INTRODUCTION TO D-wave COMPUTERS

Gabriella Bettonte
Mail: g.bettonte@cineca.it

Website: https://www.quantumcomputinglab.cineca.it/

Thanks to Daniele Ottaviani for the slides

CINECA QUANTUM COMPUTING LAB

The Quantum Annealing Algorithm

- From the moment of publication of this paper to the first realization of a machine prototype capable of implementing this algorithm **there is a gap of 14 years!**
- First Quantum Annealer model from **D-Wave** in 2012

PHYSICAL REVIEW E VOLUME 58, NUMBER 5 NOVEMBER 1998

Quantum annealing in the transverse Ising model

Tadashi Kadowaki and Hidetoshi Nishimori
Department of Physics, Tokyo Institute of Technology, Oh-okayama, Meguro-ku, Tokyo 152-8551, Japan
(Received 30 April 1998)

We introduce quantum fluctuations into the simulated annealing process of optimization problems, aiming at faster convergence to the optimal state. Quantum fluctuations cause transitions between states and thus play the same role as thermal fluctuations in the conventional approach. The idea is tested by the transverse Ising model, in which the transverse field is a function of time similar to the temperature in the conventional method. The goal is to find the ground state of the diagonal part of the Hamiltonian with high accuracy as quickly as possible. We have solved the time-dependent Schrödinger equation numerically for small size systems with various exchange interactions. Comparison with the results of the corresponding classical (thermal) method reveals that the quantum annealing leads to the ground state with much larger probability in almost all cases if we use the same annealing schedule. [S1063-651X(98)02910-9]

PACS number(s): 05.30.−d, 75.10.Nr, 89.70.+c

I. INTRODUCTION

The technique of simulated annealing (SA) was first proposed by Kirkpatrick *et al.* [1] as a general method to solve optimization problems. The idea is to use thermal fluctuations to allow the system to escape from local minima of the cost function so that the system reaches the global minimum under an appropriate annealing schedule (the rate of decrease of temperature). If the temperature is decreased too quickly, the system may become trapped in a local minimum. Too slow annealing, on the other hand, is practically useless although such a process would certainly bring the system to the global minimum. Geman and Geman proved a theorem on the annealing schedule for a generic problem of combinatorial optimization [2]. They showed that any system reaches the global minimum of the cost function asymptotically if the temperature is decreased as $T=c/\ln t$ or slower, where c is a constant determined by the system size and other structures of the cost function. This bound on the annealing schedule may be the optimal one under generic conspecific model system, rather than to develop a general argument, to gain insight into the role of quantum fluctuations in the situation of optimization problem. Quantum effects have been found to play a very similar role to thermal fluctuations in the Hopfield model in a transverse field in thermal equilibrium [5]. This observation motivates us to investigate dynamical properties of the Ising model under quantum fluctuations in the form of a transverse field. We therefore discuss in this paper the transverse Ising model with a variety of exchange interactions. The transverse field controls the rate of transition between states and thus plays the same role as the temperature does in SA. We assume that the system has no thermal fluctuations in the QA context and the term ''ground state'' refers to the lowest-energy state of the Hamiltonian without the transverse field term.

Static properties of the transverse Ising model have been investigated quite extensively for many years [6]. There have, however, been very few studies on the dynamical behavior of the Ising model with a transverse field. We refer to the work by Sato *et al.* who carried out quantum Monte

Optimization problems

The *standard form* of a continuous optimization problem is[1]

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

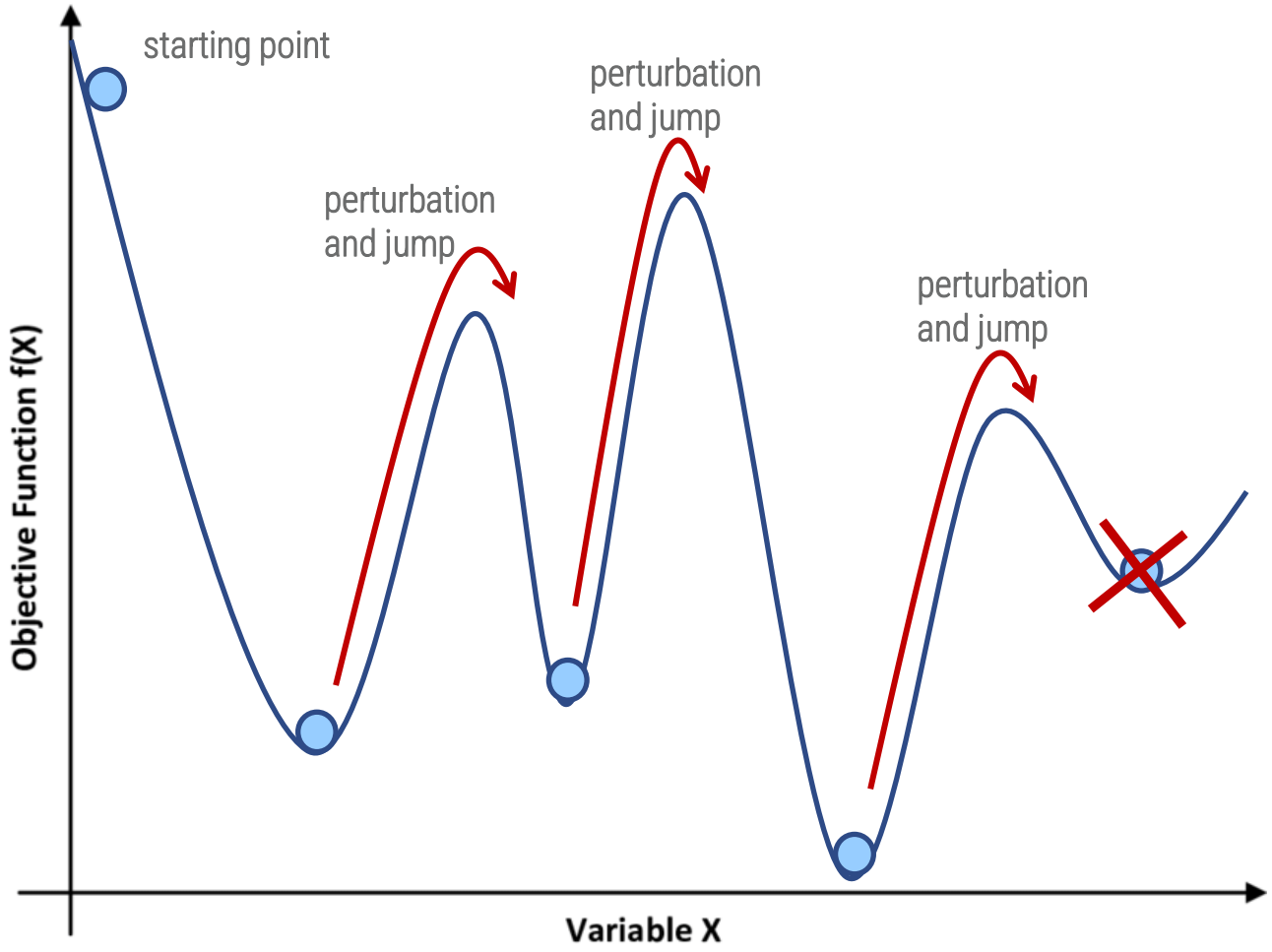
where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the **objective function** to be minimized over the n-variable vector x,
- $g_i(x) \leq 0$ are called **inequality constraints**
- $h_j(x) = 0$ are called **equality constraints**, and
- $m \geq 0$ and $p \geq 0$.

If $m = p = 0$, the problem is an unconstrained optimization problem. By convention, the standard form defines a **minimization problem**. A **maximization problem** can be treated by negating the objective function.

