Taller Quantum Annealing & Quantum Inspired Computing

Primera sesión

Agenda

Quantum Annealing

Computación Clásica vs. Cuántica

FUJITSU

Computación Clásica

Basada en transistores, que pueden tener los valores 0 o 1

La potencia se incrementa en relación 1:1 con el número de transistores

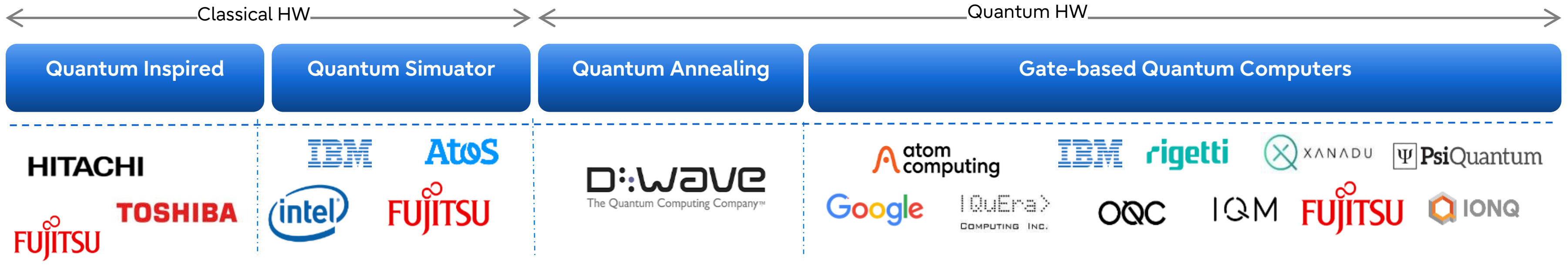
Menores ratios de error y operan a temperatura ambiente

Computación Cuántica

Basada en qubits, que pueden tener los valores 0 y 1 al mismo tiempo

La potencia se incrementa exponencialmente en proporción al número de qubits

Tienen mayores ratios de error y necesitan condiciones extremas (temperatura y/o vacío)



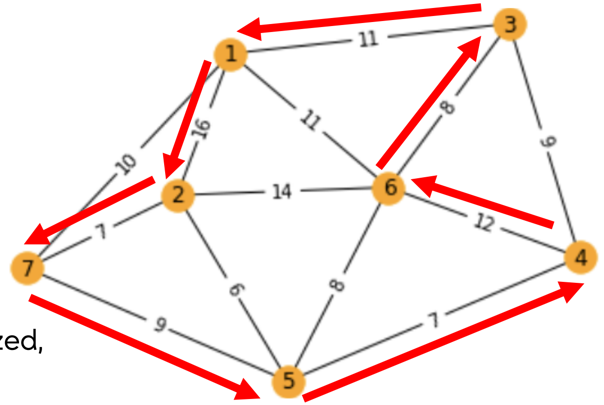
Quantum Annealing

What kind of problems can be solved?

Optimization problems: Problems where we are looking for a unique optimal solution, that matches the ground state of the Hamiltonian

Sampling problems: Problems where we are looking for several low energy states, either because several good enough solutions are available (we may not need the best solution out there), or when we intend to reconstruct the energy landscape to build a model of reality

Combinatorial Optimization Problems



A general combinatorial optimization problem consists of:

- Set of Binary Variables
- Target Function
- Set of constraints

The goal is to determine the best assignment of variables so that the target function is optimized, while satisfying all constraints.

If each pair of cities are connected, # of possible travelling paths:
=42! ~10^{51} for 43 cities (# atoms in the earth ~10^{50})
=59! ~10^{80} for 60 cities (# atoms in the universe ~10^{80})

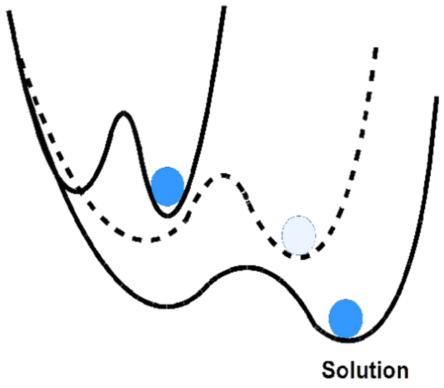
Quantum Annealing

1. The solution of a problem is **encoded into** the ground state of a time-dependent **quantum Hamiltonian**.

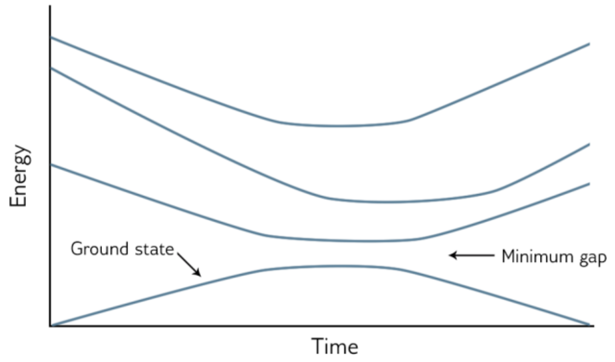
2. Physical devices called Quantum Annealers are used in which we start from an initial Hamiltonian with a well-known ground state.

3. The Hamiltonian evolves over a limited time into a new Hamiltonian in which the problem of interest is encoded.

4. The **solution** of the problem **will be encoded in the ground state of the Hamiltonian of interest.**



Adiabatic evolution



- ✓ The adiabatic theorem says that **if we evolve slow enough** the system from one Hamiltonian to another **the system will stay** always **in the ground state**.

