (10000 runs each QA) for each configuration, arithmetic mean and standard deviation are calculated to evaluate and compare the quality of the solutions provided by QA and SA.
Handwriting the objective function for a problem with 100 variables is unfeasible. However, there are Python libraries such as PyQUBO that have been build to this aim [7]. More details are provided in the next section.

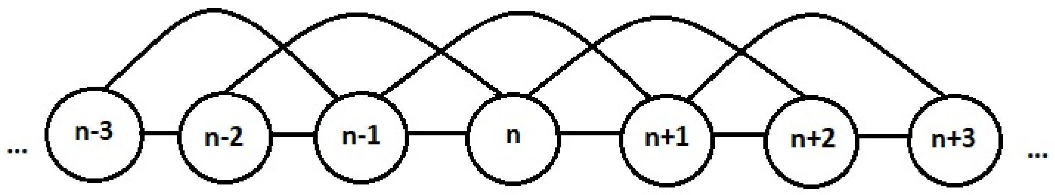

Figure 7: A representation of the graphs with many variables. The variables are periodically connected between them: the n^{th} variable is connected with the 2 following and 2 previous variables.

4.2 Results

The output from the Python script is an array containing all the configurations of the problem that the annealing finds, states of the system, and its relative energy, number of occurrences and the percentage of variable values that are based on broken chains.
Starting with the case study, there are a lot of possible configurations that satisfy the constraints of the problem. One of these is showed in Fig. 8, found by all methods used. This corresponds to have $x_1 = x_6 = x_9 = x_{11} = x_{13} = x_{18} = 1$ and the other variables equal to 0.

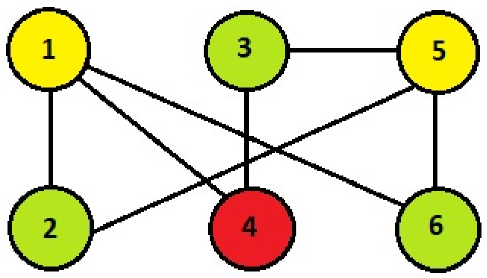

Figure 8: A solution obtained for the case study.

Fig. 9 displays the distribution of the solutions obtained with the D-Wave annealers and SA for the case study (18 logical variables). The energy of correct solutions is 0. SA finds at 99% a correct solution to the problem. Correct solutions are checked automatically with a Python

script. Instead, with D-Wave annealers percentages decrease until 71% for Advantage_system1.1 and 61% for DW_2000Q_6.
However, passing from DW_2000Q_6 to Advantage_system1.1 there is an improvement regarding the probability of finding a correct solution among all solutions calculated.
Furthermore, in general, the energy corresponding to correct solutions for other types of problems depends on the formulation of the objective function by the user, thus the energy spectrum is managed by the values that the user gives to scalars in the penalty functions.
In the end, QA seems to be less efficient in seeking correct solutions than SA.