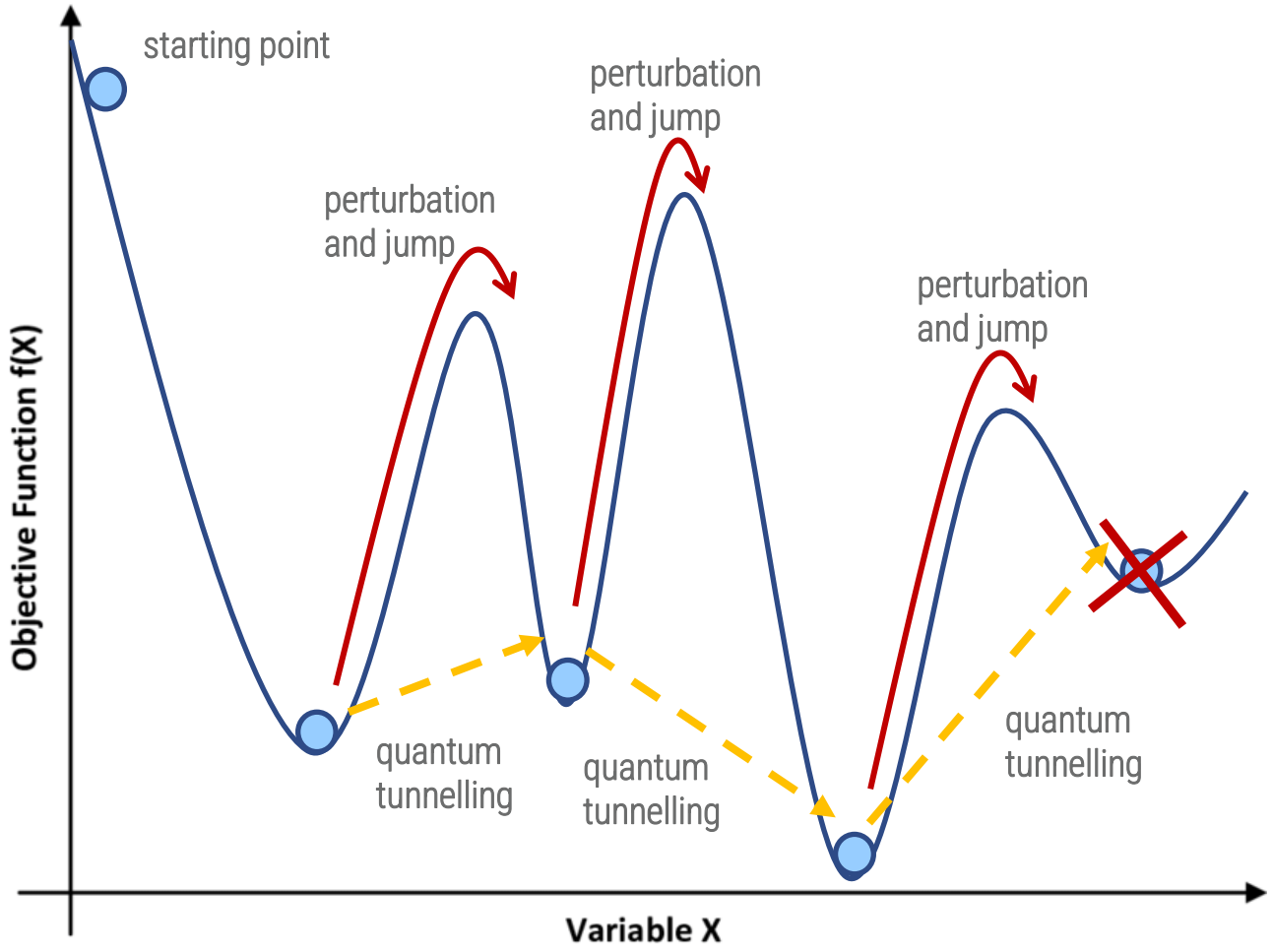
Annealing Algorithms

- Fortunately, brute force is **not** the only algorithm for solving problems of this type.
- There are many algorithms capable of identifying the optimal point of an objective function without having to analyze each of its points
- One of these is known as **Simulated Annealing**
- Simulated annealing is a probabilistic strategy used to solve optimization problems
- Without going into too much detail of the algorithm, we can explain the idea behind it by thinking of a **ball that rolls along the graph of a function, falling into the holes defined by the minima.**
- Every time a ball lands in a hole it receives a certain amount of energy, enough to make it jump over another piece of graph, looking for deeper minima points.



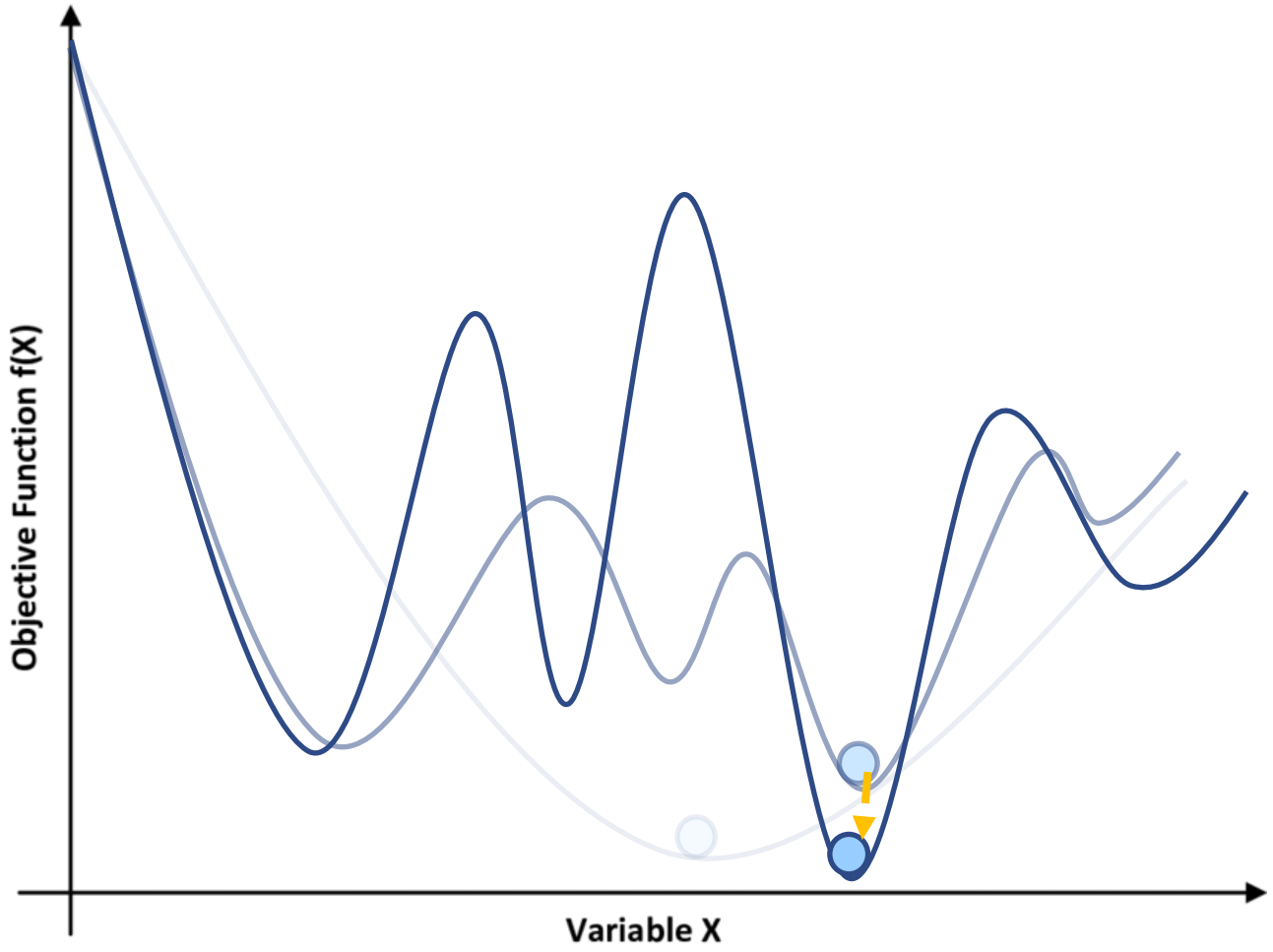
Quantum Annealing

- **Quantum Annealing** is the quantum version of simulated annealing
- The principle of quantum mechanics that is most exploited during the run of a quantum annealing is the **phenomenon of quantum tunneling**
- Visually, we can consider the quantum annealing process as a simulated annealing process where the ball, a macroscopic object, is replaced by a microscopic particle.
- How does the Quantum Annealing process work? The core of the algorithm is in the **Adiabatic Theorem**:
  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



Quantum Annealing

- Optimization through quantum annealing begins with choosing an objective function **different** than the one you want to optimize.
- The choice always falls on a **simple function, of which the global minimum is known** (for example).
- The annealing process consists in **slowly modifying** the objective function to gradually change its shape
- The process lasts until the initial objective function **becomes equivalent** to the objective function whose you really want to optimize
- **If the annealing took place slowly enough, the adiabatic theorem assures us that in all the transformation phases of the objective function the global minimum point has adapted to the shape of the function**