- ✓ The "speed" should be inversely proportional to the difference between the distance between the minimum gap.

Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Quantum Annealing

For a quantum system, a Hamiltonian is a function that maps certain states, called eigenstates, to energies. The collection of eigenstates with defined eigen energies make up the eigen spectrum.

For a quantum annealer, the Hamiltonian may be represented as

$$F(v) = \underbrace{\sum_i a_i v_i}_{\text{Linear Term}} + \underbrace{\sum_i \sum_{j>i} b_{i,j} v_i v_j}_{\text{Quadratic Term}} + c$$

$$\downarrow$$

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

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on a qubit q_i, and h_i and $J_{i,j}$ are the qubit biases and coupling strengths.[1]

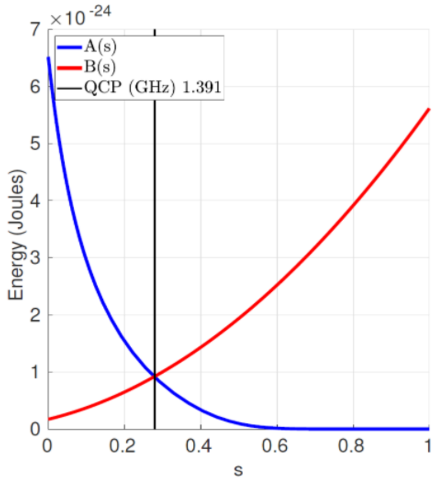


Fig. 10 Annealing functions $A(s)$, $B(s)$. Annealing begins at $s = 0$ with $A(s) \gg B(s)$ and ends at $s = 1$ with $A(s) \ll B(s)$. Data shown are representative of D-Wave 2X systems.

Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

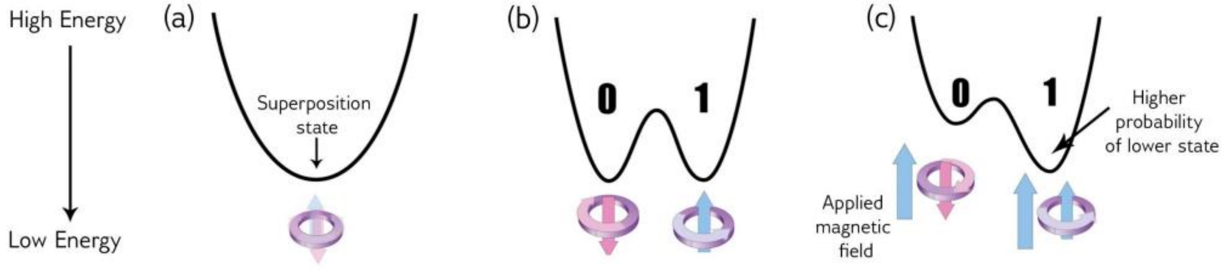
Quantum Annealing

The physics of this process can be visualized with an energy diagram as depicted below. This diagram changes over time, through the following steps:

1) We start with a single valley in the energy landscape, that represents our initial superposition state.

2) The quantum annealing process starts, the potential barrier is raised, and this turns the energy diagram into what is known as a double-well potential. Here, the low point of the left valley corresponds to the 0 state, and the low point of the right valley corresponds to the 1 state. The qubit will end up in one of these valleys at the end of the annealing process.

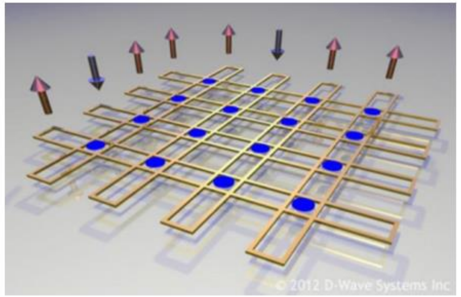
3) Everything else being equal, the probability of the qubit ending in the 0 or the 1 state would be equal. You can, however, control the probability of it falling into the 0 or the 1 state by applying an external magnetic field to the qubit. This field tilts the double-well potential, increasing the probability of the qubit ending up in the lower well. 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.

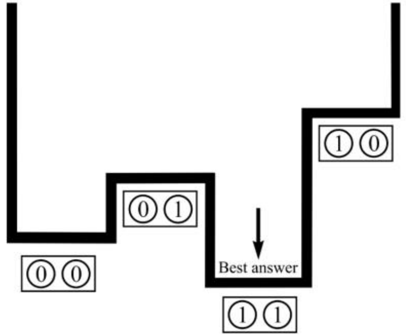


Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Quantum Annealer

- ✓ The bias term alone would only allow users to play with linear terms, without qubits being able to influence each other.
- ✓ To achieve this, we use 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. Like a qubit bias, the correlation weights between coupled qubits can be programmed by setting a coupling strength.
- ✓ When you use a coupler, you are using another phenomenon of quantum physics called entanglement. When two qubits are entangled, they can be thought of as a single object with four possible states. In that scenario, the energy landscape would show four states, each corresponding to a different combination of the two qubits: (0,0), (0,1), (1,1), and (1,0). The relative energy of each state depends on the biases of qubits and the coupling between them.





Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Digital Annealer

Quantum Computing

“The thing driving the hype is the realization that quantum computing is actually real.

