University of Padova

Department of Mathematics

Master Thesis in Data Science

Quantum Integer Programming: an Annealing approach to Job Shop Scheduling problem

Supervisor
Francesco Rinaldi
University of Padova

Master Candidate
Andrea Di Trani

Academic Year
2022-2023

To everyone who pushed me doing better and being a better person

Abstract

This thesis develops a complete workflow to get optimal solutions to the JSS problem. In this work, many of the steps required in the usage of quantum annealers are deeply analyzed. In detail, two different approaches to the minor-embedding procedure are considered and compared to the current default heuristic implemented by D'Wave. Moreover, in order to expand the scope of QA, a hybrid algorithm that extracts the Graver basis of the problem to augment an initial set of feasible solutions to obtain optimal ones. The obtained results show that the quantum annealers can get to optimal solutions in a competitive time but, due to the limited number of working qubits and the sparse connectivity among them, only small instances can be efficiently solved with the current hardware.

Contents

Listing of figures

Listing of tables

Listing of acronyms

JSS . . . . . Job Shop Scheduling Problem

QUBO . . . . . Quadratic Unconstrained Binary Optimization problem

IP . . . . . Integer Program

ILP . . . . . Integer Linear Program

QA . . . . . Quantum Annealing algorithm

SA . . . . . Simulated Annealing algorithm

1 Introduction

The thesis focuses on the implementation of quantum annealing algorithms for solving optimization problems. The first step will be the introduction of the theoretical basis regarding QUBO problems and the Ising model being the fundamental concepts upon which the whole framework stems. It explores the use of different graph structures, such as the Chimera graph and the Pegasus graph, as well as techniques like minor-embedding and chain strength to improve the efficiency of the algorithms. The thesis also discusses the energy spectrum of the solutions obtained through simulated annealing and highlights the benefits of using the higher-density Pegasus graph. Additionally, it mentions the use of template-based minor embedding and a graph-theoretic framework for optimizing adiabatic quantum program compilation. During this work, several algorithms and research papers in the field of adiabatic quantum optimization will be used and referenced. Overall, the thesis provides insights into the implementation and optimization of quantum annealing algorithms for solving optimization problems.

2 QUBO problem

This chapter will set part of the theoretical landscape of the work done, it will be organized in 3 sections. Section 2.1 defines what a QUBO problem is, how to reformulate a constrained program to fit the definition and some of the applications. Second Section 2.2 is going to analyze some of the exact or heuristic/metaheuristic approach to solution currently used to tackle this kind of problems. In the third and last Section 2.3 are presented some numerical instances of QUBO problems.

2.1 Definition and Applications

Given a set of binary variables $\mathbf{x} \in \{0,1\}^n$ and a matrix $Q \in \mathrm{M}^{n\times n}$, a QUBO problem takes the following form:

$$\min/max_{\mathbf{x}\in\{0,1\}^n} \mathbf{x}^\intercal Q\mathbf{x} \tag{2.1}$$

By selecting an appropriate matrix Q it is possible to formulate a wide variety of problems in this form, some of them are:

- Quadratic Assignment Problems
- Multiple Knapsack
- SAT problems

- Max-Cut problem
- Number Partitioning problem

2.1.1 Natural QUBO formulation

Some optimization problems have a natural QUBO formulation, in the following subsections some examples are going to be presented. More formulations like these can be found in [1].

Number Partitioning problem

Let $S = \{a_1, a_2, ..., a_m\}$ be a set of numbers, then define

$$sum_1 = \sum_j^m s_j x_j$$

$$sum_2 = \sum_j^m s_j - sum_1$$

Where for each s_j the variable x_j takes value 1 if the number is in the first group. The Number Partitioning problem aims to minimize the difference:

$$diff^2 = (c - 2\sum_j^m s_j x_j)^2 = c^2 + 4\mathbf{x}^\intercal Q \mathbf{x} \tag{2.2}$$

and by dropping the multiplicative and additive constant the QUBO form arises.

Max-Cut problem

Given an undirected graph (V, E), the Max-Cut problem aim to partition V into 2 sets such that the edges between them is as large as possible, this can be formalized as:

$$max \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) \tag{2.3}$$

where for each $j \in V$ $x_j = 1$ if the node belongs to the first set, 0 otherwise.

2.1.2 QUBO reformulation of constrained binary problems

The usefulness and the applicability of the QUBO problem can be further extended by reformulating several constrained optimization problems. This can be done by translating each constraint in a corresponding penalty term on the objective function, here are listed some of the most common constraints and the relative penalty function.

Constraint	**Penalty term**
$\sum_i x_i \leq 1$	$\sum_{i \neq j} x_i x_j$
$\sum_i x_i \geq 1$	$1 - \sum_i x_i + \sum_{i \neq j} x_i x_j$
$\sum_i x_i \leq 1$	$1 - \sum_i x_i + 2 \sum_{i \neq j} x_i x_j$

Table 2.1: Table of conversion from constraint to QUBO penalty term

Similar transformations are going to be used in the conversion of the JSS problem to a QUBO form while a more general way of conversion, able to tackle any ILP will be presented later.

2.2 Classical Solution Methods

During the last 30 years, several methods got developed to obtain solutions of QUBO problems. Those algorithms can be essentially classified in 2 groups, exact and heuristic/metaheuristic methods.

2.2.1 Exact Methods

Most of the current exact methods are based on branch-and-bound or branch-and-cut algorithms.

The following algorithms assure, given enough time and memory, the global optimal solution, they have been tested successfully for a few hundreds of variables as reported in [1]

Branch-and-Bound

Proposed in [2] in 1960, as suggested by the name, this method is divided in 2 different operations. First there is the branch, the operation of generating the feasible solutions, it can be formalized as:

> Given an optimization problem $z = \min/\max_{\mathbf{x} \in X} f(x)$ and a partition of the feasible region X in $X_1, X_2, ..., X_n : \bigcup_i X_i = X$, let $z^{(k)} = \min/\max_{\mathbf{x} \in X_k} f(x)$. Then the solution of the problem is $\min/\max\ z^{(k)}$

Since, in general, generating the full tree has an exponential cost, the algorithm greatly benefits from and efficient way of exploring the tree. This is done by bounding the value of the objective function for each solution represented by a node.

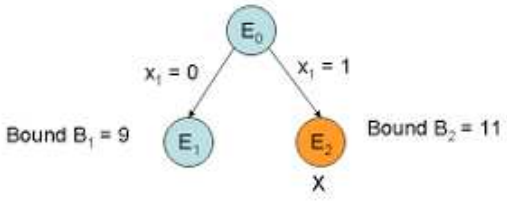


Figure 2.1: Bound usage.

In the above example, the node E_0, corresponding to the whole feasible region, is split in 2 by assigning a value to the variable x_i, supposing that a bound can be computed on both E_1 and E_2 it is possible to completely discard the exploration of the E_2 branch of the tree.

Branch-and-Cut

An evolution of the Branch-and-Bound algorithm, it iteratively refines the search region by solving a continuous LP relaxation of the problem at hand before branching. During the execution it is needed to store every cut obtained from the separation phase. One of the earliest application of this algorithm can be found in [3], where it was used to solve the traveling salesman problem.

2.2.2 Heuristics and Metaheuristics

Since the QUBO problem is NP-hard (with the exception of few special cases), as proved in [4], a large amount of heuristic/metaheuristic approaches has been developed to obtain near-optimal solutions. In the following are depicted some of them. There are also stochastic approaches for solving QUBO problems.

Tabu search

Formalized by Glover in [5] is one of the most famous methods of obtaining sub-optimal solution of integer and binary programs. This algorithm has drawn a lot of attention from the scientific community and several improvements has been proposed, starting by Glover himself in [6]. The main strategy of this approach is to start from a set of sub-optimal solution and then improve them by iteratively explore their neighbors.

Simulated annealing

Introduced in 1983 by Kirkpatrick et al. in [7], revolves around random sampling solutions of the current problem by reducing at each iteration the probability of accepting a worse solution by gradually reducing the "temperature" of the system.

3
Ising Model

This chapter will define and analyze the Ising Model, a simple physical model that describes a set of $\frac{1}{2}$-spins in a d-dimensional lattice in the presence of an external magnetic field. This model has been largely studied by physicists since 1920, the year of its invention, due to its simplicity and theoretical content. The chapter is organized into 3 sections: in the first one 3.1 the definition and some basic properties will be discussed, then in 3.2 some numerical analysis will be performed, showing some interesting behavior of the system, and in the end at 3.3 the connection between QUBO problems and Ising models will be made clear.

3.1 Definition and Properties

Given a set of $\frac{1}{2}$-spins distributed in a lattice like in figure the figure below, the Hamiltonian of the Ising model is:

$$E = -J \sum_{<ij>} s_i s_j - \mu H \sum_i s_i \tag{3.1}$$

where $s_i = +1$ if the i-th spin is 'up' and -1 otherwise, μ is the magnetic moment (which depends on the nature of the spins, may them be electrons, currents in a superconductor ecc.), H is the external magnetic fields, J is the interaction energy and $< ij >$ indicate a summation among nearest neighbor.

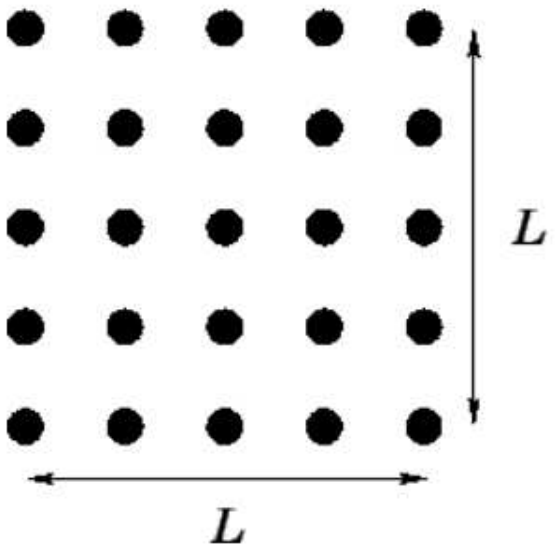


Figure 3.1: Example of a 2d lattice.

The assumption that the only interactions between spins is among nearest neighbor is quite realistic since on average the exchange effect has a very short range.

Studying the Ising model is Useful for several reasons:

- It presents a phase transition at some critical temperature.
- Has a symmetry breaking behavior at low temperatures
- Is exactly solvable for low dimensional lattices

3.2 Numerical Solution and Phase Transition

While Ernst Ising found an exact solution for the 1-dimensional case in his doctoral dissertation in 1925, such low dimensionality fails to show some of the most interesting features of the model. In fact, the 1-dimensional case does not present any phase transition.