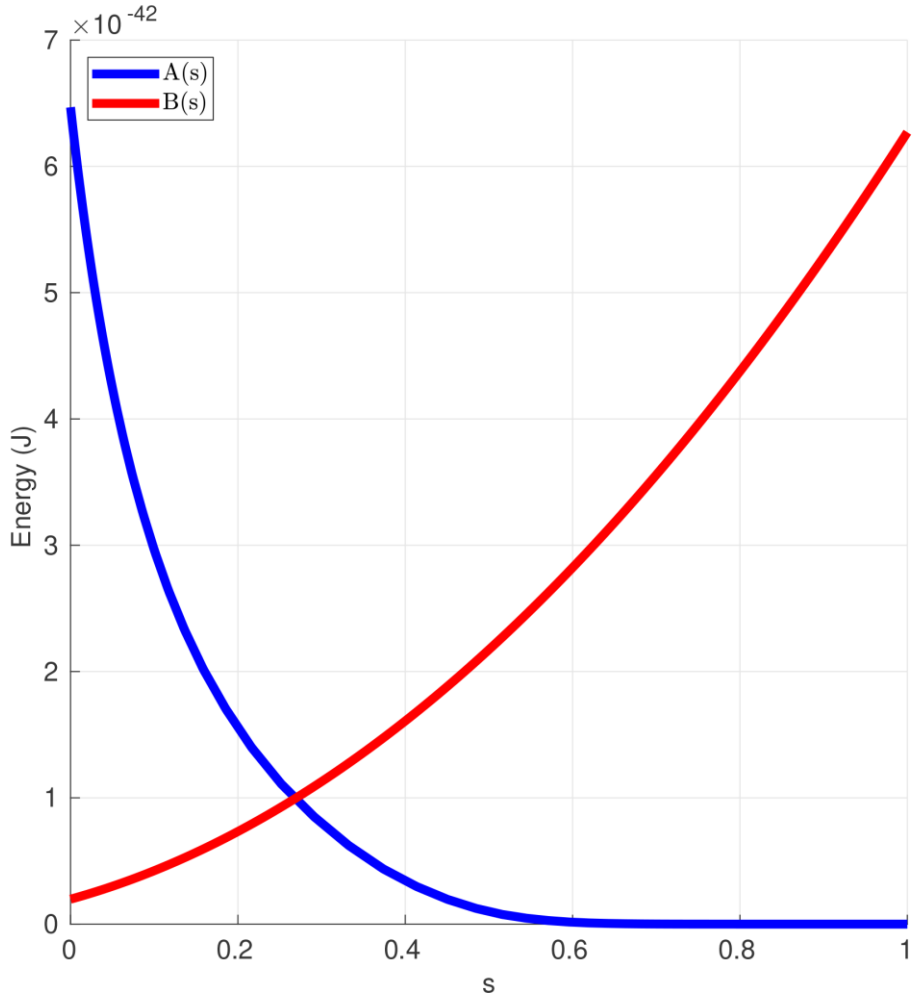


Quantum Annealing

D-wave Docs
https://docs.dwavesys.com/docs/latest/guides_solvers.html

$$\underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}}$$

- where σ are Pauli matrices operating on a qubit *q*,
- and *h* and *J* are the qubit **biases** and coupling



- BIAS: The programmable quantity that controls the external magnetic field is called a *bias*, and the qubit minimizes its energy in the presence of the bias.
- COUPLERS: link qubits together so they can influence each other. This is done with a device called a *coupler*. A coupler can make two qubits tend to end up in the same state—both 0 or both 1—or it can make them tend to be in opposite states.

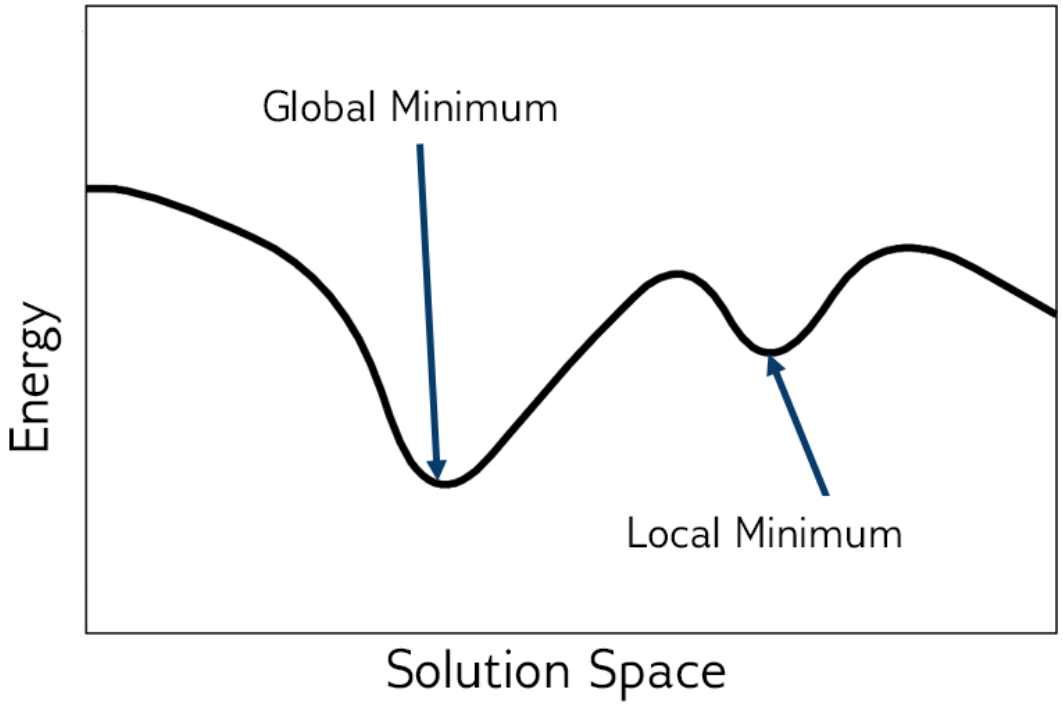
- Initial Hamiltonian (first term)—The lowest-energy state of the initial Hamiltonian is when all qubits are in a superposition state of 0 and 1. This term is also called the *tunneling Hamiltonian*.
- Final Hamiltonian (second term)—The lowest-energy state of the final Hamiltonian is the answer to the problem that you are trying to solve. The final state is a classical state, and includes the qubit biases and the couplings between qubits. This term is also called the *problem Hamiltonian*.

The Quantum Annealer

- Currently, we can talk about **The** quantum annealer and not about **a** generic quantum annealer **since today there is only one manufacturer for this type of device.**
- The company in question is called **D-Wave**
- At the moment the latest quantum annealer model has more than 5000 qubits and about 30,000 connectors
- We will see in the course of the lesson the importance of these numbers
- To understand how to interact with a quantum annealer, we need the following concepts:
  - Objective functions
  - Ising model (Ising Hamiltonian)
  - Quadratic Unconstrained Binary Optimization problems (QUBO problems)
  - Graphs and embedding

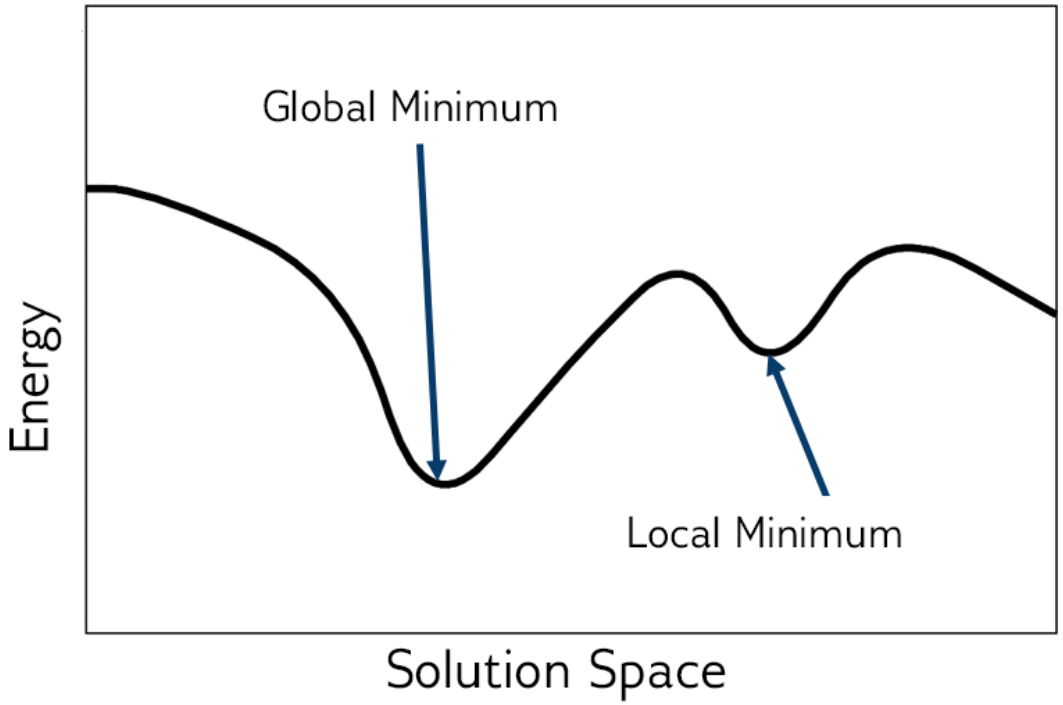
Objective Function

- To express a problem in a form that allows its resolution through quantum annealing, we need an objective function,
- An objective function is a mathematical expression of the **energy of a system. Put simply, it represents the function whose minimum you want to find**
- When the solver is a QPU, energy is a function of the binary variables that represent its qubits; for classical quantum hybrid solvers, energy might be a more abstract function.
- For most problems, the lower the energy of the objective function, the better the solution. Sometimes any state of local minimum for energy is an acceptable solution to the original problem; for other problems only optimal solutions are acceptable.