It is no longer a physicist’s dream — it is an engineer’s nightmare”

Isaac Chuang
MIT Professor of Physics and Engineering

Quantum Computing

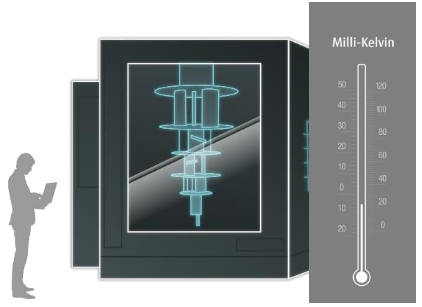
Technical Difficulties of Quantum Computing

Stability

Costs & Infrastructure

Accuracy

Low Connectivity



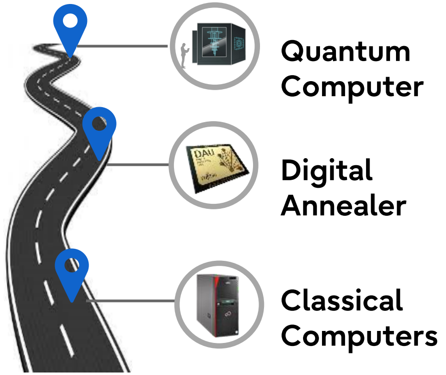
Digital Annealer: The First Step Towards Quantum Computing

What is Digital Annealer?

Digital Annealer is a solution (hardware + software) inspired by Quantum Computing, which allows you to solve combinatorial optimization problems in a very efficient way.

Why Digital Annealer?

- Enables the development of quantum applications.
- Hardware-Mature Technology.



Quantum Inspired: Simulated Annealing

Description

Simulated Annealing (**SA**) Algorithm is a general-purpose metaheuristic algorithm to solve NP-Hard optimization problems.

To understand SA, one needs to understand the core idea behind the algorithm and the parameters involved in the algorithm, which play a vital role in obtaining an optimal/best solution for an **NP-hard problem**.

Simulated Annealing is conceptually based on metallurgical annealing where a crystalline solid is heated and then allowed to cool very slowly until it achieves its most stable lattice energy state. **If the cooling schedule is sufficiently slow, the final configuration results in a solid with best structural stability**.

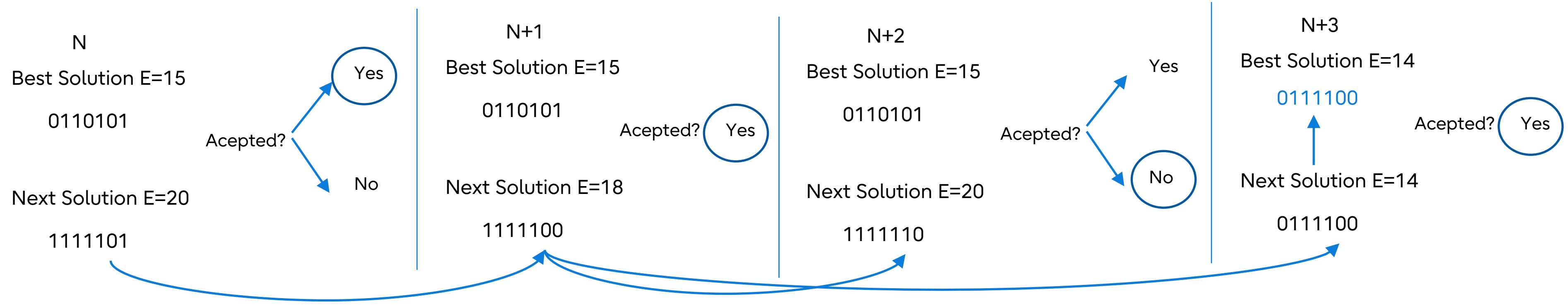
Quantum Inspired: Simulated Annealing

FUJITSU

Step by Step

1 At each iteration of a SA, the algorithm retains the current solution candidate, and generates a new solution.

2 The new solution is obtained by selecting a local neighbour of the current solution. If the new solution gives a lower energy it is accepted. However, if the new solution yields a higher energy, it is not discarded, instead, it is accepted with a certain probability.



The key algorithmic feature of SA is that **it provides a means to escape local optima** by allowing hill-climbing moves (i.e., moves which worsen the objective function value)

Quantum Inspired: Simulated Annealing

Aceptance Probability

The SA starts with some initial configuration $\mathbf{x}$ and computes the current energy as $f(\mathbf{x})$. The configuration $\mathbf{x}$ then goes through a perturbation, where certain bits in the vector $\mathbf{x}$ are chosen and flipped from 0 to 1 (and vice-versa). The perturbation step allows us to generate a new solution, say $\mathbf{x}'$, with the energy $f(\mathbf{x}')$. The next step involves the decision to accept this new configuration for the next step or to discard it.

If the new configuration offers a better solution (*i.e.* a lower energy) it is accepted as the current solution and the state $\mathbf{x}'$ is taken as the new $\mathbf{x}$. However, if $f(\mathbf{x}') > f(\mathbf{x})$, the new configuration goes through a Metropolis acceptance criterion. With this criterion the new higher energy solution $\mathbf{x}'$ is accepted with a certain probability $P(\mathbf{x}, \mathbf{x}')$ which depends on the current temperature T_i, such that

$$P(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & f(\mathbf{x}') \leq f(\mathbf{x}) \; , \\ \exp\left(\frac{-(f(\mathbf{x}') - f(\mathbf{x}))}{T_i}\right) & f(\mathbf{x}') > f(\mathbf{x}) \; . \end{cases}$$

Quantum Inspired: Simulated Annealing

Acceptance Probability

This acceptance probability is the basic element of the search mechanism in SA.