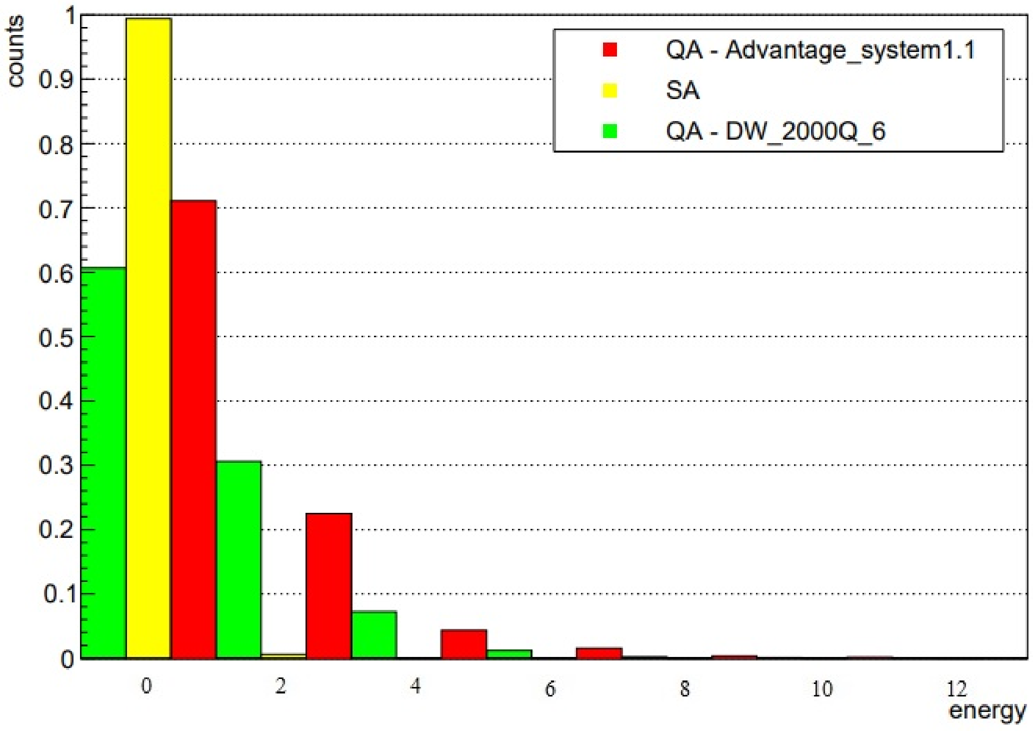

Figure 9: Normalized histogram represented the results of the case study. There are yellow bins until at *energy* = 2, while bins connected to QA arrive at *energy* = 12.

Table 3 shows results about the graph with 100 variables, connected as in Fig. 7, coloured by 3,4,5 or 6 colours. Sampling times and correct solutions found are listed. All results are based on runs of 10000 annealing cycles using the Advantage_system1.1 and SA, the maximum possible. To embed the problem with 300, 400, 500 and 600 variables were needed, respectively, $769.3 \pm 10.6, 1206.3 \pm 14.3, 2271.9 \pm 37.3$ and 3204 ± 160 qubits (numbers are means and standard deviations calculated on 50 repetitions for each case).
In the table, the number of variables are 300, 400, 500 and 600 and not 100 because, for example, if one variable can assume 5 or 6 different colours, the number of logical variables in the Python script used to solve the problem is 5 or 6.
The table underlines what was discussed before about the efficiency of QA and SA: the SA find correct solutions also when QA does not give any; if with few variables the difference is notable,

enhancing the number of variables, QA seems to fail. Probably, the number of qubits used to embed the problem is too high that errors due to environment or due to broken chains during the annealing, bring QA to not find correct solutions. These problems are absent if the graph colouring is solved by SA.

Always in Table 3, it is possible to study the sampling time and how it changes for an increasing number of qubits. In both methods, it expands when the number of variables increases. Sampling time includes the time to execute one run and the readout time where the results are elaborated to be sent to the user.
In the end, SA seems to be less fast than QA, however, the sampling time between the two methods is relative if a correct solution is not found. This last fact seems to happen in QA in graph colouring problems with 300, 400, 500 and 600 variables.

Variables	Sol. - SA	Sol. - Adv.	Sampl. time - SA (s)	Sampl. time - Adv. (s)
300	0	0	101,35	1,50
400	0	0	157,78	1,71
500	5220	0	237,84	1,72
600	9440	0	281,64	1,74

Table 3: Correct solutions and sampling time for SA and QA using Advantage_system1.1 for a problem with 300, 400, 500 and 600 logical variables (the numbers in the table are not means).

Fig. 10 describes how QA sampling time behaves increasing the qubits used to embed the problem.
The graphs with 30 variables are here useful to compare the two machines. Sampling times tends to be constant, but there are some fluctuations, especially in Advantage_system1.1.
To note is the different velocities of the two annealers: this shows the improvements applied in Advantage_system1.1, even if the fluctuations are increased.