Objective Function

- Expressing a problem through a minimizable objective function means **thinking of every problem as a minimization problem**
- Mathematically speaking, this is always a possible operation
- Although, in some cases it becomes very difficult.
- The objective functions accepted by the quantum annealer of D-Wave are of two types (equivalent to each other): **Ising Hamiltonians and QUBO formulations**



$$x + 1 = 2$$

$$\min_x [2 - (x + 1)]^2$$

Ising Model

- The Ising Model is a well-known model in statistical mechanics.
- Quadratic and binary model, an Ising Hamiltonian has as variables +1 and -1 (commonly called **spin variables**: spin up for the value +1, spin down for the value -1).
- The relationships between the spins, represented by the coupling values of the Hamiltonian, represent the **correlations or anti-correlations**.
- Mathematically, it is expressed in this form

$$E_{ising}(s) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j$$

- Where the coefficients h represent the bias values associated with the qubits and the coefficients J represent the strength of the coupling bonds

QUBO Problems

- QUBO (Quadratic Unconstrained Binary Optimization) problems are well known problems in the field of combinatorial optimization.
- A QUBO problem is defined by a **matrix Q** (upper triangular) and a **vector of binary variables x**.
- Its mathematical form is

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j$$

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

$$\min_{x \in \{0,1\}^n} x^T Q x.$$

Equivalence QUBO and ISING model

• **The Ocean software** (libraries for running problems on D-Wave quantum computers) **accepts either Ising or QUBO problems**, but flags may need to be turned on to indicate whether we are interested in a "BINARY" (QUBO) or "SPIN" (Ising) solution.

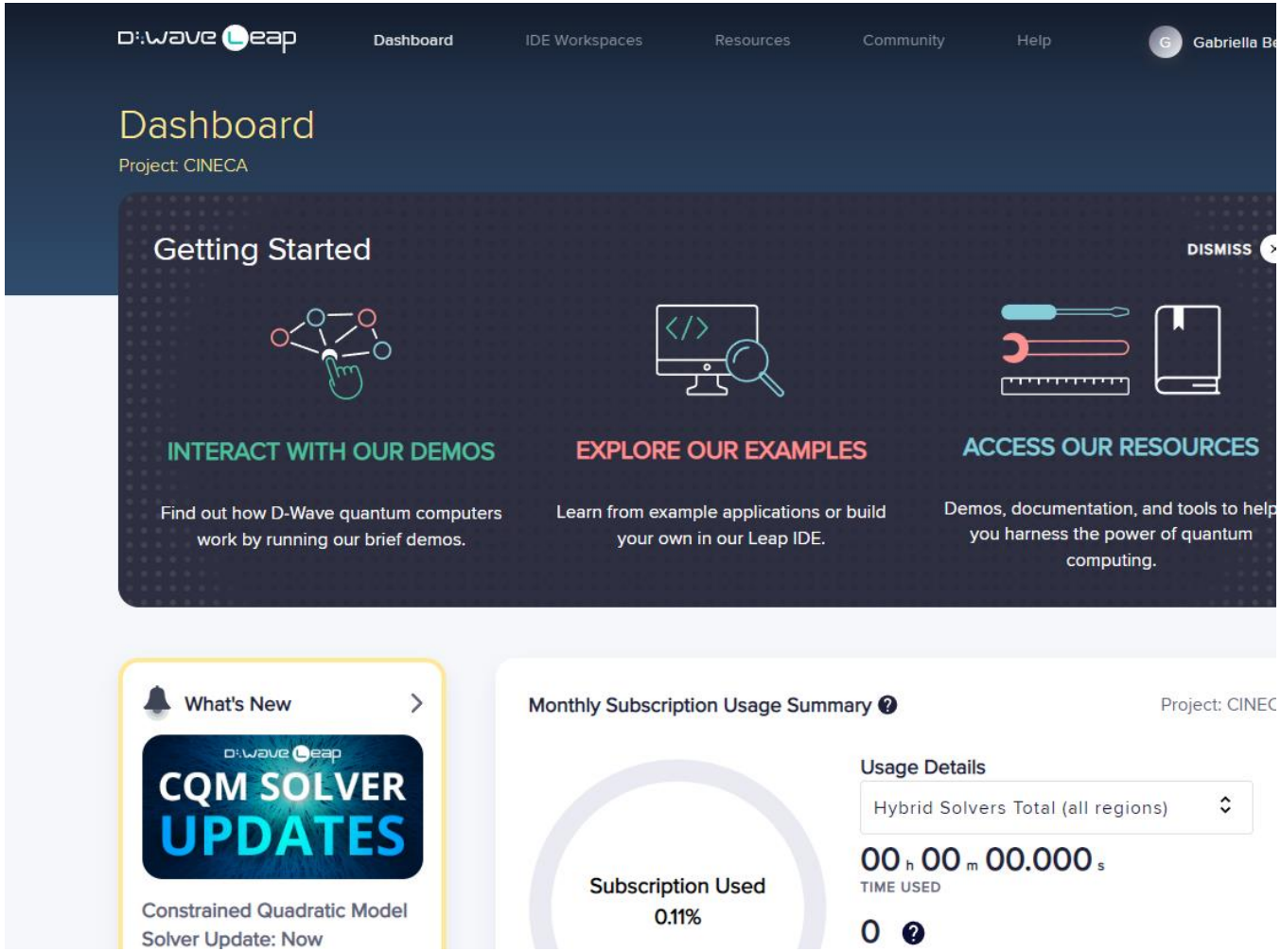
• **Ising and QUBO expressions are isomorphic.** See proof: https://support.dwavesys.com/hc/en-us/community/posts/360017439853-Difference-between-BQM-Ising-and-QUBO-problems-

WARNING: the choice of spin or binary can affect the way the problem can be expressed

→ QUBOs can always be fully expressed in both expanded and matrix forms ($x_i^2 = x_i$: $0^2 = 0$ and $1^2 = 1$)

→ while Ising can be fully expressed in the expanded form, but not completely in the matrix form. $x_1 + 2x_2 + 3x_3 + 4x_1x_2 + 5x_2x_3 + 6x_1x_3 \neq x_1^2 + 2x_2^2 + 3x_3^2 + 4x_1x_2 + 5x_2x_3 + 6x_1x_3$

D-wave Leap



https://cloud.dwavesys.com/leap/

- 1 minute for free

  BUT we can provide you more computation hours

- Many examples!
- ATTENTION: do not store here your code!

https://qcsc.dfa.unipd.it/

QUANTUM
COMPUTING
AND
SIMULATION
CENTER

How to access to more computational hours?

through our PROJECT CINECA

→ QCSC Calls

Programming a Quantum Annealer

- **INSTALLATION AND CONFIGURATION INSTRUCTIONS :**

https://docs.ocean.dwavesys.com/en/stable/overview/install.html

Create a virtual environment for your Ocean work, not mandatory but recommend!

```
python -m venv ocean
. ocean/bin/activate
```

It is necessary to have Python installed!

For a standard installation

```
pip install dwave-ocean-sdk
```

Programming a Quantum Annealer

- **INSTALLATION AND CONFIGURATION INSTRUCTIONS :**

Set Up Your Environment

```
$ dwave setup

Optionally install non-open-source packages and configure your environment.

Do you want to select non-open-source packages to install (y/n)? [y]: ↵

D-Wave Drivers
These drivers enable some automated performance-tuning features.
This package is available under the 'D-Wave EULA' license.
The terms of the license are available online: https://docs.ocean.dwavesys.com/eula
Install (y/n)? [y]: ↵
Installing: D-Wave Drivers
Successfully installed D-Wave Drivers.

D-Wave Problem Inspector
This tool visualizes problems submitted to the quantum computer and the results returned.
This package is available under the 'D-Wave EULA' license.
The terms of the license are available online: https://docs.ocean.dwavesys.com/eula
Install (y/n)? [y]: ↵
Installing: D-Wave Problem Inspector
Successfully installed D-Wave Problem Inspector.

Creating the D-Wave configuration file.
Using the simplified configuration flow.
Try 'dwave config create --full' for more options.

Creating new configuration file: /home/jane/.config/dwave/dwave.conf
Profile [defaults]: ↵
Updating existing profile: defaults
```

Programming a Quantum Annealer

- PyQUBO is a python library, with a C ++ backend, written by DWAVE to use its quantum annealer.