This so called 'Metropolis Acceptance Criterion' helps the algorithm to avoid falling into a local minimum. The figure shows the acceptance of a bad solution (during hill climbing) and the overall convergence of SA.

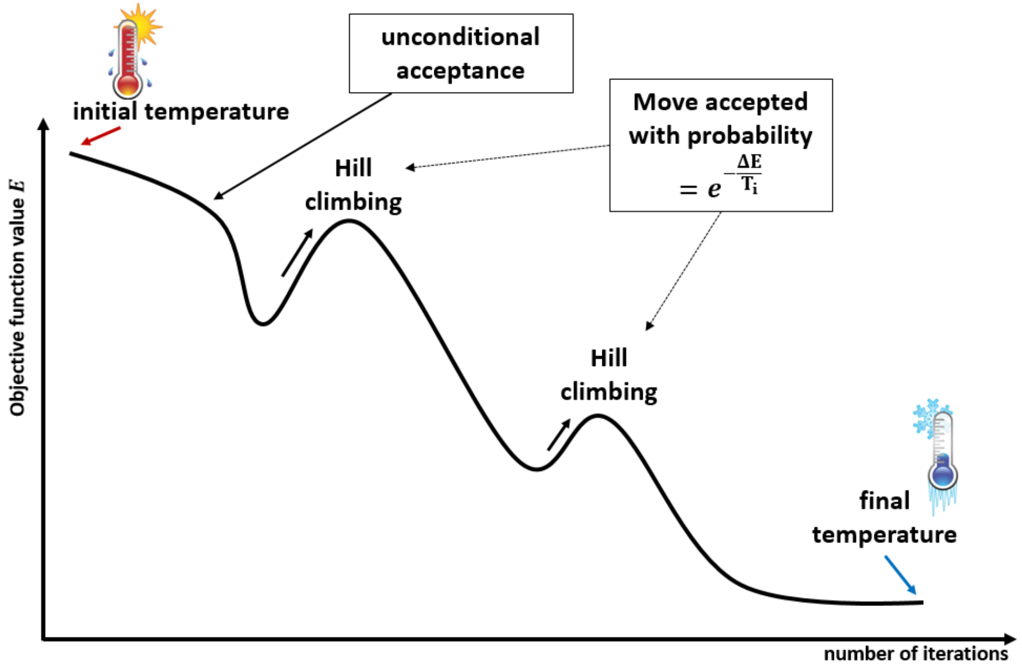


Figure depicting convergence of Simulated Annealing algorithm as the iterations increase and the temperature is reduced

Quantum Inspired: Simulated Annealing

Annealing Temperatures

Appropriate annealing temperatures are of vital importance to obtaining an optimal/good solution through simulated Annealing. To understand the role of temperature in SA, we study the acceptance criterion once again, given by equation below.

$$P(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & f(\mathbf{x}') \leq f(\mathbf{x}) \, , \\ \exp\left(\frac{-\left(f(\mathbf{x}') - f(\mathbf{x})\right)}{T_i}\right) & f(\mathbf{x}') > f(\mathbf{x}) \, . \end{cases}$$

Case 1: Temperature too high ($T_i \longrightarrow \infty$)

Case 2: Temperature too low ($T_i \longrightarrow 0$)

PUBLIC

Quantum Inspired: Simulated Annealing

Stopping Criteria

Stopping criteria is important to decide when to stop the annealing process.

- One of the standard ways is to define a maximum number of iterations that must take place in the annealing process
- Another stopping criterion is the stalling energy. One may record some number of previous iterations' energies, and the algorithm may be halted if the energies are stalled.

In the DADK, the stopping criterion is the number of iterations. In this criterion, the annealing process explained above is executed for the given number of iterations.

Quantum Inspired: Simulated Annealing

Detailed process

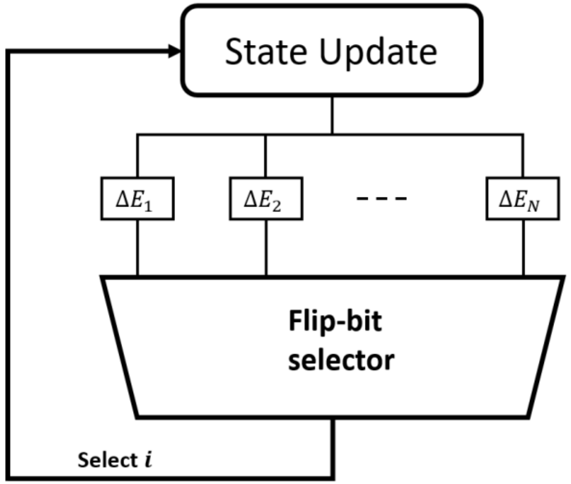
We now present a short overview of the implementation of the above-mentioned annealing process and its **parallelization**.

1 **Trial Phase**:

- a single bit is chosen from the current state, and the energy change for flipping the bit is computed.
- This energy change is used to determine whether to accept the bit flip in accordance with the Metropolis-Hastings criterion.
- The Metropolis acceptance of all possible bit flips up to 8192 can be calculated in parallel.