To study higher dimensional Ising models it is useful to approach the system using an approximation called *mean field approximation*. The energy of a single atom is

$$e_i = -\mu H_{eff} s_i \tag{3.2}$$

Where $H_{eff} = H + \frac{J}{2\mu}\sum_k s_k$ is the effective magnetic field. Consider a single atom in a magnetic field H_{eff} at temperature T, according to the Boltzmann distribution, the mean spin is

$$\overline{s} = tanh(\beta\mu H_{eff}) \tag{3.3}$$

where $\beta = \frac{1}{kT}$.

Now, applying the mean field approximation, suppose that every component of the system has identical spin $s_i = \overline{s}$ so it is possible to write

$$H_{eff} = H + \frac{zJ\overline{s}}{2\mu} \tag{3.4}$$

The critical temperature is defined as

$$T_c = \frac{zJ}{2k} \tag{3.5}$$

From the following plots it is easy to understand that when the temperature gets to the critical value the system goes through a phase transition

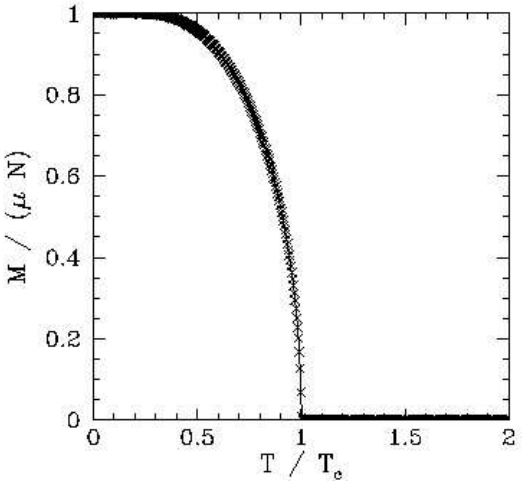


Figure 3.2: Net magnetization in absence of external magnetic field.

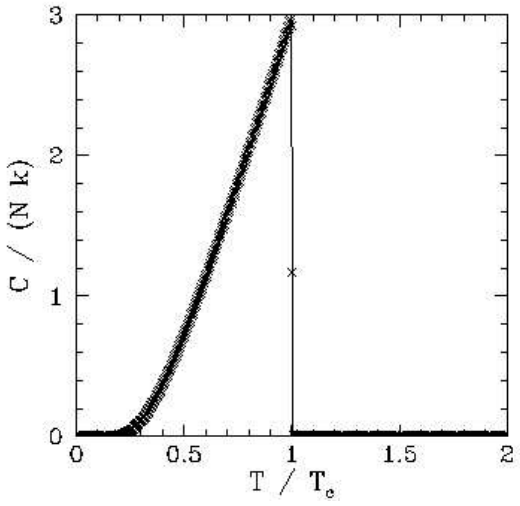


Figure 3.3: Heat capacity in absence of external magnetic field.

3.3 Ising formulation of a QUBO problem

The interest in the Ising model in the context of this thesis is the similarity it has with a QUBO problem. In fact, with a simple transformation it is possible to switch between the two.

Consider a QUBO problem

$$min\ \mathbf{x}^\intercal Q\mathbf{x} \tag{3.6}$$

then, using the relation $x_i = \frac{1+s_i}{2}$ it is possible to obtain an Ising Hamiltonian where the ground state represents the optimal solution of the QUBO problem by replacing the spin state -1 with the variable value 0. The full procedure, with examples and more details, can be found in [8].

4

Quantum Annealing algorithm

The name Quantum Annealing algorithm (QA) refers to a class of algorithms that differs from one another in some implementation choices that are left to the programmer. The main idea behind QA is to encode the objective function of an optimization problem in an Ising Hamiltonian, then, exploiting the quantum adiabatic theorem, let the system evolve to the ground state of such system obtaining the optimal solution of the original problem.

This is done in the hope that the capability of quantum system to perform quantum jumps (tunneling) will benefit the exploration of the search space by bypassing barriers of potential.

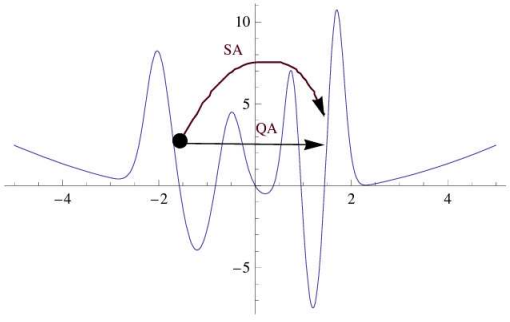


Figure 4.1: Quantum jumps compared thermal jumps performed by SA, figure from [9].

4.1 Definition and Theoretical Analysis

Given a cost function E, define the associated Hamiltonian H_f as explained in 3.3, with the corresponding operator $\hat{H}_F$ that when applied to a state $\sigma \in \{-1, +1\}^n$ behaves as follows:

$$\hat{H}_f |\sigma\rangle = E(\sigma) |\sigma\rangle \tag{4.1}$$

Consider then an arbitrary Hamiltonian H_i with a known ground state $\mathbf{g}_i \in \{-1, +1\}^n$, this will be the initial state of the system during the execution of QA.

Let $f(t)$ be a decreasing function such that it goes from a large value $f(0)$ to $f(t) = 0$ for $t \to \mathcal{T}$. This function will control the rate of interpolation between the initial Hamiltonian H_i and the target H_f. The final time-dependent Hamiltonian is defined as:

$$H(t) = H_{Ising} - \Lambda(t) H_{kin} \tag{4.2}$$

The state $|\psi(t)\rangle$ of the system will then evolve according to the Schrodinger equation

$$i \frac{d}{dt} |\psi(t)\rangle = \hat{H}(t) |\psi(t)\rangle \tag{4.3}$$

While this definition leaves room for different implementations, like the initial Hamiltonian H_{kin}, the time computation time $\mathcal{T}$ and the interpolation function, there are some requirements that have to be met in order to have a guaranteed convergence.

4.1.1 Adiabatic theorem and Convergence requirements

Consider a general Hamiltonian $H(s)$ which depends on time only by a dimensionless variable $s = t/\mathcal{T}$. Let $|k(s)\rangle, \epsilon_K(s)$ be, respectively, the k-th eigenstate and the k-th eigenvalue of $H(s)$ at time s

$$H(s) |k(s)\rangle = \epsilon_k(s) |k(s)\rangle \tag{4.4}$$

then, from differentiation of 4.4

$$\langle j(s)| \frac{d}{ds} |k(s)\rangle = \frac{-1}{\epsilon_j(s) - \epsilon_k(s)} \langle j(s)| \frac{dH(s)}{ds} |k(s)\rangle \tag{4.5}$$

It is possible now to prove the most important theoretical component of the QA formulation.

Theorem 1 *If the instantaneous ground state of the Hamiltonian $H(s)$ is not degenerate for $s \geq 0$ and the initial state is the ground state at $s = 0$, the state vector $|\psi(s)\rangle$ has the asymptotic form in the limit of large $\mathcal{T}$ as*

$$|\psi(s)\rangle = \sum_j c_j(s) e^{-i\mathcal{T}\phi_0(s)} |j(s)\rangle \tag{4.6}$$

$$c_0(s) \approx 1 + O(\mathcal{T}^{-2}) \tag{4.7}$$

$$c_{j\neq 0} \approx \frac{i}{\mathcal{T}} \left(A_j(0) - e^{i\mathcal{T}(\phi_j(s)-\phi_0(s)} A_j(s) \right) + O(\mathcal{T}^{-2}) \tag{4.8}$$

where $\phi_j(s) = \int_0^s ds' \epsilon_j(s')$, $\Delta_j(s) = \epsilon_j(s) - \epsilon_0(s)$ and

$$A_j(s) = \frac{1}{\Delta_j(s)^2} \langle j(s)| \frac{dH(s)}{ds} |k(s)\rangle \tag{4.9}$$

The proof of this theorem is given in [10]

To perform an adiabatic evolution it is needed that the excitation probability in 4.8 to be as small as possible, this is achieved for

$$\mathcal{T} \gg |A_j(s)| \tag{4.10}$$

Using the original time variable t the adiabatic condition can be written as:

$$\frac{1}{\Delta_j(t)^2} \left| \langle j(t)| \frac{dH(t)}{dt} |0(t)\rangle \right| = \delta \ll 1 \tag{4.11}$$

If the Hamiltonian H_{kin} represents an Ising model with only transverse field interaction

$$H_{kin} = \sum_j \sigma_j^x \tag{4.12}$$

It is possible to express the function $f(t)$ in the following form

$$\Lambda(t) = a(\delta t + c)^{-1/(2N-1)} \tag{4.13}$$

While if the initial Hamiltonian is composed of ferromagnetic interactions between spins as

well

$$H_{kin} = \sum_j \sigma_j^x + \sum_{ij} \sigma_i^x \sigma_i^x \tag{4.14}$$

the dependence of time takes the form of

$$\Lambda(t) \propto t^{-1/(N-1)} \tag{4.15}$$

4.2 Computational Complexity

The computational time needed to QA in order to reach a sufficiently low error rate ϵ are estimated in [10] from 4.13 and 4.15:

- $\mathcal{T} = \frac{1}{\delta}\left(\frac{1}{\epsilon}\right)^{2N-1}$ for the transverse field Hamiltonian 4.12
- $\mathcal{T} = \frac{2^{N-2}}{\delta\epsilon}$ for the ferromagnetic Hamiltonian 4.14

From the above results is clear that the presence of non-zero non-diagonal elements in the initial Hamiltonian leads to faster, but still exponential computational complexity. However, as will be exemplified in the next section, the choice of an optimal annealing schedule may allow for an even faster rate of convergence.

4.2.1 Better Annealing Schedules

As proved in [11], the general bound to excitation probability is in the order of $\mathcal{T}^{-2}$. However in [10], is demonstrated that this bound can be improved to $\mathbf{T}^{-2m}$ at the cost of a small numerical factor on computational complexity.

Given the asymptotic form of the excitation amplitudes 4.8, the upper bound of excitation probabilities is:

$$|\langle j(1)|\psi(1)\rangle|^2 = |c_{j\neq 0}(1)|^2 \tag{4.16}$$

If the $\mathcal{T}^{-2}$ vanishes, i.e. when $H'(0) = 0$ and $H'(1) = 0$, then the excitation amplitudes are at most of order $\mathcal{T}^{-2}$

$$c_{j\neq 0} \approx \frac{1}{\mathcal{T}}\left(A_j^{(2)}(0) - e^{i\mathcal{T}|\phi_j(s)-\phi_0(s)|}A_j^{(2)}(1)\right) + O(\mathcal{T}^{-3} \tag{4.17}$$

where

$$A_J^{(m)}(s) = \frac{1}{\Delta_j(s)^{m+1}}\langle j(s)|\frac{d^m H(s)}{ds^m}|0(s)\rangle \tag{4.18}$$

The generalization of this argument leads to the result expressed with the next theorem

Theorem 2 *if the k-th derivative of $H(s)$ i zero at $s = 0$ and $s = 1$ for all $k = 1, ..., m - 1$, the excitation probability has the upper bound*

$$| \langle j(1)| |\psi(1)\rangle |^2 \lesssim \frac{1}{\mathcal{T}^{2m}}(|A_j^{(m)}(0) + |A_j^{(m)}(1)|)^2 + O(\mathcal{T}^{-2m-1}) \tag{4.19}$$

Although the general time-dependent Hamiltonian was investigated until now, the standard form of the Hamiltonian of QA is

$$H(s) = f(s)H_{pot} + (1 - f(s))H_{kin} \tag{4.20}$$

where H_{pot} is the Ising Hamiltonian of the problem and H_{kin} is the Hamiltonian with a trivial ground state from which the annealing process starts. From theorem 2 and equation 4.20, it is easy to understand that the requirements on the derivative of $H(s)$ translate in the same fashion to the derivatives of $f(s)$.