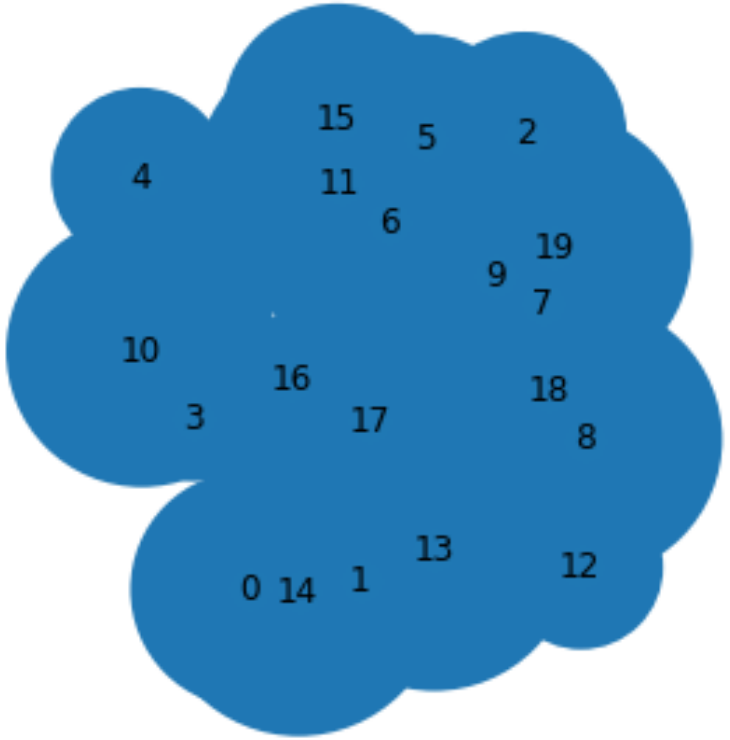
- PyQUBO is a very handy utility for writing problems in QUBO or ISING form (but it's not the only way). Let's see how to use it

- Solve a problem set via pyQUBO
- After setting the Hamiltonian of the problem, compiled and transformed into a bqm object (stands for Binary Quadratic Model and is just a general term that encompasses both Ising and QUBO problems.)

```
>>> from pyqubo import Binary
>>> x1, x2 = Binary('x1'), Binary('x2')
>>> H = (x1 + x2 - 1)**2
>>> model = H.compile()
>>> bqm = model.to_bqm()
```

Example: Antenna placement problem

- Suppose we have a certain number of antennas and a certain number of possible sites to place these antennas.
- Each antenna with its signal can cover a certain area. When multiple signals overlap, however, unpleasant interference is generated
- Our task is to position the antennas in order to maximize the surface covered by the signal and at the same time minimize interference between the antennas.



QUBO Problems

OUR MODELING (OTHER WAYS EXIST!):

- The area covered by a single antenna such as the area of the circle whose radius is the parameter that describes the range of action of each individual antenna (problem data)

$$A_i = r_i^2 \cdot \pi$$

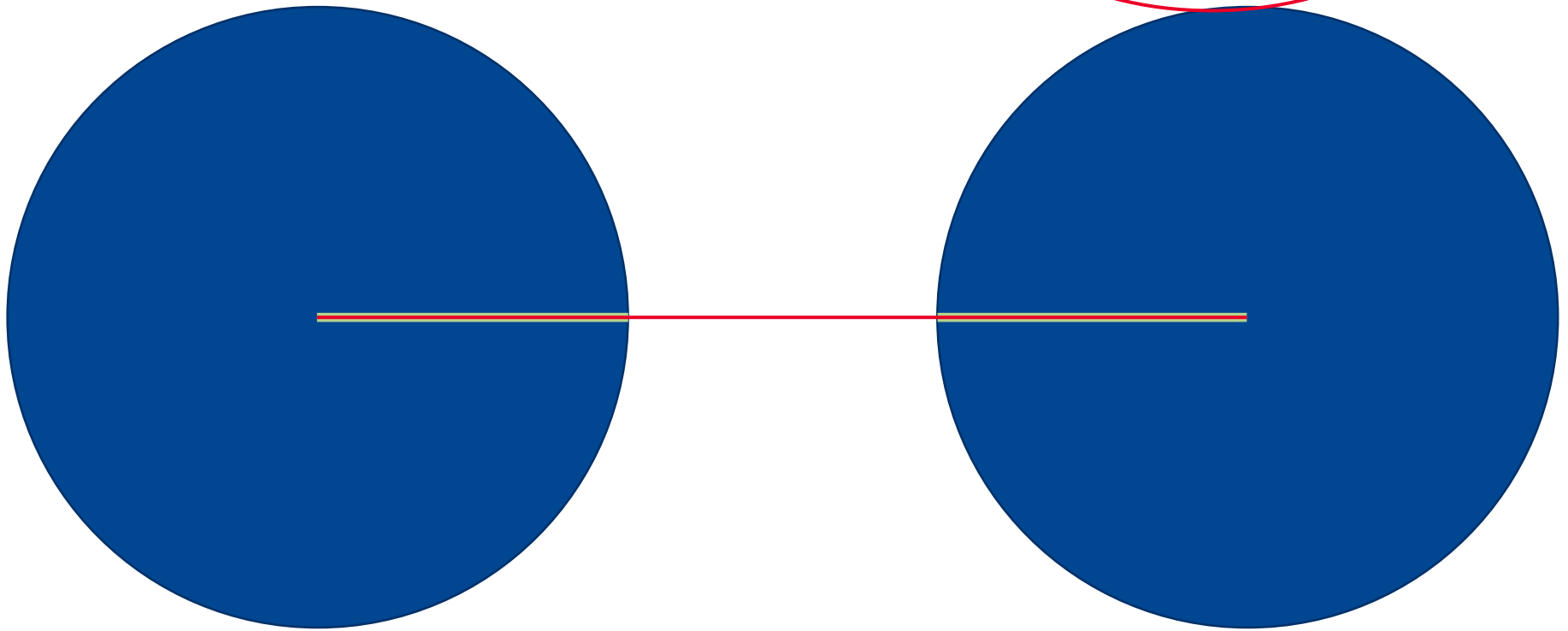
- The interference surface between two antennas as the area of the circle whose radius is given by the following formula

$$\rho_{ij} = \max\left\{0, r_i + r_j - \text{dist}\left(c_i, c_j\right)\right\}$$

- where r_i and r_j are the parameters relating to the range of action of the antennas i and j and dist(c_i, c_j) is the distance between the points where the antennas are positioned

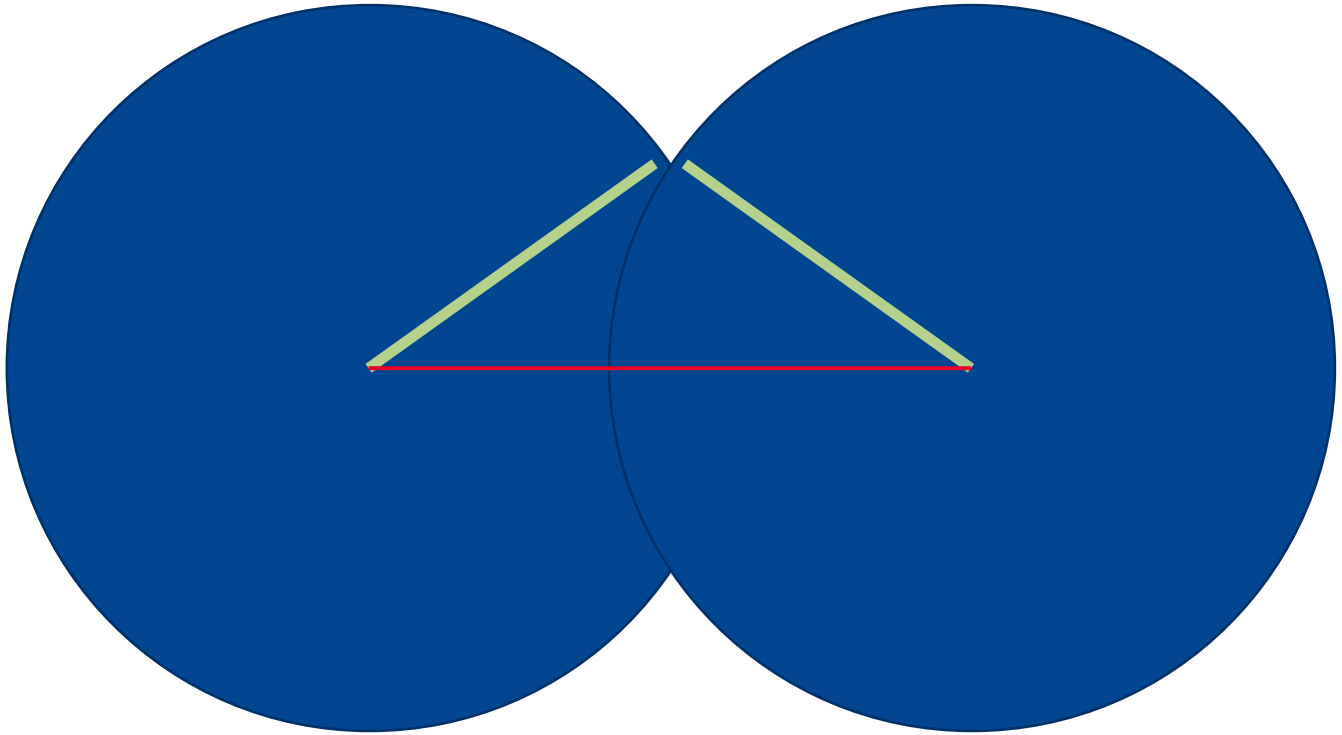
QUBO Problems

$$\rho_{ij} = \max\left\{0, r_i + r_j - \text{dist}\left(c_i, c_j\right)\right\}$$



QUBO Problems

$$\rho_{ij} = \max\left\{0, r_i + r_j - \mathrm{dist}\left(c_i, c_j\right)\right\}$$