2 In the **update phase**, the flip-bit selector selects one bit to be flipped and updates the value of that bit In the DA

DA offers 16 independent trial-update processes on different base states so that 16 independent annealing processes can be executed in parallel.

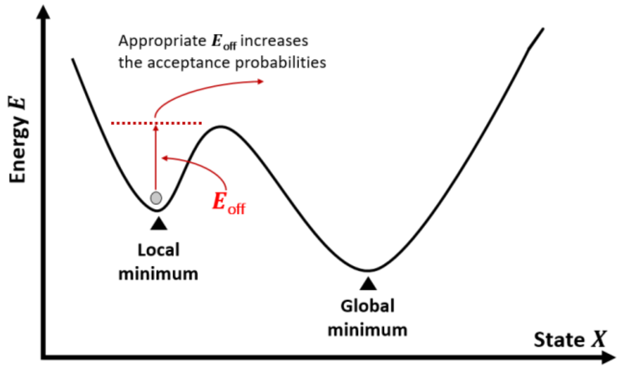


A schematic diagram of the parallel search technique.

Quantum Inspired: Simulated Annealing

Offset

Imagine your random walk is in a local minimum and the temperature is so low that the probability of acceptance << 1. With such a low acceptance probability the annealing process is very unlikely to make a step.



Appropriate E_{off} increases the acceptance probabilities
Energy E
E_{off}
Local minimum
Global minimum
State X

A schematic diagram of the escape technique using offest energy.

As one can imagine, the offset value must not be too low, as this would not allow the solution to come out of local minimum in few steps. Additionally, a very high offset value might totally lose the result of the annealing process, as a very high offset energy would allow for any new solution to be accepted regardless of how worse the new solution is.

QUBOs and ISING Models

QUBOS and Ising Models

QUBO (Quadratic Unconstraint Binary Optimization Problem) is a polynomial of order two at most in which variables are binary. We want to find the configuration that minimize the polynomial.

General Formulation

$$\underbrace{\sum_i a_i v_i}_{\text{Linear Term}} + \sum_i \underbrace{\sum_{j>i} b_{i,j} v_i v_j}_{\text{Quadratic Term}} + c$$

Examples

$$x_1^2 + x_2 + 7 = 0$$

$$x_2 x_3 + 7x_1 = 0$$

$$x_3 + 7x_1 = 0$$

Where:

- ✓ The constants a_i and $b_{i,j}$ depends on the definition of our problem
- ✓ The variables v_i are the values we are looking for. The values for each v_i that minimize the overall expression, are the solution of our problem
- ✓ As we are just looking for the global minimum of the expression, the value of the constant c is usually ignored because It just shifts the entire mathematical function up or down, increasing or decreasing the final value by a constant amount, so the location of the global minimum will not be changed if we ignore this constant

QUBOS and Ising Models

- ✓ **QUBOs and Ising Models are almost the same thing**. Their main difference is that, for each one of the binary values that take part in the QUBO, the binary value can be either 1 or -1 in the case of Ising models (mostly used by physicists), or 0 or 1 in the case of QUBOs (mostly used by computer scientists and mathematicians)

- ✓ We will use QUBOs and Ising models as our objective functions

- ✓ QUBOs can also be expressed as an upper diagonal matrix Q of size NxN (being N the number of variables), where the diagonal terms a_i = $a_{i,i}$= $Q_{i,i}$ refers to the linear coefficients, and the nonzero off diagonal terms $b_{i,j}$ = $Q_{i,j}$ are the quadratic coefficients

- ✓ We can transform the previous expression into a more concise form $x^T Qx$

Where:

- Q refers to the matrix previously described
- x is a column vector of size N that contains the variables xiwhose values we want to get
- xT is just its transposed version to a row vector

QUBO vs Ising Models

Ising and QUBO models are isomorphic. We can prove it by considering the expression for an Ising model, and converting it to a QUBO model by transforming the original variables $si \in -1,1$ into $xi \in 0,1$ arriving at the expression for a QUBO model

Ising

$$
\begin{aligned}
&min\left(\sum_i h_i s_i + \sum_i \sum_{j>i} J_{i,j} s_i s_j\right) \overset{[1]}{=} \\
&= min\left(\sum_i h_i(2x_i - 1) + \sum_i \sum_{j>i} J_{i,j}(2x_i - 1)(2x_j - 1)\right) = \\
&= min\left(\sum_i (2h_i x_i - h_i) + \sum_i \sum_{j>i} J_{i,j}(4x_i x_j - 2x_i - 2x_j + 1)\right) = \\
&= min\left(\sum_i 2h_i x_i - \sum_i h_i + \sum_i \sum_{j>i} 4J_{i,j} x_i x_j - \sum_i \sum_{j>i} 2J_{i,j} x_i - \sum_i \sum_{j>i} 2J_{i,j} x_j + \sum_i \sum_{j>i} J_{i,j}\right) = \\
&= min\left(\sum_i 2h_i x_i - \sum_i h_i + \sum_i \sum_{j>i} 4J_{i,j} x_i x_j - \sum_i x_i \sum_{j>i} 2J_{i,j} - \sum_i \sum_{j>i} 2J_{i,j} x_j + \sum_i \sum_{j>i} J_{i,j}\right) = \\
&= min\left(\sum_i \left(2h_i - 2 \cdot \sum_{j>i} J_{i,j}\right) x_i + \sum_i \sum_{j>i} (4J_{i,j}) x_i x_j + \left(\sum_i \sum_{j>i} J_{i,j} - \sum_i h_i\right) - 2 \cdot \sum_i \sum_{j>i} J_{i,j} x_j\right) \overset{[2]}{=} \\
&= min\left(\sum_i \left(2h_i - 2 \cdot \sum_{(j<i)} J_{j,i} - 2 \cdot \sum_{j>i} J_{i,j}\right) x_i + \sum_i \sum_{j>i} (4J_{i,j}) x_i x_j + \left(\sum_i \sum_{j>i} J_{i,j} - \sum_i h_i\right)\right) = \\
&= min\left(\sum_i a_i x_i + \sum_i \sum_{j>i} b_i x_i x_j + c\right)
\end{aligned}
$$