Some examples of annealing schedule $f_m(s)$ with $\mathcal{T}^{-2m}$ error rate are reported in [10]:

- $f_1(s) = s$
- $f_2(s) = s^2(3 - 2s)$
- $f_3(s) = s^3(10 - 15s + 6s^2)$
- $f_4(s) = s^4(35 - 84s + 70s^2 - 20s^3)$

4.3 Example: Grover algorithm

The process of optimizing QA is well demonstrated for the problem of searching in an unordered dataset, where the same quantum speedup reached by gate-based algorithms can be achieved also by QA by selecting the right annealing schedule. Consider a database of N unordered items, one among them is marked. The problem consists of finding the marked item in the shortest amount of time possible. Any classical algorithm can perform this task with an average of $N/2$ query to the database, in contrast, the gate-based quantum algorithm developed by Grover in [12] can find the solution with $O(\sqrt{N})$ complexity. Sometime later, in [13], an equivalent QA was found.

The Hilbert space of this problem is described by the basis states $|i\rangle$ with $i \in 1, ..., N$ and the state $|m\rangle$ represents the marked item. The Hamiltonian is then constructed as:

$$H_{pot} = 1 - |m\rangle \langle m| \tag{4.21}$$

$$H_{kin} = 1 - \frac{1}{N} \sum_i \sum_j |i\rangle \langle j| \tag{4.22}$$

The initial state (ground state of H_{kin}) is

$$|\psi(0)\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle \tag{4.23}$$

and the energy gap is

$$\Delta(s) = \sqrt{1 - 4\frac{N-1}{N} f(s)(1 - f(s))} \tag{4.24}$$

that has a minimum at $f(s) = 1/2$ thus, following the adiabaticity condition 4.10 the value of δ is

$$\frac{\sqrt{N-1}}{\mathcal{T} N \Delta_1(s)^3} \frac{df}{ds} = \delta \tag{4.25}$$

And integrating 4.25 the result is the optimal annealing schedule $f_{opt}(s)$:

$$f_{opt}(s) = \frac{1}{2} + \frac{2s - 1}{2\sqrt{N - (N-1)(2s-1)^2}} \tag{4.26}$$

giving an annealing time of

$$\mathcal{T} = \frac{\sqrt{N-1}}{\delta} \tag{4.27}$$

The optimal annealing schedule shows the $\mathcal{T}^{-2}$ error rate because it is derivative is non-vanishing at 0 and 1, as demonstrated in [10], this can be circumvented, bringing the error rate at $\mathcal{T}^{-2m}$ by re-scaling the time s, this lead to better results on longer computational time.

This is a prime example to show that, while an exponential speedup using quantum hardware may not be always possible, the right selection of the initial Hamiltonian H_{kin} and the annealing schedule $f(s)$ can lead to faster annealing time.

4.4 Workflow and Implementation

Even though the theory of QA may seem enough for the implementation of the algorithm that's not the case. Several steps are required to efficiently run QA, most of them are hardware-related, this is why this section will open with a survey on currently available quantum annealers.

The first thing to point out in regards to current quantum annealers is that the qubits (spins) are not densely coupled, this means that in both Hamiltonian's H_{kin} and H_{pot} the quadratic terms that are possible to construct are the one relative to physically connected qubits. The pattern that describes the qubits coupling is called the hardware graph. In the future the hope is to develop hardware with bigger and denser graphs in order to accommodate bigger instances of optimization problems.

4.4.1 Chimera graph

Used on the D'Wave 2000Q the so-called Chimera graph is of the previous generation of D'Wave's QPUs. It is simplicity is helpful in understanding the basic concepts and to study newer topologies, since they will be derived from this one.

A schematic description of the structure of the graph is presented in [14] and it goes as follows: consider a $M \times M$ grid of $K_{4,4}$ bipartite graphs called cells, each node of those cells is going to be referred to as $n_{(x,y),i,p}$ where (x, y) are the coordinate of the cell, $i \in \{0, 1\}$ represent the 'side' where the node lies, and $p \in \{0, 1, 2, 3\}$ is the position of the node (0 being the top node, 3 being the bottom one). Each cell is connected to the neighboring ones by edges of the form:

$$(n_{(x,y),i,p}, n_{(x+k_1,y+k_2),i,p}) s.t. k_1, k_2 \in \{0, 1\} and k_1 + k_2 = 1 \tag{4.28}$$

To better understand how the couplers are distributed, it is introduced the qubits orientation and relative positioning

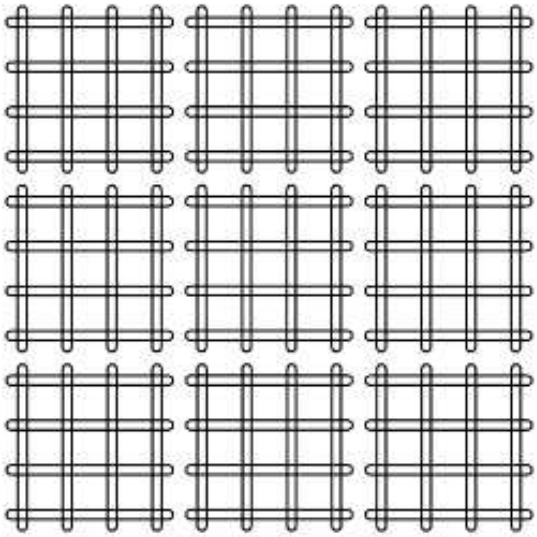


Figure 4.2: Chimera qubits orientation, each qubit is visualized as a vertical or horizontal loop, figure from [14].

To connect those qubits, 2 sets of couplers are put in place:

- **Internal couplers**
  Internal couplers connect pairs of orthogonal qubits

- **External couplers**
  External coupler connect collinear pairs of qubits

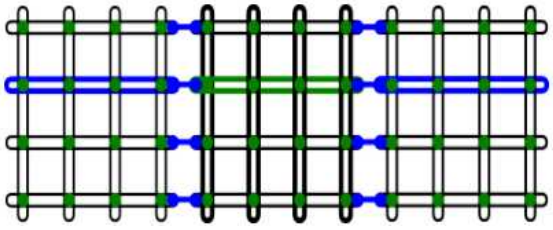


Figure 4.3: Chimera couplers, internal couplers in green, external in blue, figure from [14].

To reveal more about the structure of the Chimera graph there is another useful visualization

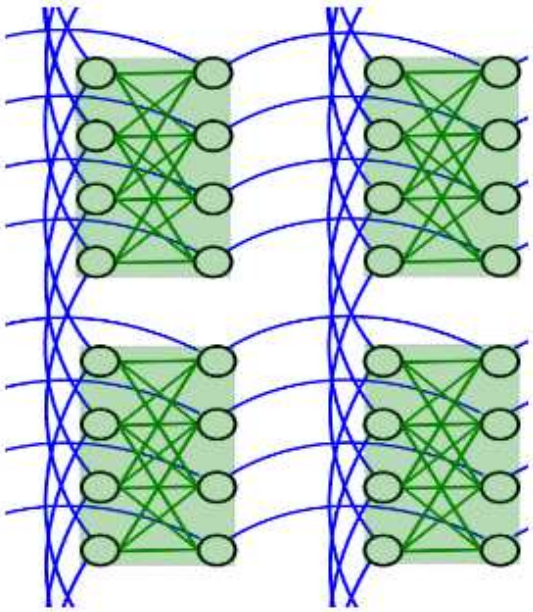


Figure 4.4: Chimera couplers, internal couplers in green, external in blue, each node of the graph is represent a qubit, figure from [14].

From the above figure is clear that a Chimera graph is composed of a 2-d square lattice of $K_{4,4}$ bipartite graphs where each edge represent an internal coupler, connected from external couplers.

4.4.2 Pegasus graph

The current generation hardware graph is called Pegasus and it is realized on the Advantage QPUs. As seen before, a Chimera graph is essentially composed of an $X \times Y$ grid of $K_{4,4}$ graphs. A Pegasus graph is composed of $Z = 3$ layers of Chimera, so it is possible to point out a single $K_{4,4}$ cell using 3 indices. Moreover, each 'side' of the $K_{4,4}$ cells is identified by a binary index i and the nodes on each side correspond to 2 more binary indices j, k. In this notation, each node is identified by the unique tuple (x, y, z, i, j, k).

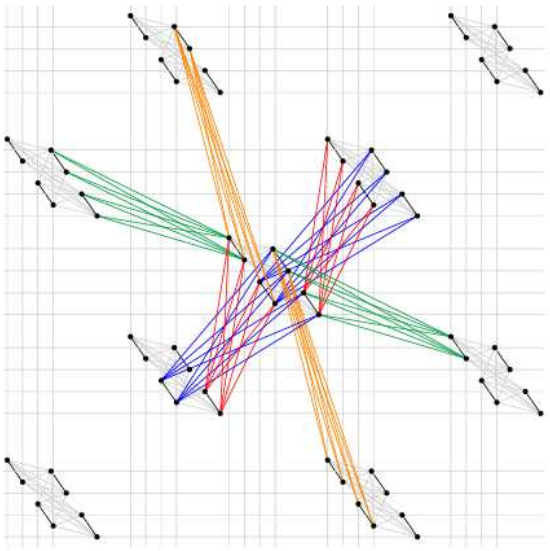


Figure 4.5: Pegasus node visualization, the couplers not part of Chimera are highlighted, figure from [15].

With this topology 2 new sets of couplers are added to the one present in Chimera:

1. **New edges in $K_{4,4}$ cells**
   to each cell are added the edges described as:

$$(x, y, z, , i, j, 0) \longleftrightarrow (x, y, z, i, j, 1) \tag{4.29}$$

2. **Edges connecting Chimera layers**
   the connections between different Chimera layers follow the rule:

$$(x, y, z, i, j, k) \longleftarrow (x - j\bar{i} + \delta_{z2}, y - ji + \delta_{z2}, (z + 1)mod3, \bar{i}, j', k') \tag{4.30}$$

The new couplers can be grouped in a set called *Odd couplers* better described in the orientation based visualization.