QUBO Problems

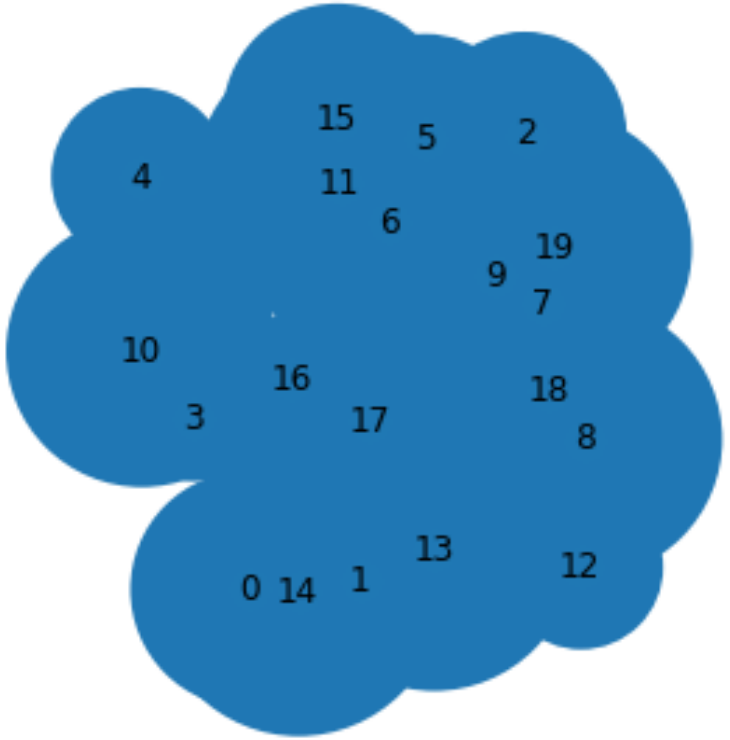
- With the definition of the rho radius, we can define the interference area between the overlap of two antennas i and j as:

$$B_{ij} = \rho_{ij}^2 \cdot \pi$$

reason: homogeneity of coefficients

- Now we model the antennas with the help of a vector of binary variables. We simply associate a binary variable q_i with each possible site. The variable will take the value 1 if it is a place where it is recommended to install an antenna, 0 otherwise

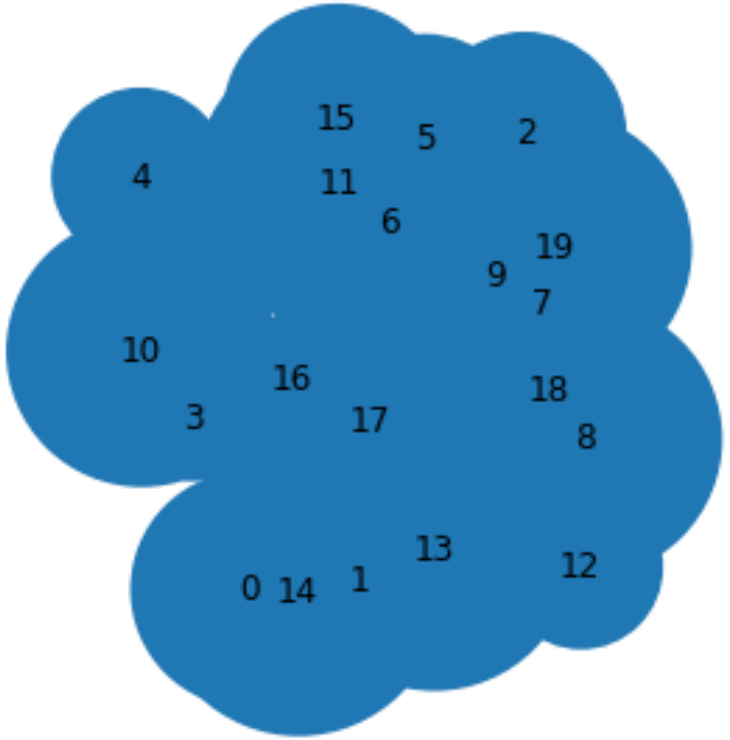
$$[q_0, \cdots, q_{19}]$$



QUBO Problems

- Let's formulate our problem. At this stage, we must always think about a minimization problem. To maximize, simply reverse the sign. Keeping in mind that

$$A_i = r_i^2 \cdot \pi \qquad B_{ij} = \rho_{ij}^2 \cdot \pi$$



- Maximize covering area and minimize the interference,
- Alpha parameter: how much interference impacts my model

(remember: we consider approximate solutions)

$$QUBO = -\sum_{i=0}^{N} A_i q_i + \alpha \sum_{i<j} B_{ij} q_i q_j$$

Add a Constraint to a QUBO Problem

- By definition, a QUBO problem admits no constraints

Quadratic
Unconstrained
Binary
Optimization

- Still, there is a way.

QUBO Problem is a general problem!

D-wave built-in functions, but building you own QUBO is more efficient

QUBO model is more generable than you could think:

- Equality constraints
- Inequality constraint → add variables
- Higher order terms→ add variables
- Non binary variables → add variables

Penalty function

An algorithm for solving constrained optimization problems. In the context of Ocean tools, penalty functions are typically employed to increase the energy level of a problem's objective function by penalizing non-valid configurations.

→ **penalty function method**

ORIGINAL_QUBO + LAMBDA * EQ.CONSTRAINT

Cons: many variables!

WARNING: problem structure changes (To be considered)!

Penalty function method

problem → combinatorial optimization problem

$$z^* = \underset{z \in S}{\operatorname{argmax}} \; f(z)$$

$$\begin{cases} g_l(z) = 0, & l = 1,\ldots,L \\ h_m(z) < 0 & m = 1,\ldots,M \end{cases}$$

Penalty function method

$$z^* = \underset{z}{\operatorname{argmax}} \; f(z) + \lambda g(z)$$

QUBO (Quadratic Unconstrained Binary Optimization)

EXAMPLE

Cost function:

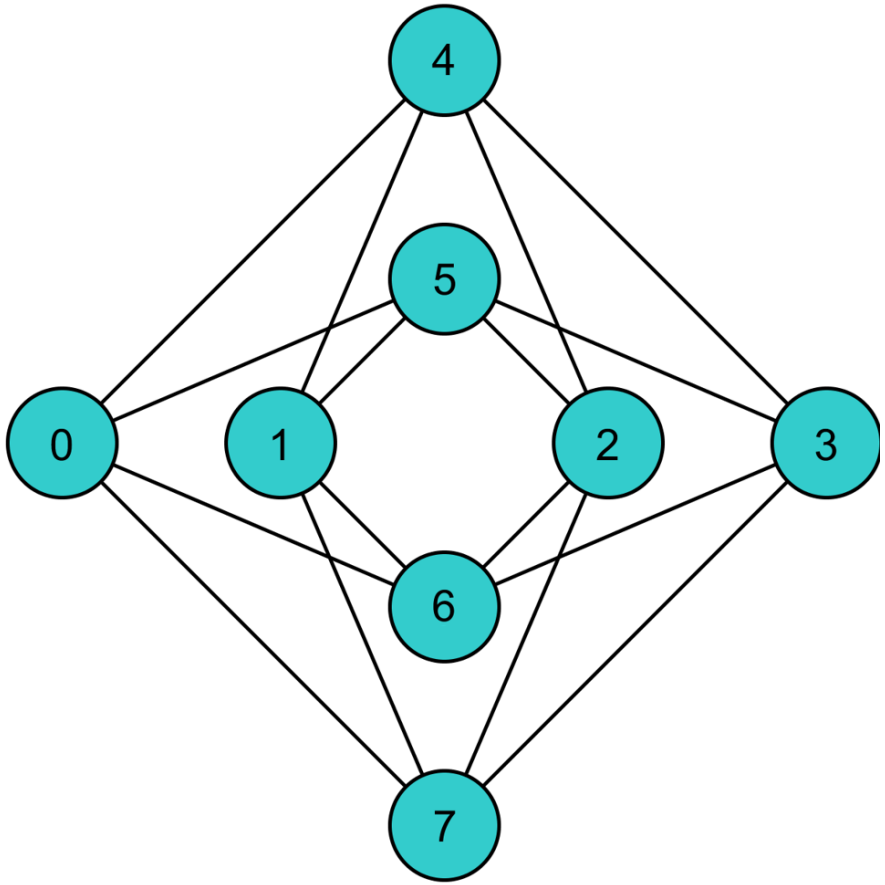
$$argmax_x \left(\sum_{I=i}^{N} c_i x_i \right)$$

Constraint: $x_3 + x_4 = 1$

QUBO:

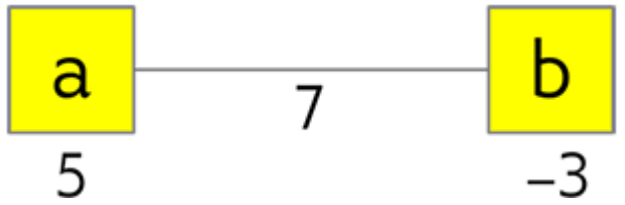
$argmax_x \left(\sum_{i=1}^{N} c_i x_i - \lambda (1 - x_3 - x_4)^2 \right) =$
$argmax_x (c_1 x_1^2 + c_2 x_2^2 + (c_3 + \lambda)\, x_3^2 + (c_4 + \lambda)\, x_4^2 - 2\lambda x_3 x_4 + (c_5 + \lambda)\, x_5^2)$