QUBO

DADK Introduction

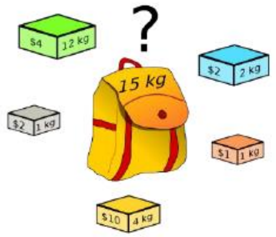
Let's code!

Knapsack Problem

Knapsack Problems

Definition

Knapsack problems are a particular type of problems within the field of **combinatorial optimization.** Given a set of items, each with a specific weight and profit we must choose which ones should be taken in a knapsack of a particular capacity without exceeding it, and while maximizing the total profit of the items taken.



- ✓ The **weight**, any metric that limits the maximum items we can carry or pick. We may be considering as weight the available budget that we can spend, or the total volume or actual physical weight of the items that we can carry, or the time required for a task if we are considering a scheduling problem. The weight takes a specific value for each item and does not vary depending on which other items we pick.
- ✓ The **profit**, any metric that represents the benefit associated to a certain item. We could be referring to the expected return of an investment, the number of tasks completed, or any other metric that helps us decide which items should be dropped if we cannot take all items in our knapsack.

Uses Cases

Portfolio Optimization

The maximum weight is the available budget, and the expected revenue for each investment will represent its profit.

Project Priorization/Selection

The maximum weight will be the available time and profit will be the revenue of each project.

Source: https://en.wikipedia.org/wiki/Knapsack_problem

Knapsack Problems

Knapsack problems take different mathematical forms depending on the details of each problem, but the most basic form, the 0 1 knapsack problem looks as follows:

$$\mathit{Maximize} \sum_{j=0}^{n} p_j \cdot x_j, \qquad \mathit{subject\ to} \sum_{j=0}^{n} w_j x_j = W$$

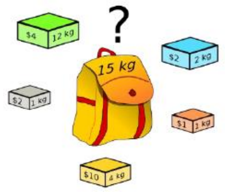
Where:

$x_j \in \{0,1\}$ represents whether the item j is added to the knapsack or not.

$p_j \in \mathbb{R}$ is the profit associated to item x_j. It is only added to the final profit if $x_j = 1$.

$w_j \in \mathbb{R}$ is the weight associated to item x_j.

$W \in \mathbb{R}$ is the maximum weight allowed by our problem. In some literature, it is referred to as c.



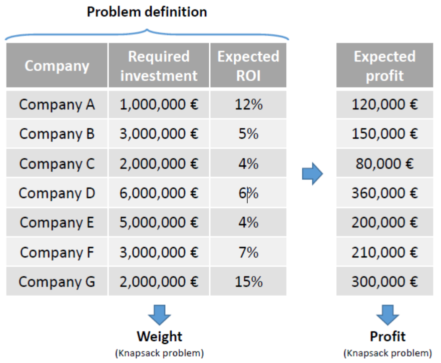
We will shape this type of problems with the following Hamiltonian, adding a constant α, which is called penalty, or Lagrange multiplier. It helps us balance the different parts of the model.

$$H = \overset{\substack{\text{Maximization}\\ \text{to minimization}}}{-} \underbrace{\sum_{j=0}^{n} p_j x_j}_{\text{Profit}} + \underset{\substack{\text{Lagrange}\\ \text{multiplier}}}{\alpha} \cdot \left(\overset{\substack{\text{Maximum}\\ \text{weight}}}{W} - \underbrace{\sum_{j=0}^{n} w_j x_j}_{\text{Weight}} \right)^2$$

Source: https://en.wikipedia.org/wiki/Knapsack_problem

Knapsack Problems

Let us consider an example of portfolio optimization, let's assume that we have a budget of 10 000 000 EUR that we would like to invest in $\boldsymbol{n}$ different companies, based on the expected ROI (Return On Investment) or profit for each investment opportunity.



Problem definition

Company	Required investment	Expected ROI
Company A	1,000,000 €	12%
Company B	3,000,000 €	5%
Company C	2,000,000 €	4%
Company D	6,000,000 €	6%
Company E	5,000,000 €	4%
Company F	3,000,000 €	7%
Company G	2,000,000 €	15%

Expected profit
120,000 €
150,000 €
80,000 €
360,000 €
200,000 €
210,000 €
300,000 €

Weight
(Knapsack problem)
Profit
(Knapsack problem)

j	w_j	p_j
0	1	12
1	3	15
2	2	8
3	6	36
4	5	20
5	3	21
6	2	30

The following changes were applied:

- ✓ Companies have been renamed with indices from 0 to $\boldsymbol{n-1}$ (being n the number of companies).
- ✓ The weight of each investment have been divided by 1,000,000, and the profits by 10,000, to get numbers that are easier to handle.

