Advanced Parallel School 2022
Quantum Computing – Day 3
Qubo formulation of NP-Hard problems

Mengoni Riccardo, PhD

16 Feb 2022

CINECA QUANTUM COMPUTING LAB

Content

- QUANTUM ANNEALING
- NP-HARD PROBLEMS
- UNSUPERVISED LEARNING
  - K-MEANS CLUSTERING
  - QUANTUM ANNEALING FOR CLUSTERING
  - MINIMUM SPANNING TREE CLUSTERING
  - QUANTUM ANNEALING FOR MINIMUM SPANNING TREE
- SUPERVISED LEARNING
  - KERNEL METHODS AND SUPPORT VECTOR MACHINE
  - GRAPH EDIT DISTANCE
  - QUANTUM ANNEALING FOR GRAPH EDIT DISTANCE

0. Quantum Annealing

Quantum Annealing

Annealing in hardware. Once a native instance is formulated, it can be sent to the quantum annealing hardware for solution. The quantum annealing algorithm used by D-Wave incorporates the initial Hamiltonian

Initial Hamiltonian:

$$\mathcal{H}_I = \sum_i \sigma_i^x,$$

and the problem Hamiltonian

Final Hamiltonian:

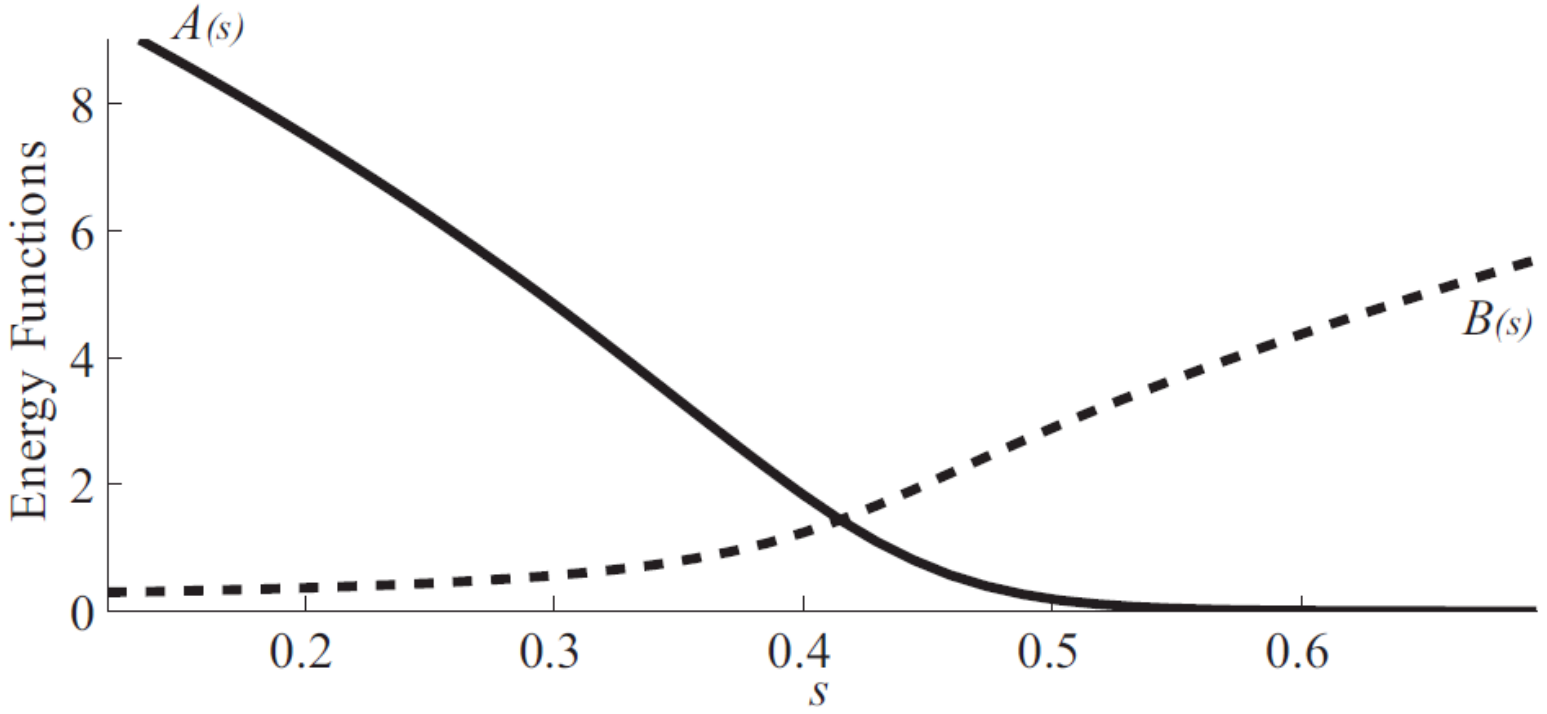
$$\mathcal{H}_{IM} = \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z,$$