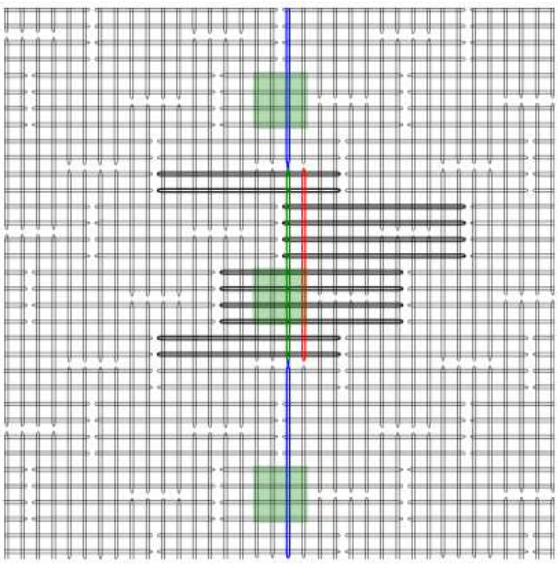


Figure 4.6: Pegasus odd couplers visualization, the green vertical qubit is coupled with the red one by an odd coupler, figure from [15].

4.4.3 Minor-embedding and chain strength

To overcome the limitations in the structure of the problem graph, an intermediate step has to be taken before the annealing phase. The main idea behind minor-embedding is, instead of using one qubit for every logical variable, to map each problem variable to a sub-tree of the hardware graph by forcing each qubit of the sub-tree in the same state; this permits a higher number of connections.

In a more formal way, a minor-embedding of a graph Y in X is defined as a map

$$\phi : V(Y) \mapsto V(X) \tag{4.31}$$

such that

$$\forall (y_1, y_2) \in E(Y), \exists (x_1, x_2) \in E(X) \; s.t. \; x_1 \in \phi(y_1), x_2 \in \phi(y_2) \tag{4.32}$$

The subset $\phi(y)$ of $V(X)$ will be referred as vertex model of y. A trivial example of this process is the following:

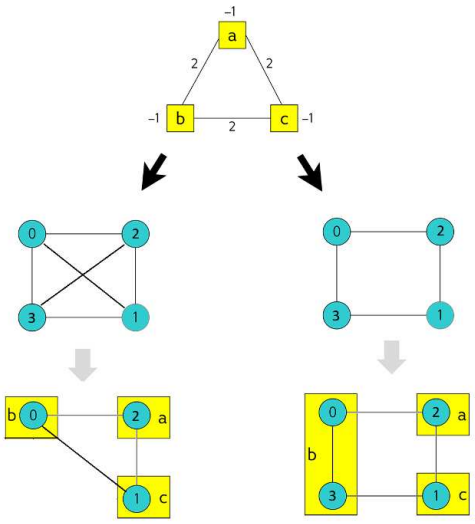


Figure 4.7: Minor-embedding example, on the left the hardware graph is fully connected while on the left is a sparse, figure from [16].

The last thing that has to be specified in this procedure is the mechanism that forces the chained qubits to stay in the same state. This is done by setting the strength of the mutual interaction among those qubits to a big negative value. The optimal value to assign at those interactions is an open research area, the chain strength has to be high enough to put any state in which a chain is broken high enough in the energy spectrum to make the rupture unlikely but can't be arbitrarily high due to the limited resolution of current hardware.

In the D'Wave framework, the default option to perform this task is performed through the uniform torque compensation, which establishes the chain strength in the following way: Given a BQM (QUBOs and BQMs can be converted into one another), let J_{ik} be the quadratic terms the chain strength is computed as

$$\gamma \sqrt{\frac{\sum_{ik} J_{ik}^2}{|J|}} \sqrt{\overline{d}} \tag{4.33}$$

where $\overline{d}$ is the average degree of the variables.

In recent years multiple methods of finding ϕ got developed, three different algorithms will be described in the next chapter.

5

Minor-embedding algorithms

As mentioned above, minor-embedding takes a crucial part in the implementation of QA, this chapter will delve in three different approaches:

- **Minorminer** the default heuristic used in the D'Wave framework, fast but doesn't prove minor exclusion.
- **IP-embedding** an optimization based algorithm that can optimize several aspects of the resulting minor.
- **Template embedding** use a template graph, easily and efficiently embeddable in the hardware graph, to find a minor in a deterministic way

In [17] is shown a polynomial time algorithm that establishes if a fixed graph H is contained as a minor in a given graph. However, the interest in developing various algorithms to perform this task comes from the work [18], where it is given the proof of the NP-completeness of the minor-embedding problem when both graphs are part of the input. After that, 2 main results were obtained the first in [19] with an execution time of

$$O(3^{k^2}(h+k-1)!m)$$

improved by the the algorithm presented in [20] with a time of the order

$$O(2^{(2k+1)logk}h^{2k}2^{2h^2}m)$$

where $k = bw(G),\ n = |V(G)|,\ m = |E(G)|$ ($bw(G)$ is the branchwidth of G).

5.1 Minorminer

A complete description and some numerical results regarding this algorithm can be found in [21], where it was firstly introduced, here will be presented the main concepts and ideas.

5.1.1 Algorithm

Let $X,\ Y$ be the hardware and logical graphs respectively, suppose that $y, y_1, ..., y_k \in V(Y)$ and y is adjacent with every $y_1 ... y_k$. The search of a vertex model $\phi(y)$ is done by ensuring that the vertex model shares an edge with each $\phi(y_1), ...\phi(x_k)$.

To do so, for each y_j, it is computed the shortest path between $\phi(y_j)$ and every vertex g of the subgraph of X of the unused nodes. This is then recorded as a cost function $c(g, j)$ and the vertex $g*$ with minimal cost $\sum_j c(g, j)$ is chosen as the root of $\phi(y)$. The remaining part of $\phi(y)$ is constructed by taking the shortest paths between $g*$ and each $\phi(y_j)$.

If no such $g*$ exists, it is possible to temporarily allow for vertex models to overlap, in this case, the vertex model is chosen to be one that uses a minimal amount of vertex already used in other models. The computation of this cost is done by weighing each $g \in V(X)$ by

$$wt(g) = D^{|\{i: g \in \phi(y_i)\}|} \tag{5.1}$$

and then define the weight of a path the sum of the weight of the vertices in the path. This ensures that the root of $\phi(y)$ is chosen with the lowest overlap possible.

The pseudo-code of this algorithm looks like:

findMinorEmbedding(G,H)
Input: graph H with vertices $\{x_1, \ldots, x_n\}$, graph G
Output: vertex-models $\phi(x_1), \ldots, \phi(x_n)$ of an H-minor in G, or "failure".

randomize the vertex order $x_1, \ldots, x_n$
set $stage := 1$
for $i \in \{1, \ldots, n\}$ do
    set $\phi(x_i) := \{\}$
while $\max_{g \in V(G)} |\{i : g \in \phi(x_i)\}|$ or $\sum_i |\phi(x_i)|$ is improving, or $stage \leq 2$
    for $i \in \{1, \ldots, n\}$ do
        for $g \in V(G)$ do
            set $w(g) := \mathrm{diam}(G)^{|\{j \neq i : g \in \phi(x_j)\}|}$
        $\phi(x_i) :=$ findMinimalVertexModel$(G, w, \{\phi(x_j) : x_j \sim x_i\})$
    set $stage := stage + 1$
if $|\{i : g \in \phi(x_i)\}| \leq 1$ for all $g \in V(G)$
    return $\phi(x_1), \ldots, \phi(x_n)$
else
    return "failure"

findMinimalVertexModel($G, w, \{\phi(x_j)\}$)
Input: graph G with vertex weights w, neighbouring vertex-models $\{\phi(x_j)\}$
Output: vertex-model $\phi(y)$ in G such that there is an edge between $\phi(y)$ and each $\phi(x_j)$

if all $\phi(x_j)$ are empty
    return random $\{g^*\}$
for all $g \in V(G)$ and all j do
    if $\phi(x_j)$ is empty
        set $c(g, j) := 0$
    else if $g \in \phi(x_j)$
        set $c(c, g) := w(g)$
    else
        set $c(g, j) :=$ w-weighted shortest-path distance$(g, \phi(x_j))$ excluding $w(\phi(x_j))$
set $g^* := \mathrm{argmin}_g \sum_j c(g, j)$
return $\{g^*\} \cup \{$paths from g^* to each $\phi(x_j)\}$

Figure 5.1: Minorminer heuristic, figure from [21].

5.2 IP-embedding

This approach, presented in [22], is a IP formulation of the equational model developed in [23]. The major advantage of this formulation is that is solvable using commercial solvers, avoiding the expensive computation of the Graver basis.

Moreover, the equational approach has some nice features when compared to other heuristics like *minorminer*:

- Prove infeasibility
- Provide quality guarantees

5.2.1 Variables and Constraints

In the equational approach, embedding a logical graph Y in the hardware graph X is represented by a surjective map $\pi : X \mapsto Y$ such that $\phi(y) = \pi^{-1}(y)$.

The expression for $\pi(x_i)$ is:

$$\pi(x_i) = \sum_{j:y_j \in V(Y)} \alpha_{ij} y_j \; \forall x_i \in V(X) \tag{5.2}$$

where α_{ij} are binary coefficients.