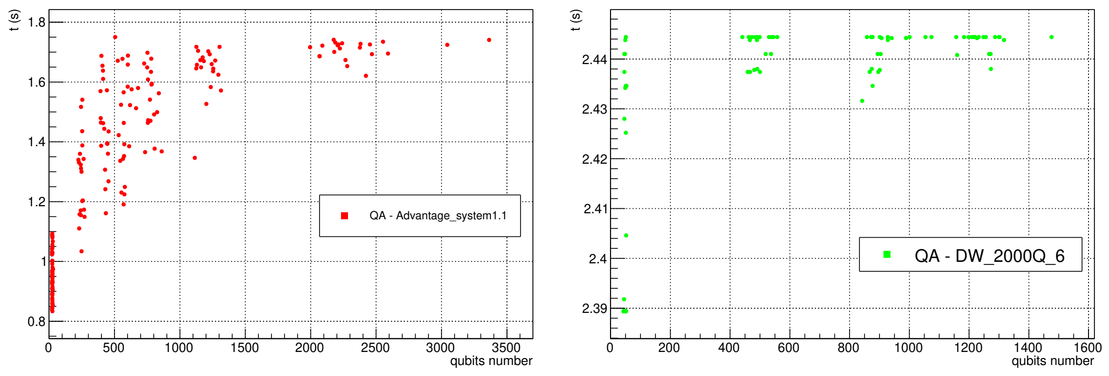

Figure 10: A representation of time versus number of qubits for both annealers.

Fig. 11 and Fig. 12 show how the number of qubits and the maximum chain length depending on the number of qubits.
From images turn out that a double number of qubits are needed to implement the same problem with the same quantity of logical variables on the DW_2000Q_6 with respect to

Advantage_system1.1. In the same conditions also the maximum chain length is double in DW_2000Q_6.
The maximum chain length is the maximum number of qubits in a chain. If it is higher than the degree of connectivity, it means that the embedding is too complex, thus could occur errors of chain break during the annealing frequently. The chain length is an index to evaluate the degree of connectivity.
The increase of the number of qubits with the enhance of logical variables and, at the same way, for the maximum chain length, is due to the two different topologies, Pegasus in Advantage_system1.1 and Chimera in DW_2000Q_6: Pegasus topology having more connectivity degree than Chimera topology allows implementing the problem easily, fewer errors of chain break during QA and fewer number of qubits used.
Sometimes, if the chain length is greater than the connectivity degree it is necessary to introduce auxiliary variables, thus new qubits are used and there are more possibilities of error.
This is a hint that an improvement in the connectivity might be more important than having a great number of qubits. In the end, graphs demonstrate improvements in these two features, however, they are not yet sufficient to compare QA and SA about their efficiency.

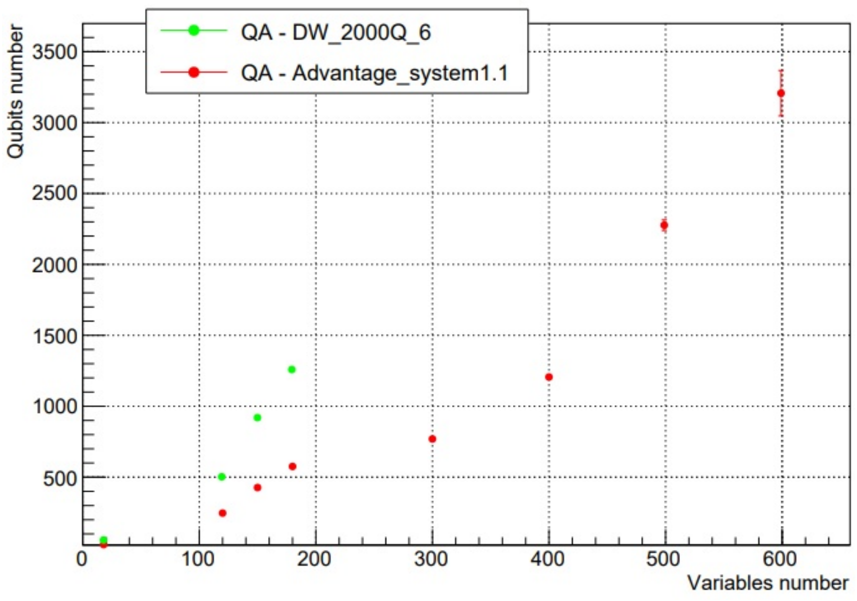

Figure 11: The evolution of number of qubits versus the number of logical variables.