Graphs

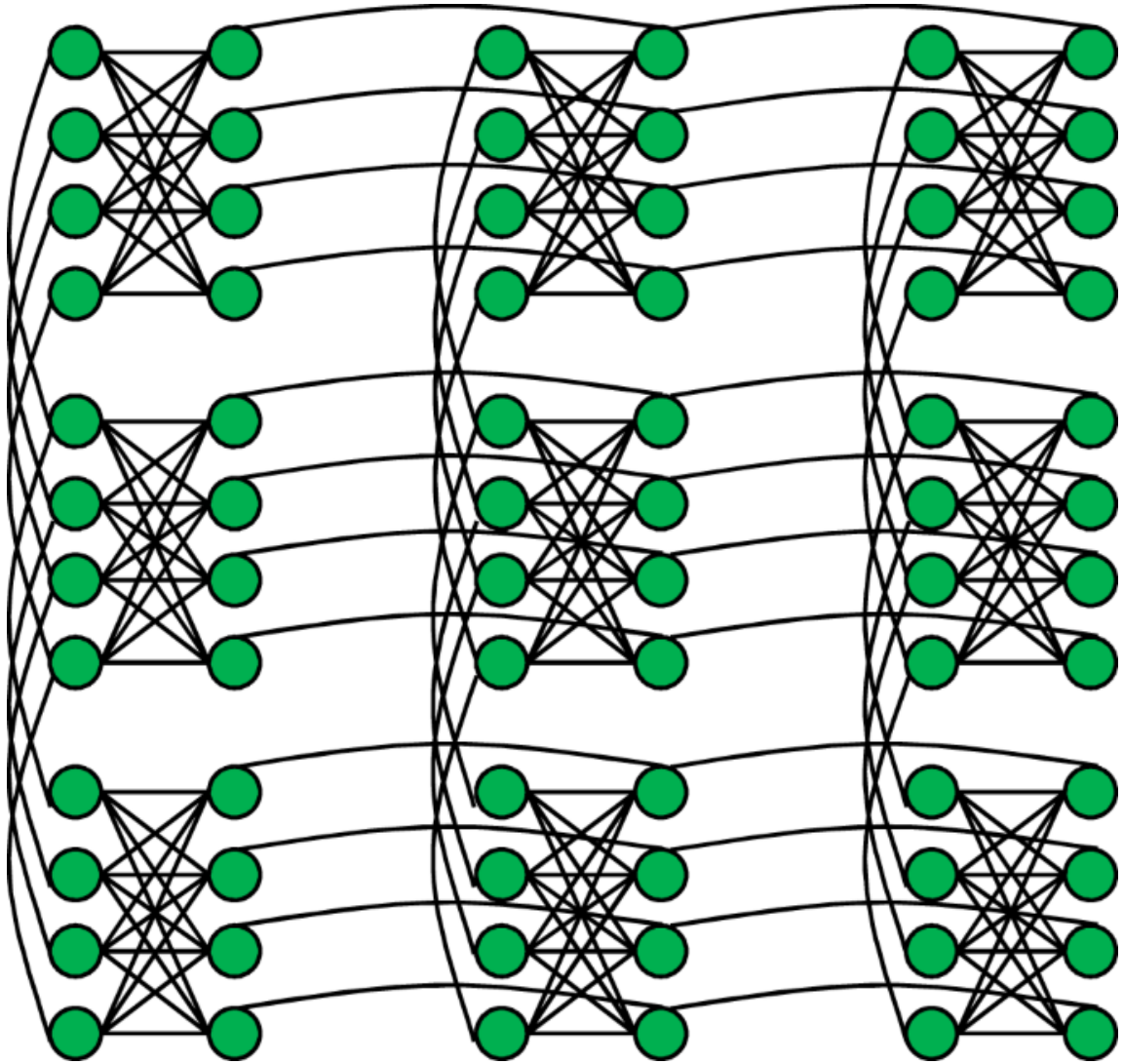


- Mathematically speaking, an undirected graph is defined as a set of vertices $V = \{v_1, \dots, v_N\}$
- and a set of edges $E \subseteq V \times V$
- Each node and each edge can be weighted with an arbitrary value (in this case we are talking about a weighted graph)
- In this way it is possible to establish a one-to-one correspondence between a weighted graph and a QUBO function

$$H(a,b) = 5a + 7ab - 3b$$



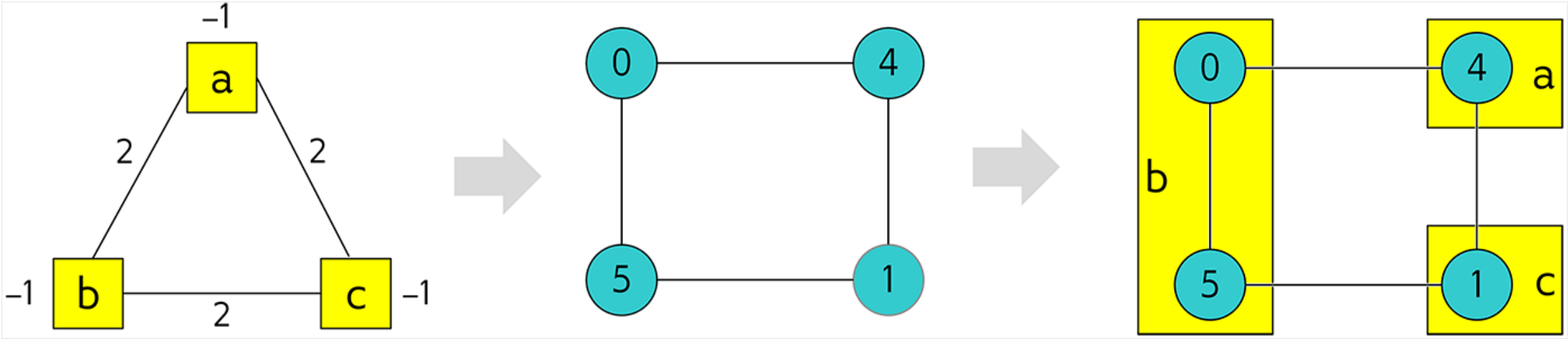
Embedding a problem on a graph



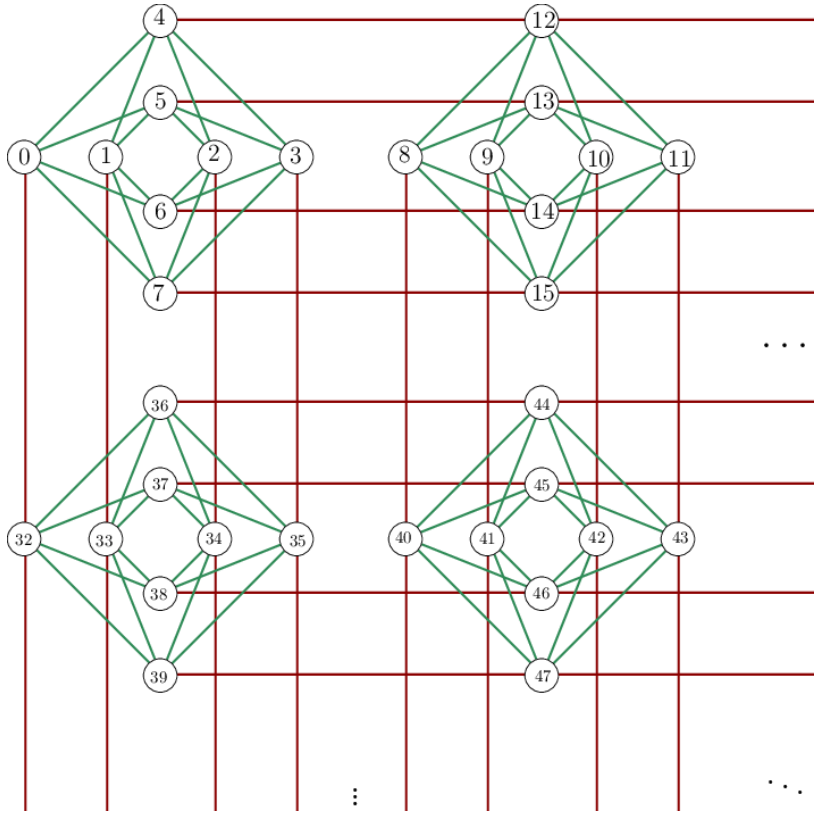
- But what if the graph with which we want to represent the QUBO function does not have enough vertices or edges to do so?
- In the case of the vertices, there is nothing to do: we have to change the problem and / or graph!
- In the case of the edges, however, something is possible to do
- The core of a quantum annealer is represented by a graph: in the figure, we can observe the Chimera graph, that is the topology of one of the D-Wave models (the penultimate model)
- This means that to solve a QUBO problem it is necessary to map your problem on the graph of the selected quantum annealer
- This procedure is called embedding