To obtain feasible solutions several constraints are needed:

1. *well definition constraint*

$$\sum_{j:y_j \in V(Y)} \alpha_{ij} \leq 1 \quad \forall x_i \in V(X) \tag{5.3}$$

   This ensures that the qubit x_i embeds at most one logical variable y_j

2. *minumum and maximum size*

$$m \leq \sum_i \sum_j \alpha_{ij} \leq n \tag{5.4}$$

   where $n = |V(X)|$ and $m = |V(Y)|$

3. *fiber size constraint*

$$1 \leq \sum_i \alpha_{ij} \leq k \quad \forall y_j \in V(Y) \tag{5.5}$$

   Bound to the size of the vertex model $|\phi(y_j)|$.
   An additional refinement is added to exclude qubits when their distance is $> k$

$$1 \geq \alpha_{i_1 j} + \alpha_{i_2 j} \forall x_{i_1}, x_{i_2} \in V(X), \min\, d(x_{i_1}, x_{i_2}) > k, \forall y_j \in V(Y) \tag{5.6}$$

4. *fiber condition*
The request of having connected subtrees as vertex models is formalized as:

$$\forall x_{i_1}, x_{i_2} \in \pi^{-1}(y_j) : \alpha_{i_1 j} + \alpha_{i_2 j} + \left(\sum_{c_k(x_{i1}, x_{i2}) \in C_k(x_{i1}, x_{i2})} (\gamma_{c_k j}) - 1 \right) \leq 2 \quad (5.7)$$

where the binary variable $\gamma_{c_k j}$ takes value 1 if the fiber $c_k(x_{i_1}, x_{i_2})$ is in the vertex model of y_j, $C_k(x_{i_1}, x_{i_2})$ is the set of fibers connecting x_{i_1} and x_{i_2} of size $\leq k$. The variables $\gamma_{c_k j}$ are defined as follows

$$\gamma_{c_k j} = \prod_{l : x_l \in int(c_k(x_{i_1}, x_{i_2}))} \alpha_{lj} \quad (5.8)$$

the constraint above is not enough to ensure good fiber condition, to solve this it is added the next constraint

$$\alpha_{ij} + \alpha_{i_2 j} - \sum_{c_k(x_{i_1}, x_{i_2}) \in C_k(x_{i_1}, x_{i_2})} \gamma_{c_k j} \leq 1 \quad (5.9)$$

$$\forall y_j \in V(Y)\ \forall (x_{i_1}, x_{i_2}) \in V(X), (x_{i_1}, x_{i_2}) \notin E(X), min\, d(x_{i_1}, x_{i_2}) \leq k$$

5. *pullback condition*
The requirement 4.32 is equivalent to the constraint

$$1 \leq \sum_{i_1, i_2 : (x_{i_1}, x_{i_2}) \in E(X)} (\delta^{\parallel}_{i_1 i_2 j_1 j_2} + \delta^{\perp}_{i_1 i_2 j_1 j_2}) \quad \forall (y_{j_1}, y_{j_2}) \in E(Y) \quad (5.10)$$

where the variable $\delta^{\parallel}_{i_1 i_2 j_1 j_2}$ have value one if x_{i_1}, x_{i_2} are edges of the vertex models $\phi(y_{j_1})$,$\phi(y_{j_2})$ and $\delta^{\perp}_{i_1 i_2 j_1 j_2}$ is one if x_{i_2}, x_{i_1} are edges of the vertex models $\phi(y_{j_1})$,$\phi(y_{j_2})$. This condition is equivalent to the quadratic constraint

$$\delta^{\parallel}_{i_1 i_2 j_1 j_2} = \alpha_{i_1 j_1} \alpha_{i_2 j_2}$$

$$\delta^{\perp}_{i_1 i_2 j_1 j_2} = \alpha_{i_1 j_2} \alpha_{i_2 j_1}$$

that can be formulated with the following set of linear inequalities

$$\begin{cases} \delta^{\parallel}_{i_1 i_2 j_1 j_2} \leq \alpha_{i_1 j_1} \\ \delta^{\parallel}_{i_1 i_2 j_1 j_2} \leq \alpha_{i_2 j_2} \\ \delta^{\parallel}_{i_1 i_2 j_1 j_2} \geq \alpha_{i_1 j_1} + \alpha_{i_2 j_2} - 1 \end{cases} \quad (5.11)$$

with the following and last constraint it is excluded the option of having $\delta^{\parallel}_{i_1 i_2 j_1 j_2}$ and $\delta^{\perp}_{i_1 i_2 j_1 j_2}$ from having both value one

$$\delta^{\parallel}_{i_1 i_2 j_1 j_2} + \delta^{\perp}_{i_1 i_2 j_1 j_2} \leq 1 \quad \forall (x_{i_1}, x_{i_2}) \in E(X), \forall (y_{j_1}, y_{j_2}) \in E(Y) \tag{5.12}$$

5.2.2 Objective function

Within this framework several objective functions can be adopted depending on the context at hand, given the current limitations of commercial available quantum hardware, the metric considered in this work is the total embedding size

$$\sum_{i:x_i \in V(X)} \sum_{j:y_j \in V(Y)} \alpha_{ij} \quad s.t. \ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\delta}^{\parallel}, \boldsymbol{\delta}^{\perp}) \in F \tag{5.13}$$

where F is the feasible region, intersection of the constraints listed in 5.2.1.

5.3 Template embedding

The key idea of this approach is to identify a graph T called template that has the following properties:

- It is easily and efficiently embeddable in the hardware graph
- Has some nice topological structure that can be exploited to achieve simpler constraints and combinatorial complexity

The application of this concept considered in this work can be found in [24], where two templates are defined for the Chimera graph 4.4.1.

5.3.1 Template graph

Following the work of Goodrich et al. [25], where the problem was solved by finding a minimum size OCT, the template graph is chosen to be a $K_{ML,ML}$ bipartite graph, a minor of $C_{M,M,L}$ Chimera graph (in current hardware $L = 4$) that can be embedded by assigning every left(right)-hand side vertices of each cell (shown in 4.3) in a column(row) to a left(right) vertex of the template graph:

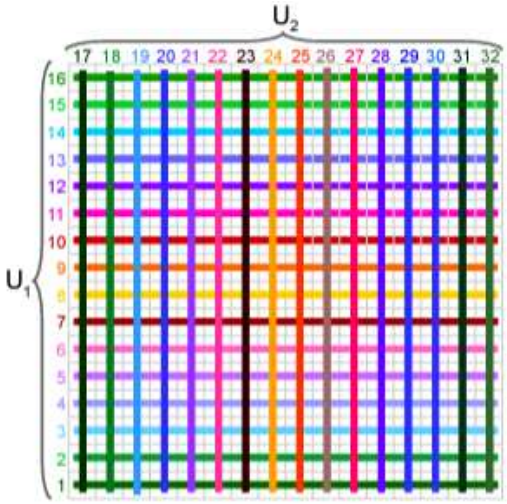


Figure 5.2: $K_{64,64}$ embedded in $C_{16,16,4}$, figure from [24].

5.3.2 Formulation

The following formulation defines the combinatorial problem that finds a minor of the logical graph Y in the template graph. The problem can then be solved using any commerciallly available solver.

Variables For each vertex $y \in V(Y)$ and $k \in \{1, 2\}$ the variable v_{yk} has value one if the vertex y is assigned to the partition k and the variable v'_y denotes if the vertex y is assigned to any partition.

Objective Function The goal is to assign as many vertices as possible, this is done by maximizing the number of v'_y variables with value one, this can be translated as objective function as follows:

$$max \sum_{i:y_i \in V(Y)} v'_{y_i} \tag{5.14}$$

Constraints

- the association between variables is

$$v'_y \leq v_{y1} + v_{y2} \quad \forall y \in V(Y)$$

- no more than ML vertices per partition

$$\sum_{y} v_{yk} \leq ML \quad \forall k \in \{1, 2\}$$

- for each edge $(y_i, y_j) \in E(Y)$, vertices y_i, y_j should not be assigned to a single and same partition

$$v_{y_I 1} + v_{y_j 1} - v_{y_i 2} - v_{y_j 2} \leq 1$$
$$v_{y_I 2} + v_{y_j 2} - v_{y_i 1} - v_{y_j 1} \leq 1$$

This formulation has some nice features: by solving the problem described above a certificate of embeddability via a bipartite template is obtained. This also implies that if the algorithm used to get the solution finds an upper bound on the objective function that is lower than $|V(Y)|$, then the search can be terminated. Moreover, it can be generalized to N-partite templates, as demonstrated for the $N = 4$ case in [24].

5.4 Experiments

Here are presented the results of the experiments where each of the embedding algorithms presented in this chapter was tested.

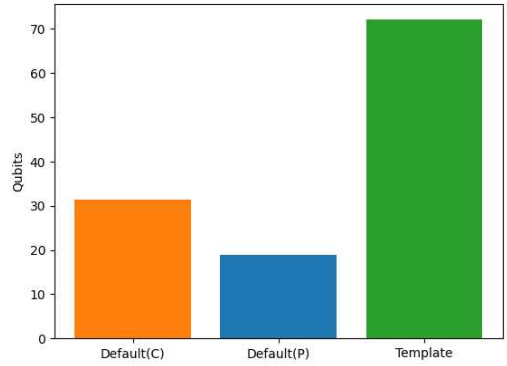
The experiments were conducted by embedding two sets of 100 randomly generated instances with the following parameters

	N_{jobs}	$N_{machines}$	$Duration_{max}$	$Timespan_{max}$
Set 1	2	3	1	5
Set 2	3	3	2	8

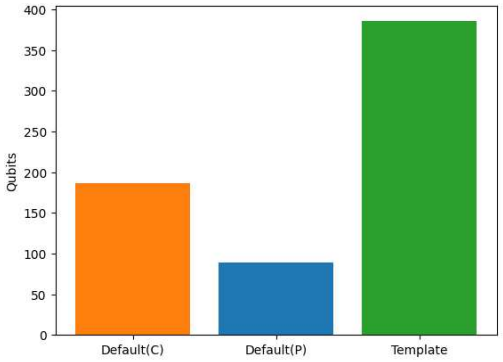
Table 5.1: Instance parameters for embedding testing

The following results will be relative to template embedding on Chimera and minorminer on both Pegasus and Chimera. The IP approach proved to be too complex for the resources available for this work even on the smaller set of instances, Some experimental results about it will be presented later.

5.4.1 Embedding size and Chain length

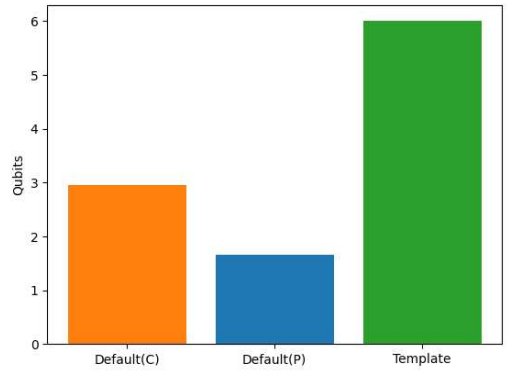


(a) Set 1

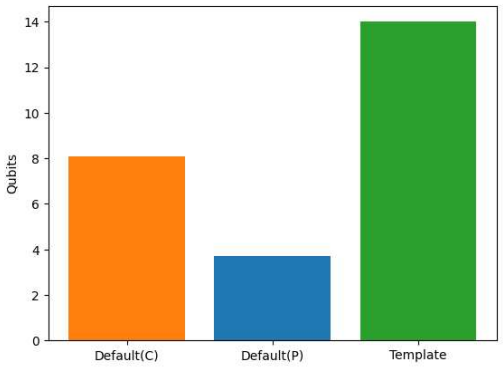


(b) Set 2

Figure 5.3: Embedding size results



(a) Set 1



(b) Set 2

Figure 5.4: Embedding longest chain

As expected, embedding on a denser graph lead to lower usage of qubits and to shorter chains.

5.4.2 Energy landscape

Here the energy spectrum of the solutions obtained through simulated annealing (5000 reads and 1000 sweeps). The considered instance is the following:

	T1	T2	T3
Job 1	M2,1	M1,1	M3,1
Job 2	M1,1	M2,1	M3,1

Table 5.2: Instance description, each task is represented by a couple (machine, duration).