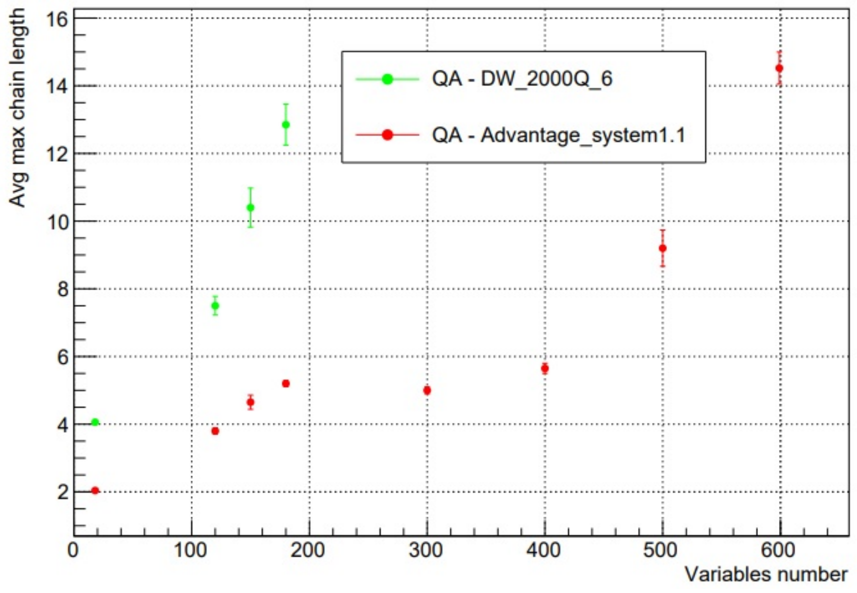

Figure 12: The evolution of maximum chain length versus the number of logical variables.

5 Conclusion

The goal of this thesis is to go through the whole process needed to solve a problem on a D-Wave annealer, from its mathematical formulation to the final implementation.
In the first chapter, QA was introduced and the concepts of qubits, couplers and QA Hamiltonian were described.
The second chapter introduced the QUBO and Ising formulation and the concepts of minor embedding.
In the last chapter, an application to the graph colouring problem was presented and was studied. The study turned out that passing from DW_2000Q_6 to Advantage_system1.1 there was an improvement regarding the percentage of correct solutions found but not yet at the level achieved by SA. Indeed, in the problems with 300, 400, 500 and 600 variables, annealers seem to fail in seeking the correct solution.
A possible solution, that comes to light after this thesis, is to improve the topology of the annealer. It is crucial to reduce the errors, especially between chains during the QA: more is the degree of connectivity, less is the probability to have errors of a broken chain.
In the end, the next step will be to explore problems with real variables that can assume values different to 0 and ±1, which are difficult to solve with SA.

6 Bibliography

1. T. Kadowaki and H. Nishimori, *Quantum Annealing in the Transverse Ising Model*, 1998.
2. M. Born and V. A. Fock, *Beweis des Adiabatensatzes*, 1928.
3. D-Wave Systems Inc., *Getting Started with the D-Wave System*, 2021.
4. D-Wave Systems Inc., *QPU-Specific Physical Properties: DW_2000Q_6*, 2020.

5. D-Wave Systems Inc., *QPU-Specific Physical Properties: Advantage_system1.1*, 2020.

6. F. Glover, G. Kochenberger and Y. Du, *A Tutorial on Formulating and Using QUBO Models*, 2019.

7. D-Wave Systems Inc., D-Wave Ocean Software Documentation: *https://docs.ocean.dwavesys.com/en/stable/*, 2021.

8. A. Cho, *Quantum or not, controversial computer runs no faster than a normal one*, Science, 2014.

9. T. Albash, T. F. Ronnow, M. Troyer, and D. A. Lidar, *Reexamining classical and quantum models for the D-Wave One processor*, 2014