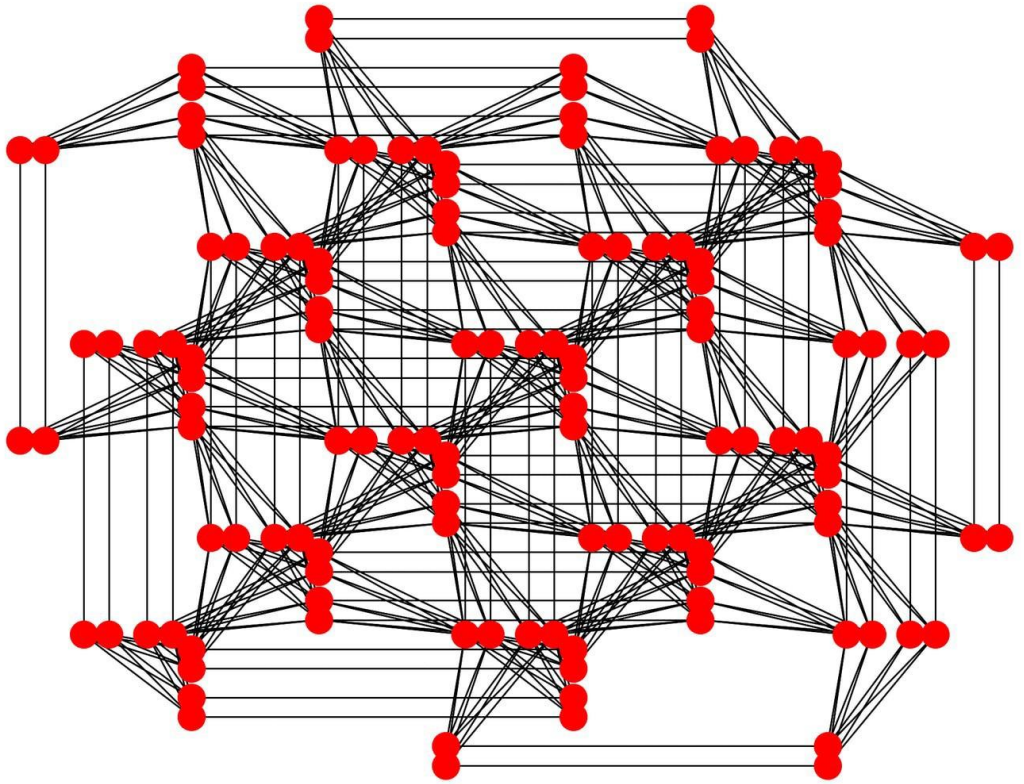
Embedding a problem on a graph

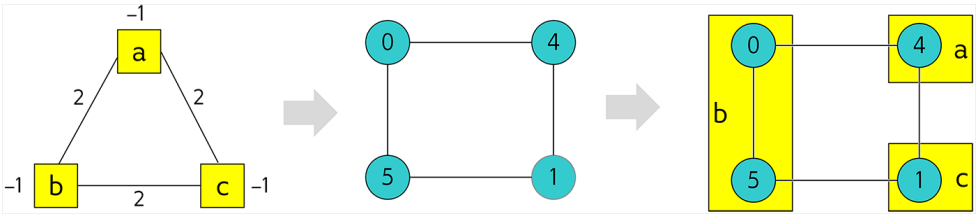
- Suppose we have a QUBO problem that can be translated with the following graph
- Suppose we also have a quantum annealer with a graph of this shape
- By looking at them, it seems impossible to map our problem to the target graph
- The embedding procedure allows for this mapping by forcing multiple qubits to behave as one
- In a certain sense, we can say that the qubits engaged in embedding are placed in entanglement relationship: they are forced to collapse in the same classical state



Embedding on Chimera and Pegasus





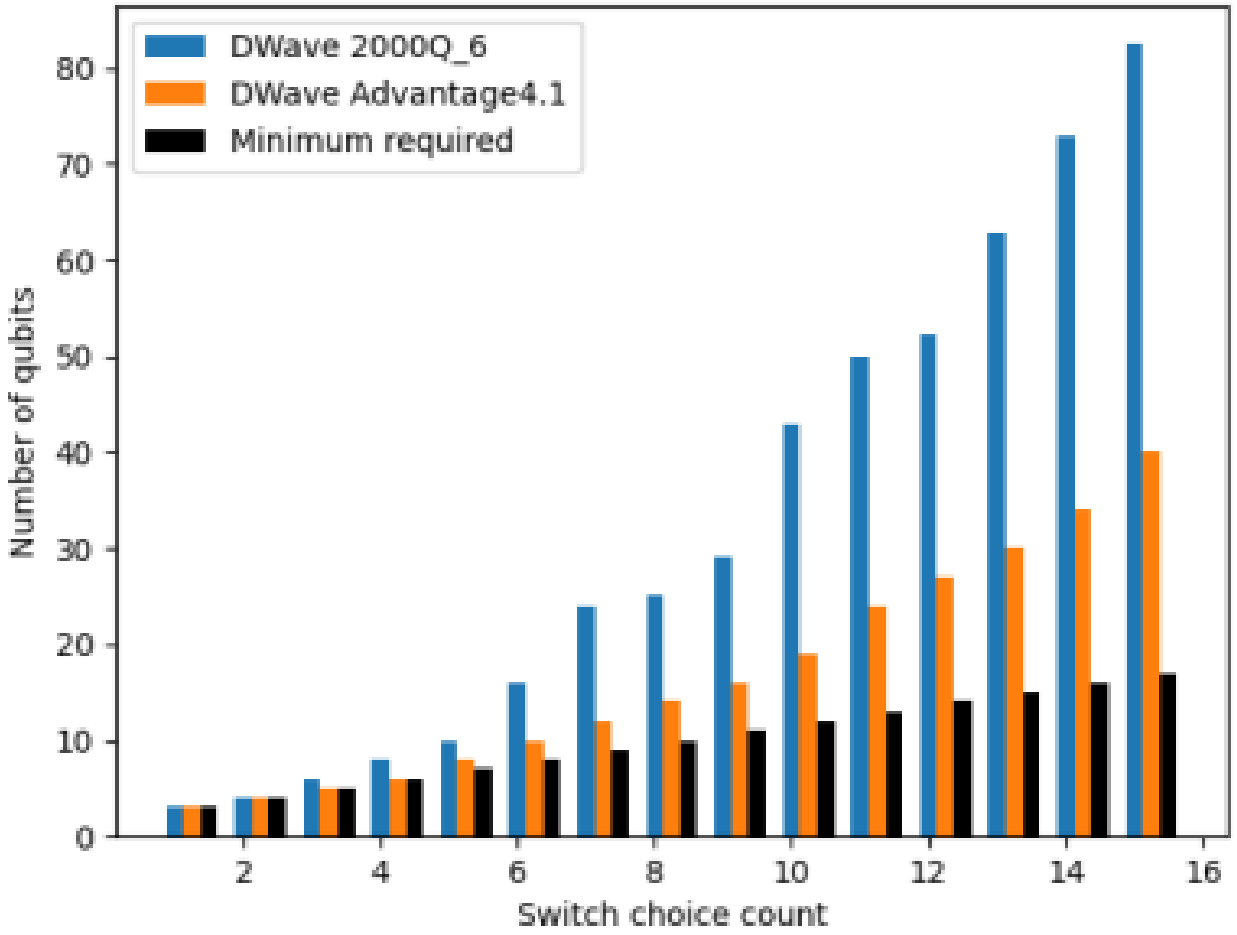


CINECA QUANTUM COMPUTING LAB

Embedding on Chimera and Pegasus

Minor-embedding

The nodes and edges on the graph that represents an objective function translate to the qubits and couplers in Chimera. Each logical qubit, in the graph of the objective function, may be represented by one or more physical qubits. The process of mapping the logical qubits to physical qubits is known as minor embedding.



Antenna Placement: Jupyter Notebook

jupyter AntennaPlacement (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [18]:
```
import numpy as np
from pyqubo import Binary
from pyqubo import Array
import neal
import networkx as nx
import matplotlib.pyplot as plt
import math
import random
from collections import Counter
```

In [19]:
```
#Distance function
def dist(pos,i,j):
    d = (pos[i][0]-pos[j][0])**2+(pos[i][1]-pos[j][1])**2
    return math.sqrt(d)
```

In [20]:
```
#let us suppose to have 20 antennas to place
#At the beginning they are at random positions

N=20

GR=nx.Graph();
for i in range(N):
    GR.add_node(i)
pos=nx.random_layout(GR)
nx.draw(GR,pos=pos,with_labels = True)
```

Let's take a look at the code!

INTRODUCTION TO D-wave COMPUTERS

Gabriella Bettonte
Mail: g.bettonte@cineca.it

Website: https://www.quantumcomputinglab.cineca.it/

Thanks to Daniele Ottaviani for the slides

CINECA QUANTUM COMPUTING LAB