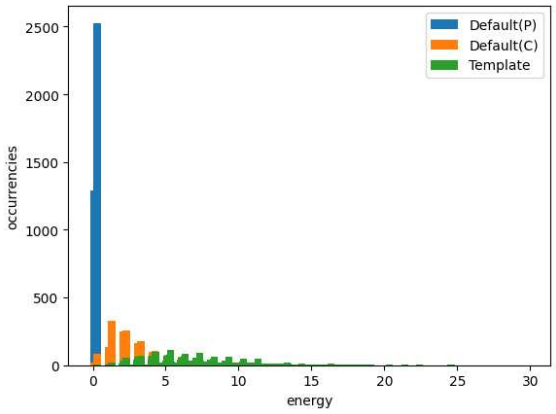


Figure 5.5: Energy spectrum of SA solutions.

It is possible to notice that, while in every case at least one optimal solution was found, the distribution of the energies greatly benefits from the higher density of the Pegasus graph.

6

GAMA algorithm

In this chapter, the hybrid algorithm developed in [26] will be presented. This algorithm obtains an optimal solution of any ILP (even with a non-linear objective), expanding the scope of the current quantum hardware.

Given a problem with the following form:

$$\begin{cases} min\ f(\mathbf{x}) \text{ s.t. } x \geq 0, x \in \mathbb{Z} \\ A\mathbf{x} = \mathbf{b} \end{cases} \tag{6.1}$$

where $\mathbf{x}$ is the vector of n variables, A is a $m \times n$ matrix and b is a vector of length m. The main idea of this procedure is to extract the Graver basis of the linear constraint by QA and then augment a sub-optimal feasible solution by adding the elements of the Graver basis to it.

6.1 Graver basis definition & properties

The Graver basis is part of a family of sets that are called *optimality certificate* or, more formally:

Definition 1 *A set $S \in \mathbb{Z}^n$ is called a optimality certificate if for any feasible sub-optimal solution x_0 of 6.1 exists t in S and $\lambda \in \mathbb{Z}^+$ such that $f(x_0 + \lambda t) < f(x_0)$.*

Given the partial ordering, called *conformal*, denoted with $\sqsubseteq$ defined as:

Definition 2 *Given $x, y \in \mathbb{Z}$, x is said to be conformal to y, $y \sqsubseteq x$ if: $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$.*

The definition of Graver basis is:

Definition 3 *The Graver basis of an integer matrix A is the set of $\sqsubseteq$ minimal elements in the lattice $L^*(A)$, denoted as $G(A)$.*

Where $L^*(A)$ is the kernel of A

$$L^*(A) = \{\mathbf{x} | A\mathbf{x} = 0, \mathbf{x} \in \mathbb{Z}^n, A \in \mathbb{Z}^{m \times n}\} \setminus \{0\} \tag{6.2}$$

With the above definitions, the following propositions hold:

Proposition 1 *the following statements regarding the Graver basis are true:*

1. *Every vector in the lattice $L^*(A)$ is a conformal sum of the Graver basis elements.*
2. *The upper bound on the number of Graver basis elements required to express a kernel vector is $(2n - 2)$.*
3. *The Graver basis is an optimality certificate for the problem 6.1.*
4. *For any $g \in G(A)$, the upper bound on the norm is:*

$$\|g\|_\infty \leq (n - r)\Delta(A) \text{ and} \|g\|_1 \leq (n - r)(r + 1)\Delta(A) \tag{6.3}$$

where $r = rank(A)$ and $\Delta(A)$ is the maximum absolute value of the determinant of a square submatrix of A.

6.2 Graver basis extraction

The extraction of the Graver basis can be summarized by the iteration of few steps:

- Build the QUBO for the extraction of the kernel elements.
- Solve the above mentioned QUBO via QA.
- Classical post-processing.

6.2.1 QUBO for kernel extraction

Since quantum annealers can only deal with binary variables, a mapping of the problem $A\mathbf{x} = 0$ to a binary counterpart. The solution to this task proposed in [26] is to encode the integer variables to binary ones using a linear mapping $x_i = \mathbf{e}_i X_i$ where $\mathbf{e}_i = (e_{i,1}, e_{i,2}, ..., e_{i,k_i})$ is the linear encoding vector and $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,k_i})$ is the vector that encodes the x_i variable. With a binary encoding vector $\mathbf{e}_i = (2^0, 2^1, ..., 2^{k_i})$, $k = \lceil \log_2 \Delta_i \rceil$ where Δ_i is the difference between the upper bound and the lower bound of the x_i variable the whole transformation can be summarized as

$$\mathbf{x} = \mathbf{L} + \mathbf{E}\mathbf{X} \tag{6.4}$$

where $\mathbf{E}$ is a $n \times \sum_i k_i$ matrix that encodes $\mathbf{x}$ into a binary vector $\mathbf{X} \in \{0, 1\}^{\sum k_i}$.

The problem of finding the kernel of the matrix A can be formulated via the following QUIO:

$$\min \mathbf{x}^\intercal Q \mathbf{x},\ Q = A^\intercal A,\ \mathbf{x} \in \mathbb{Z}^n \tag{6.5}$$

Substituting 6.5 in 6.4 the result is:

$$\mathbf{x}^\intercal Q \mathbf{x} = \mathbf{X}^\intercal \mathbf{E}^\intercal Q \mathbf{E}\mathbf{X} + (\mathbf{X}^\intercal \mathbf{E}^\intercal Q \mathbf{L} + \mathbf{L}^\intercal Q \mathbf{E}\mathbf{X}) + \mathbf{L}^\intercal Q \mathbf{L} \tag{6.6}$$

The last term of this equation is a constant and can be ignored while the linear term can be rewritten as $\mathbf{X}^\intercal diag(2\mathbf{L}^\intercal Q \mathbf{E})\mathbf{X}$. In this setup the QUBO simplifies to:

$$\min \mathbf{X}^\intercal Q_b \mathbf{X} \tag{6.7}$$

where $Q_b = \mathbf{E}^\intercal Q \mathbf{E} + diag(2\mathbf{L}^\intercal Q \mathbf{E})$

6.2.2 Sampling

The sampling of low energy solutions of 6.7 is done via QA in the same way explained in chapter 4.

6.2.3 Post-processing

In this phase of the algorithm a series of three classical post-processing operations are performed:

1. Combination of near-optimal solutions into optimal ones.

2. Encoding adaptation.

COMBINATION OF NEAR-OPTIMAL SOLUTIONS is the process that given the matrix of solutions $\mathbf{x}_u \in \mathbb{Z}^{n \times N}$ returns more optimal solutions.

Consider the matrix $A\mathbf{x}_u = E_r$ that contains the errors for each of the N solutions. Gathering the solutions with sum of absolute errors it is possible to sum them pairwise to obtain new optimal solutions, the procedure for sum of errors equal $1, 2$ is explained below.

- **Solutions with total sum error = 1:** the columns with total sum error one can be divided into two subsets, the first containing the ones with $+1$ and the other with -1 error. It is possible then to perform three sets of operations: subtracting all $+1$ columns pairwise, subtracting all -1 columns pairwise and adding any $+1$ columns to any -1 column. The computational cost of those operations is $O(\frac{3N_{u1}^2}{4m})$

- **Solutions with total sum of error = 2:** the process for solutions with $+2$ and -2 is similar to what have been done to the ± 1. The remaining solutions can be divided in four blocks $(+1,+1), (-1,-1), (-1,+1), (+1,-1)$, for each one of these, a pairwise subtraction lead to an optimal solution. Additionally it is possible to make subtractions among the groups $(+1,+1), (-1,-1)$ and $(+1,-1), (-1,+1)$. The processing of the solution with error 2 requires an additional $O(\frac{3N_{u2}^2}{4m} + \frac{3}{8}N_{u_2}^2)$ operations in total.

It is indeed possible to extend this reasoning for solutions with higher energy.

ENCODING ADAPTATION: since the choice of encoding let the user independently choose the number k_i of qubits to allocate to each variable, it is possible to efficiently adapt the range covered $[L_i, L_i + 1_{k_i}\mathbf{e}_i]$. The methode proposed in [26] is to check if some variables of a sub-optimal solution lays on the border of the range of values represented by the current choice of k_i. If this happen, a better solution may be beyond that bound so the center of the encoding is shifted towards that point and/or the encoding size is incremented by one. A scheme of the algorithm is the following:

Algorithm 3 Adaptive Adjustments

input (iteration i) Middle points vector M_i, Encoding lengths vector K_i, suboptimals vector x_i

output Middle points vector M_{i+1}, Encoding lengths vector K_{i+1}
Calculate saturated borders: *right border* and $left$ *border* based on M_i and K_i
\# Adaptive adjustment of middle points
if $(x_i == right\ border(M_i, K_i))$ **then**
    $M_{i+1} = M_i + 1$
else if $(x_i == left\ border(M_i, K_i))$ **then**
    $M_{i+1} = M_i - 1$
end if
\# Adaptive adjustment of encoding lengths
if $|x_i - M_i| \leqslant 2^{K_i - 1}$ **then**
    $K_{i+1} = K_i - 1$
end if
return M_{i+1} and K_{i+1}

Figure 6.1: Adaptive Encoding, figure from [26].

6.3 Obtaining feasible solutions

The last piece needed for this algorithm is the extraction of one or more feasible solutions to augment them via Graver basis. This is done by solving the constraints equations $A\mathbf{x} = \mathbf{b}$, which is equivalent to solve the QUIO:

$$\min \mathbf{x}^\intercal Q\mathbf{x} - 2\mathbf{b}^\intercal A\mathbf{x} \tag{6.8}$$

where $Q = A^\intercal A$. Using the same encoding present in Section 6.2.1, the resulting QUBO is:

$$\min \mathbf{X} Q_b \mathbf{X} \tag{6.9}$$

where $\mathbf{X} \in \{0,1\}^{\sum k_i}$ and $Q_b = \mathbf{E}^\intercal Q\mathbf{E} + 2diag[(\mathbf{L}^\intercal Q - \mathbf{b}^\intercal A)\mathbf{E}]$. Solutions can then be obtained via QA as done before.

6.4 Algorithm

The algorithm itself goes through three phases:

1. Graver basis extraction, as discussed in 6.2.

2. Find initial feasible points.

3. Augment to better solutions applying the third statement of 1.

Since in the second step QA is used and it will return several unique feasible solutions, it is possible to collect, among those points, the ones with lower cost value and then augment them in parallel. This is a good practice since there are no guarantees that the full Graver basis is extracted during the first phase, in fact most probably only a fraction of the elements will be available, thus it may be the case that not every initial point can be augmented in an optimal one.

7

Job Shop Scheduling

In this chapter two different formulations of JSS will be presented. The first, in 7.1, is a QUBO formulation that can be directly compiled and solved using QA while the second one takes the form of a ILP and will be solved by the hybrid algorithm GAMA depicted in 6, where the Graver basis extraction is performed via QA [27].