1. To shape this problem as a QUBO model, first, the meaning of each binary variable (or binary decision in the context of this problem) should be defined: $\boldsymbol{x_j}$**: Should we invest in company *j* / Include item *j* in our knapsack? (1: Yes, 0: No)**

2. Then, define the objective of the problem: **Objective: Maximize the expected profit**

3. Constraint: **The overall weight of the items selected should be equal to $\boldsymbol{W}$ = 10**

Knapsack Problems

1) Profit Maximization

$$Maximize \sum_{j=0}^{n} p_j x_j = 12x_0 + 15x_1 + 8x_2 + 36x_3 + 20x_4 + 21x_5 + 30x_6$$

j	w_j	p_j
0	1	12
1	3	15
2	2	8
3	6	36
4	5	20
5	3	21
6	2	30

2) Constraint

$$subject\ to \sum_{j=0}^{n} w_j x_j = x_0 + 3x_1 + 2x_2 + 6x_3 + 5x_4 + 3x_5 + 2x_6 = 10 = W$$

We can now put everything together by adding both Hamiltonians and including the Lagrange multiplier, and making sure that the resulting expression meets the shape of a binary knapsack problem:

$$H = -H_{profit} + \alpha \cdot H_{weight} = -\sum_{j=0}^{n} p_j x_j + \alpha \cdot \left(W - \sum_{j=0}^{n} w_j x_j \right)^2$$

$$\boldsymbol{H = -(12x_0 + 15x_1 + 8x_2 + 36x_3 + 20x_4 + 21x_5 + 30x_6) + \alpha(10 - x_0 - 3x_1 - 2x_2 - 6x_3 - 5x_4 - 3x_5 - 2x_6 - s_0 - 2s_1 - 4s_2 - 3s_3)^2}$$

Source: https://en.wikipedia.org/wiki/Knapsack_problem

Knapsack Problems

Let's code!

Problem Formulation

Problem Formulation

Write down the Objective and the Constraints of the problem, in your domain business:

Definition

1. **Variables:** The variables is the first thing to define.
2. **Objective:** What we are looking to minimize or maximize. Common cases involve a measurable metric that we would like to maximize or minimize.
3. **Constraint:** Business rule that must be followed, that prevents some solutions from being acceptable or valid.
4. **Transformation:** Reformulate the problem for being compatible with QUBO formulation.

Problem Formulation

Removing squared terms

- QUBO models do not have squared variables

$$x_i \in \{0,1\} \Rightarrow \; {x_i}^2 = x_i$$

Maximization to minimization

- Sometimes, the problem will present itself as a maximization problem (for example, maximizing revenues).
- The QPU will look for the ground state, so it is best suited for minimization Problems.
- In these cases, we can just convert the entire expression to a minimization problem by multiplying it by -1

$$\arg\max\bigl(E(v)\bigr) = \arg\min\bigl(-E(v)\bigr)$$

Equality to Minimization

- In case we have a constraint based on an equality, we can convert it to a minimization expression by moving all terms to the same side of the equation and squaring the nonzero side of the equation.
- This way, we will ensure that the expression is satisfied on a minimal value of 0 and unsatisfied on any greater value greater than 0.

$$x_1 = 1 \Rightarrow \; (x_1 - 1)^2 = 0$$

Problem Formulation

Inequality to Minimization

Let us consider the following inequalities obtained when trying to model a problem as a QUBO $\sum_{i=0}^{n} a_i v_i \leq m, m \in \mathbb{Z}$ $\sum_{i=0}^{n} a_i v_i \geq m, m \in \mathbb{Z}$

These situations are very common when working with constraints, that put an upper or a lower limit on some aspects of our solutions. How can we convert this inequalities to a minimization problem?

- If we just convert the $\leq$ sign to = we would be limiting the model to solutions with a total value equal to m.
- To add some flexibility, we will be adding slack variables, auxiliary variables whose final value will not be relevant (in fact, it is usually ignored) but help the model consider other solutions
- The new equation would look something like this, being S the slack variable
- We will use slack variables to allow the QPU or simulator to use them if needed, to set either a value of 0 or 1 in any of them to accommodate the expression and meet the requirement set by the constraint

$$\sum_{i=0}^{n} a_i v_i - m \leq 0 \longrightarrow \left(\sum_{i=0}^{n} a_i v_i - m + S\right)^2 = 0$$

$$\sum_{i=0}^{n} a_i v_i - m \geq 0 \longrightarrow \left(\sum_{i=0}^{n} a_i v_i - m - S\right)^2 = 0$$

Problem Formulation

Inequality to Minimization

$$\sum_{i=0}^{n} a_i v_i - m \leq 0 \longrightarrow \left(\sum_{i=0}^{n} a_i v_i - m + S\right)^2 = 0$$

- We will use binary expansion on $\boldsymbol{S}$.