The JSS can be briefly described as follows; given a set M of n_m machines and given a set J of n_J jobs, suppose that each job is composed of at most n_m tasks each that has to be executed on a different machine and with a certain order. In particular, the rules that a schedule has to comply with to be valid are:

- the tasks of a given job have to be executed in the given order
- each task can start only one time
- each job can be performed only on one machine at a time
- each machine can process one task at a time

The goal is to find, among every feasible schedule, the optimal one, the definition of this optimality is context dependent, in this case the best schedule will be the one with shortest total makespan. During the rest of this work N will be the number of jobs, M the number of machines.

7.1 Time Indexing Formulation

This formulation of the JSS, present in [28], is expressed as a QUBO by translating the constraints in penalty terms. The operations are indexed in lexicographical order as follows:

$$\begin{aligned} &\mathbf{j}_1 = \{O_1 \to ... \to O_{k_1}\} \\ &\mathbf{j}_2 = \{O_{k_1+1} \to ... \to O_{k_2}\} \\ ... \\ &\mathbf{j}_N = \{O_{k_{N-1}} \to ... \to O_{k_N}\} \end{aligned} \tag{7.1}$$

Given a deadline T, to each operation is assigned a set of binary variables

$$x_{i,t} = \begin{cases} 1 : \text{operation } O_i \text{ starts at time } t \\ 0 : \text{otherwise} \end{cases} \qquad \forall t \in \{1, ..., T\} \tag{7.2}$$

7.1.1 Constraints

The order condition and the third rule of the list above are enforced by the following quadratic penalty term

$$\sum_{\substack{k_{n-1}<i<k_n \\ t+p_i>t'}} x_{i,t}x_{i+1,t'} \; \forall n \in \{1, .., N\} \tag{7.3}$$

The constraint and the associated quadratic penalty term relative to the second rule are

$$\left(\sum_t x_{i,t} = 1 \; \forall i \in \{1, ..., k_N\}\right) \to \sum_i \left(\sum_t x_{i,t} - 1\right)^2 \tag{7.4}$$

The remaining condition can be represented by the quadratic constraint

$$\sum_{(i,t,k,t') \in R_m} x_{i,t}x_{k,t'} = 0 \; \forall m \in \{1, ..., M\} \tag{7.5}$$

where the set $R_m = A_m \cap B_m$ and

$$\begin{aligned} A_m = &\{(i, t, k, t') : (i, k) \in I_m \times I_m, \\ &i \neq k, 0 \leq t, t' \leq T, 0 < t' - t < p_i\} \end{aligned} \tag{7.6}$$

$$\begin{aligned} B_m =& \{(i,t,k,t') : (i,k) \in I_m \times I_m, \\ & i < k, t' = t, p_i > 0, p_j > 0\} \end{aligned} \tag{7.7}$$

The set A_m is forbids the start of the operation O_j from starting if the operation O_i is still being processed. The set B_m exclude the possibility of two operation from starting at the same time unless one of them has zero duration.

Summing up every penalty term listed above results in the Hamiltonian

$$H_T(\mathbf{x}) = \alpha h_1(\mathbf{x}) + \beta h_2(\mathbf{x}) + \gamma h_3(\mathbf{x}) \tag{7.8}$$

during the rest of this work the values $\alpha = \beta = \gamma = 1$ will be considered.

7.1.2 Formulation Refinement

The above described formulation can be improved in two aspects by some simple refinements with a polynomial overhead.

- The valid time window of each operation can be shaved, effectively reducing the amount of variables needed.
- More information can be extracted by each QA runs by discriminating the longest timespans.

Pruning

The first step toward the elimination of superfluous variables is the computation of head and tail of each operation, respectively referred as r_i and h_i.

To complete this task several algorithms have been developed, one of them was presented in [29] commonly called iterated Carlier and Pinson (ICT) with a computational cost of $O(N^2M^2Tlog(N))$. Here is discussed a simpler approach to this task, the heads r_i are computed as the sum of the execution times of each preceding operations and the tails h_i are treated in a similar way. After this, it is possible to refine those values in the following way: suppose the existence of 2 operations O_a, O_b with $a, b \in I_J$ such that

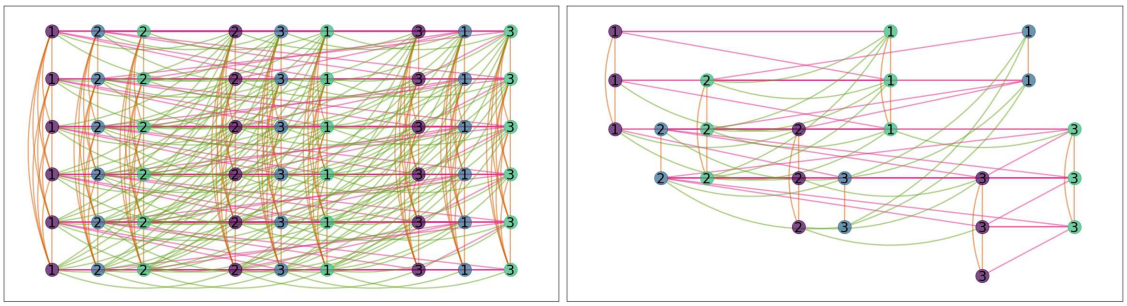
$$r_a + p_a + p_b + q_b > T \tag{7.9}$$

then O_a has to be executed after O_b, this means updating $r_a = max\{r_a, r_b + p_b\}$. The updates of the tails are done in a similar fashion.

After this initial update phase, it is possible prune the problem by fixing to zero the value of the variables that correspond to invalid starting time

$$x_{i,t} = 0 \; \forall t \notin [r_i, T - p_i - q_i]$$

It is important to consider the fact that if a time window shuts completely it means that there aren't valid schedules within the deadline T. Here and example of the effect of pruning on a random instance with three jobs and three machines.



(a) Pre-pruning problem graph **(b)** Pruned problem graph

Figure 7.1: Pruning example, the orange edges are relative to the start only once constraint, the pink ones are the one task per machine constraints and the green ones represent the operation order constraint.

Timespan discrimination

To discriminate among the valid solutions, it is possible to add ate the Hamiltonian 7.8 a term that breaks the degeneracy of the ground level. To avoid overlapping with the invalid solutions, the maximum energy gap added with this refinement has to be $\Delta E < min\{\alpha, \beta, \gamma\}$. Doing so will be helpful during the solution, improving the search speed of the right deadline T. The method used in this work to discriminate among the K longest schedules is the following:

First define the sequence of energy penalties

$$\Delta E_k = \left(\frac{1}{M+1}\right)^{k+1} \tag{7.10}$$

This choice of energy gap guarantees that a longer schedule will always have a higher energy w.r.t. a shorter one and will keep the total energy of a valid solution below the threshold

$min\{\alpha, \beta, \gamma\}$ that separates unfeasible schedules from valid ones.

7.2 Disjunctive Formulation

The disjunctive model, firstly introduced in [30] and then carefully reviewed and compared to other MIP formulations in [31], where it was found to be the most efficient one.

The decision variables x_{ij} are the integer start time of job j on the machine i.

7.2.1 Constraints and Objective function

A formal definition of this model starts with the definition of the decision variables x_{ij} are the integer start time of job j on the machine i. Moreover it is convenient to refer to the h-th operation of job j as σ_h^j.

The objective function is represented by an integer support variable C_{max}, constrained to being at least larger than the global makespan. The constraints and their meaning are listed below.

- The starting time of each operation has to be non-negative

$$x_{ij} \geq 0 \; \forall i \in \{1, ..., M\}, \; \forall j \in \{1, ..., N\}$$

- The precedence constraint is expressed as

$$x_{\sigma_h^j j} \geq x_{\sigma_{h-1}^j j} + p_{\sigma_{h-1}^j j} \forall j \in \{1, ..., N\}, \; \forall h \in \{2, ..., M\}$$

- To ensure that two jobs can't be assigned to the same machine at the same time the following constraints are needed

$$x_{ij} \geq x_{ik} + p_{ik} - V \cdot z_{ijk} \; \forall j, k \in \{1, ..., N\}, j < k, \; \forall i \in \{1, ..., M\}$$

$$x_{ik} = x_{ij} + p_{ij} - V \cdot (1 - z_{ijk}) \; \forall j, k \in \{1, ..., N\}, j < k, \; \forall i \in \{1, ..., M\}$$

  where z_{ijk} has value one if job j precedes job k on machine i, zero otherwise and $V = \sum_j \sum_i p_{ij}$

- The before mentioned constraint on the objective function C_{max} is

$$C_{max} \geq x_{\sigma_M^j j} + p_{\sigma_M^j j} \; \forall j \in \{1, ..., N\}$$

8 Experiments

Due to the very limited resources available, the focus of the experiments were set on two particular instances:

	T1	T2	T3
Job 1	M1,2	M3,2	M2,2
Job 2	M1,1	M2,2	M3,1
Job 3	M2.2	M3,2	M1,2

Table 8.1: a3 instance

	T1	T2	T3	T4
Job 1	M4,2	M2,2	M3,1	M1,1
Job 2	M4,1	M2,2	M3,3	M1,3
Job 3	M1.1	M3,3	M4,1	M2,2
Job 4	M2,1	M1,2	M4,1	M3,1

Table 8.2: a4 instance

for each of those instances, three different methods were used to obtain a solution: Gurobi, SA and QA. For a3 8 every approach was able to obtain equivalent solutions (QA returned a solution with equal timespan as the other algorithms but with higher energy), while a4 8 QA

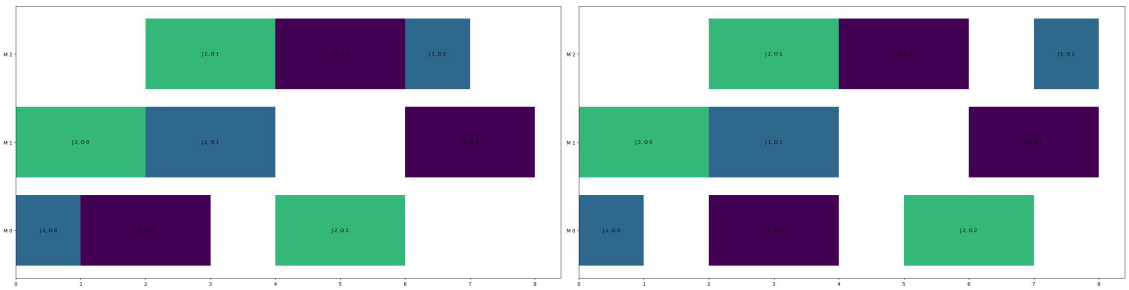
was not able to find a feasible solution before running out of computational time (1min/month for the free plan available for this project).

Follows the timing, energy and representation of the best solutions found. The settings for the experiments are:

- **Gurobi**: default settings
- **SA**: num_reads = 5000, num_sweeps = 2000.
- **QA**: num_reads = 2600, annealing_time = 200μs.

8.1 A3 RESULTS

As mentioned above, for this smaller instance every method adopted was able to find an optimal solution, in particular the solutions found were the following (the SA solution is omitted since it is equivalent to the Gurobi one):



(a) Gurobi solution **(b)** QA solution

Figure 8.1: solutions for the a3 instance

The energies relative to those solutions are 0.328 for the Gurobi one and 0.5625 for the one obtained through QA, it is important to notice that for the scope of these experiments the two solutions should be considered both optimal since both have the same total makespan 8. The difference in energy comes form the timespan discrimination method presented in 7.

The time needed to obtain such solutions is the following (the time for the embedding is excluded since the embedding can be stored and there is no need to compute it every time):

	time(s)
QA	0.03
SA	8
Gurobi	0.676

Table 8.3: a3 solution timing

8.2 A4 RESULTS

The logical graph relative to this instance is the following:

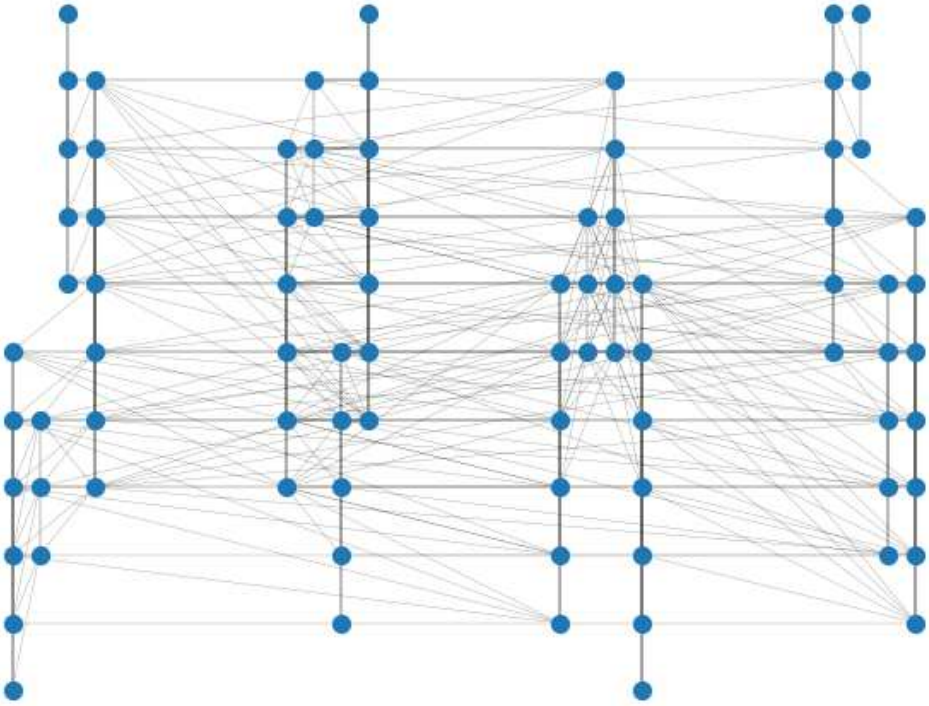
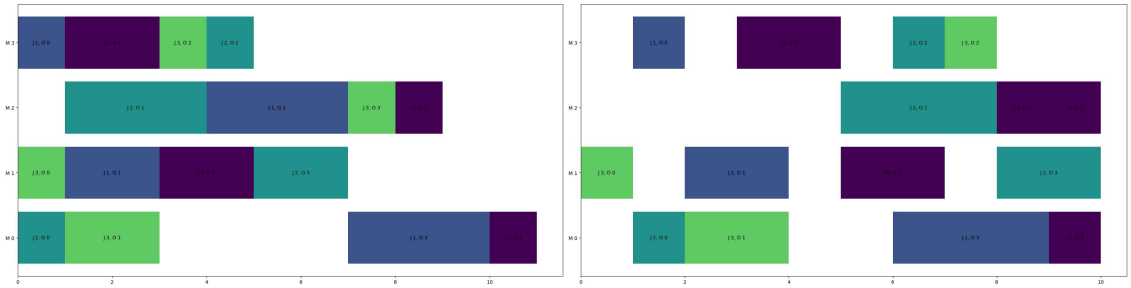


Figure 8.2: a4 logical graph

.

To tackle this instance with QA multiple runs (until resource depletion) were made maintaining the same parameters listed above. As for the a3 instance the SA solution is omitted for

the same reason.



(a) Gurobi solution **(b)** QA solution

Figure 8.3: solutions for the a4 instance

It is easy to see that the QA solution present several irregularities, confirmed by the associated energy of 5.0880, while the Gurobi obtain the optimal solution with energy 0.24192.

The time needed by each algorithm is:

	time(s)
QA	123
SA	27
Gurobi	123

Table 8.4: a4 solution timing, the time relative to QA takes under consideration every run made with the available resources.

9 Conclusion

To conclude this dissertation there will be some comments on the various techniques analyzed.

9.1 Minor-Embedding techniques

Among the algorithms used for the minor embedding the one that demonstrated superior performances is the minor-miner, the default heuristic of the D'Wave API. The other approaches can still be applied in different scenarios, i.e. the IP-embedding may find a good application in finding optimal embedding of small portions of the Hamiltonian in the form of gadgets in a similar fashion as what has been done in [32]. For the template approach, future work may find a good template for the newest generation of quantum hardware such as Pegasus and Zephyr, but even in that case, it seems that to achieve good performances with this method the logical graph has to be very dense.

9.2 QA

The quantum annealing algorithm proved to be feasible and to outperform the simulated counterpart on small instances like a3 but failed systematically to obtain feasible solutions increasing the instance size to a4. This suggests that with the current hardware, the scope of QA may be to solve some smaller sub-problems in the context of a more complex heuristic/metaheuristic.

9.3 GAMA algorithm

This algorithm represents a good opportunity to widen the scope of QA but it was not possible to test it due to the scarcity of resources available, it worked on small toy examples replacing the QA Graver basis extraction with a simulated counterpart but it was impossible to execute further tests.

References

[1] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: A survey," *Journal of Combinatorial Optimization*, vol. 28, 07 2014.

[2] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960. [Online]. Available: http://www.jstor.org/stable/1910129

[3] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991. [Online]. Available: http://www.jstor.org/stable/2030652

[4] E. Çela and A. P. Punnen, *Complexity and Polynomially Solvable Special Cases of QUBO*. Cham: Springer International Publishing, 2022, pp. 57–95. [Online]. Available: https://doi.org/10.1007/978-3-031-04520-2_3

[5] F. Glover and M. Laguna, *Tabu Search*. Boston, MA: Springer US, 1998, pp. 2093–2229. [Online]. Available: https://doi.org/10.1007/978-1-4613-0303-9_33

[6] F. Glover, G. A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," *Management Science*, vol. 44, no. 3, pp. 336–345, 1998. [Online]. Available: http://www.jstor.org/stable/2634672

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.220.4598.671

[8] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: http://dx.doi.org/10.3389/fphy.2014.00005

[9] D. de Falco and D. Tamascelli, "An introduction to quantum annealing," *RAIRO - Theoretical Informatics and Applications*, vol. 45, no. 1, pp. 99–116, jan 2011. [Online]. Available: https://doi.org/10.1051%2Fita%2F2011013

[10] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *Journal of Mathematical Physics*, vol. 49, no. 12, dec 2008. [Online]. Available: https://doi.org/10.1063%2F1.2995837

[11] S. Suzuki and M. Okada, "Residual energies after slow quantum annealing," *Journal of the Physical Society of Japan*, vol. 74, no. 6, pp. 1649–1652, 2005. [Online]. Available: https://doi.org/10.1143/JPSJ.74.1649

[12] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.*, vol. 79, pp. 325–328, Jul 1997. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.79.325

[13] J. Roland and N. J. Cerf, "Quantum search by local adiabatic evolution," *Phys. Rev. A*, vol. 65, p. 042308, Mar 2002. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.65.042308

[14] D'wave. (2023) D'wave qpu architecture: Topologies. [Online]. Available: https://docs.dwavesys.com/docs/latest/c_gs_4.html#getting-started-topologies

[15] N. Dattani, S. Szalay, and N. Chancellor, "Pegasus: The second connectivity graph for large-scale quantum annealing hardware," 2019.

[16] D'wave. (2023) Constraints example: Minor-embedding. [Online]. Available: https://docs.dwavesys.com/docs/latest/c_gs_7.html

[17] N. Robertson and P. Seymour, "Graph minors .xiii. the disjoint paths problem," *Journal of Combinatorial Theory, Series B*, vol. 63, no. 1, pp. 65–110, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0095895685710064

[18] J. Matoušek and R. Thomas, "On the complexity of finding iso- and other morphisms for partial k-trees," *Discrete Mathematics*, vol. 108, no. 1, pp. 343–364, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X9290687B

[19] I. Hicks, "Branch decompositions and minor containment," *Networks*, vol. 43, pp. 1–9, 01 2004.

[20] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos, "Faster parameterized algorithms for minor containment," *Theoretical Computer Science*, vol. 412, no. 50,

pp. 7018–7028, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397511007912

[21] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," 2014.

[22] D. E. Bernal, K. E. C. Booth, R. Dridi, H. Alghassi, S. Tayur, and D. Venturelli, "Integer programming techniques for minor-embedding in quantum annealers," 2019.

[23] R. Dridi, H. Alghassi, and S. Tayur, "A novel algebraic geometry compiling framework for adiabatic quantum computations," 2018.

[24] T. Serra, T. Huang, A. Raghunathan, and D. Bergman, "Template-based minor embedding for adiabatic quantum optimization," 2021.

[25] T. D. Goodrich, T. S. Humble, and B. D. Sullivan, "Optimizing adiabatic quantum program compilation using a graph-theoretic framework," 2017.

[26] H. Alghassi, R. Dridi, and S. Tayur, "Gama: A novel algorithm for non-convex integer programs," 2019.

[27] ——, "Graver bases via quantum annealing with application to non-linear integer programs," 2019.

[28] D. Venturelli, D. J. J. Marchand, and G. Rojo, "Quantum annealing implementation of job-shop scheduling," 2016.

[29] J. Carlier and E. Pinson, "Adjustment of heads and tails for the job-shop problem," *European Journal of Operational Research*, vol. 78, no. 2, pp. 146–161, 1994, project Management and Scheduling. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377221794903794

[30] A. S. Manne, "On the job-shop scheduling problem." *Operations Research 8(2):219-223.*, 1960.

[31] W.-Y. Ku and J. C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," *Computers & Operations Research*, vol. 73, pp. 165–173, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054816300764

[32] N. Dattani and N. Chancellor, "Embedding quadratization gadgets on chimera and pegasus graphs," 2019.