- We should start by determining the minimum value that the S value has to compensate, and then, the maximum value that S may need to reach to compensate the expression, and how many slack variables will be needed to make that possible:
    - The minimum value in this example will be 0
    - The maximum value will be m

- We should look for the power of 2 equal to the maximum value of $\boldsymbol{S}$ obtained earlier. This will help us identify how many binary variables will be required

$$S \in \{0, 1, 2, \ldots, 6\} \to 6_{dec} = 110_{bin} \to s_0, s_1, s_2\,;\, s_i \in \{0, 1\}\,;\, S = 4s_0 + 2s_1 + 1s_2 \longrightarrow S = 3s_0 + 2s_1 + 1s_2$$

- Therefore, the final equation would look as follows just for our example the number of slack variables and their coefficients will vary from case to case:

$$\sum_{i=0}^{n} a_i v_i + 3s_0 + 2s_1 + 1s_2 = m$$

Problem Formulation

Inequality to Minimization

Summary

1. Less Than or Equal To inequality can be summarized as follows

$$\sum_{i=0}^{n} a_i v_i \leq m \,, m \in \mathbb{Z} \rightarrow \sum_{i=0}^{n} a_i v_i + S = m$$

Apply binary expansion to S considering $max\, S = m - min \sum_{i=0}^{n} a_i v_i$

2. Greater Than or Equal To we will apply the same strategy, modifying just some aspects of the operation

$$\sum_{i=0}^{n} a_i v_i \geq m \,, m \in \mathbb{Z} \rightarrow \sum_{i=0}^{n} a_i v_i - S = m$$

Apply binary expansion to S considering $max\, S = max \sum_{i=0}^{n} a_i v_i - m$

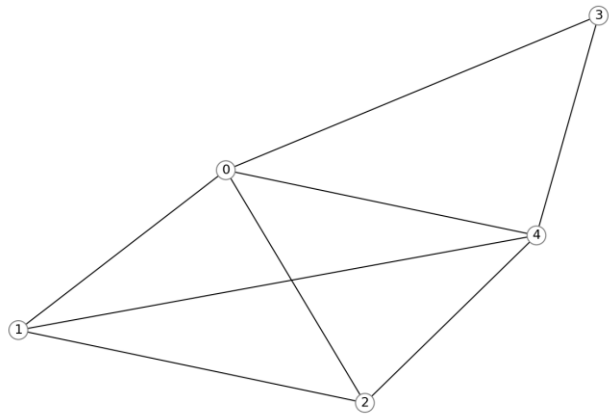
Problem Formulation

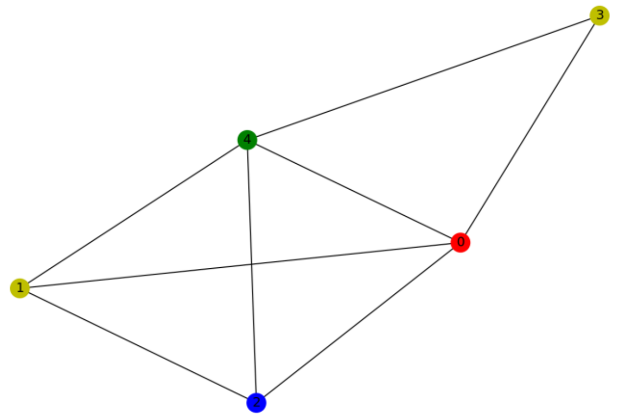
Let's code!

Bonus: Graph Coloring

Graph Coloring

- ✓ Graph coloring is a classical NP-hard problem from graph theory
- ✓ **Goal**: assign colors to the nodes of a graph such that adjacent nodes are assigned different colors
- ✓ It has a wide range of applications such as scheduling, register allocation, designing seating plans, Frequency Assignment , etc.





Graph Coloring

- ✓ Formally: given a graph $G = (N, E)$, and a set of k different colors C, assign a mapping $f: V \rightarrow C$ such that $f(v_i) = f(v_j)$ if and only if $(v_i, v_j) \notin E$
- ✓ Variable $x_{n,k} = 1$ if vertex n is colored with color labelled as k

Constraints

Each vertex is assigned one color only

$$\sum_{k=0}^{K-1} x_{n,k} = 1 \,, \forall n$$

Adjacent edges can't have same color

$$given\ edge\ (n,m) \in E, x_{n,k} = 0\ or\ x_{m,k} = 0, \forall k$$

Graph Coloring

- ✓ Formally: given a graph $G = (N, E)$, and a set of k different colors C, assign a mapping $f: V \rightarrow C$ such that $f(v_i) = f(v_j)$ if and only if $(v_i, v_j) \notin E$

- ✓ Variable $x_{n,k} = 1$ if vertex n is colored with color labelled as k

Constraints

Each vertex is assigned one color only

$$\sum_{k=0}^{K-1} x_{n,k} = 1, \forall n \longrightarrow \sum_{n=0}^{N-1}\left(1 - \sum_{k=0}^{K-1} x_{n,k}\right)^2$$

Adjacent edges can't have same color

$$given\ edge\ (n,m) \in E, x_{n,k} = 0\ or\ x_{m,k} = 0, \forall k \longrightarrow \sum_{(n,m)\in E} \sum_{k=0}^{K-1} x_{n,k} x_{m,k}$$

Graph Coloring

✓ Formally: given a graph $G = (N, V)$, and a set of k different colors C, assign a mapping $f: V \to C$ such that $f(v_i) = f(v_j)$ if and only if $(v_i, v_j) \notin V$

✓ Variable $x_{n,k} = 1$ if vertex n is colored with color labelled as k

QUBO

$$H = \alpha \sum_{n=0}^{N-1} \left(1 - \sum_{k=0}^{K-1} x_{n,k}\right)^2 + \beta \sum_{(n,m)\in E} \sum_{k=0}^{K-1} x_{n,k} x_{m,k}$$

Graph Coloring

Let's Code!

Thank you very much!