where h_i and J_{ij} are constrained to match the hardware working graph.

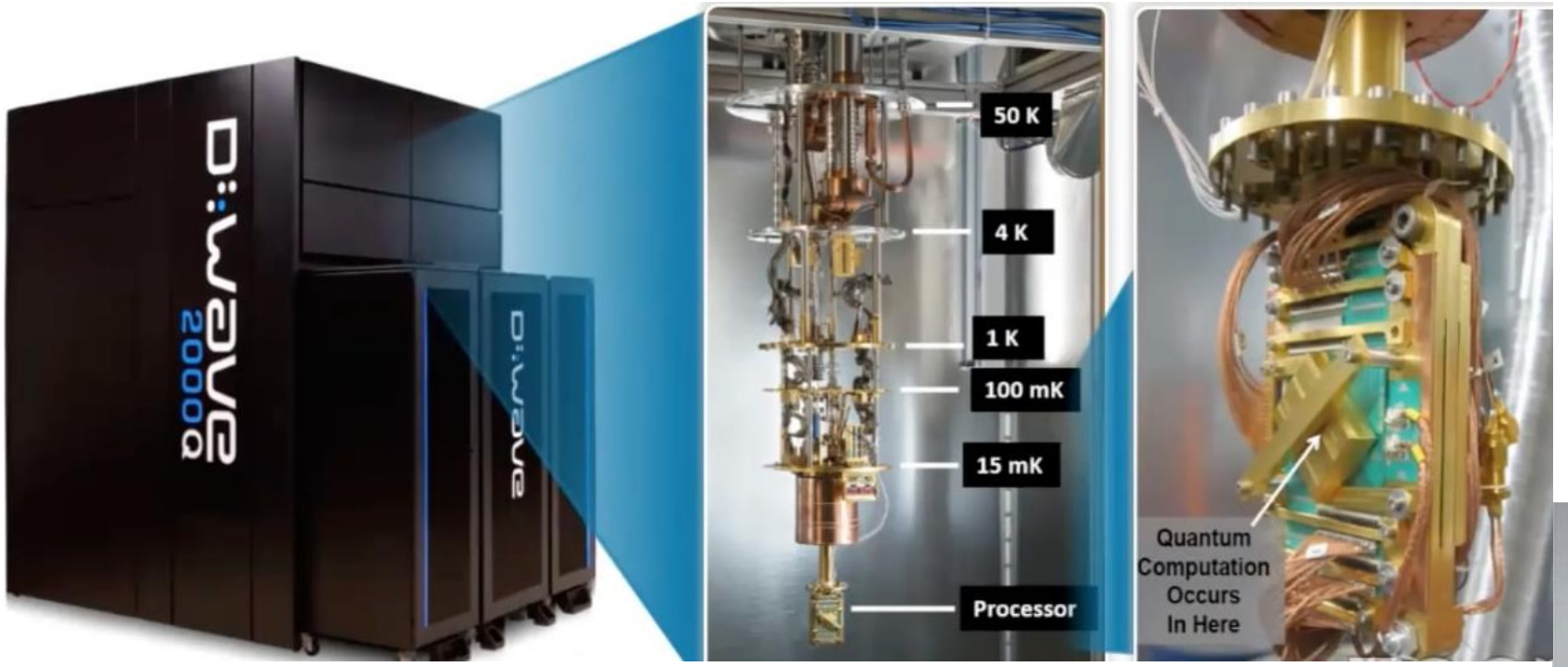
Quantum Annealing

Thus D-Wave quantum annealers implement the following Hamiltonian:

$$\mathcal{H}(s) = A(s)\mathcal{H}_I + B(s)\mathcal{H}_{IM}.$$

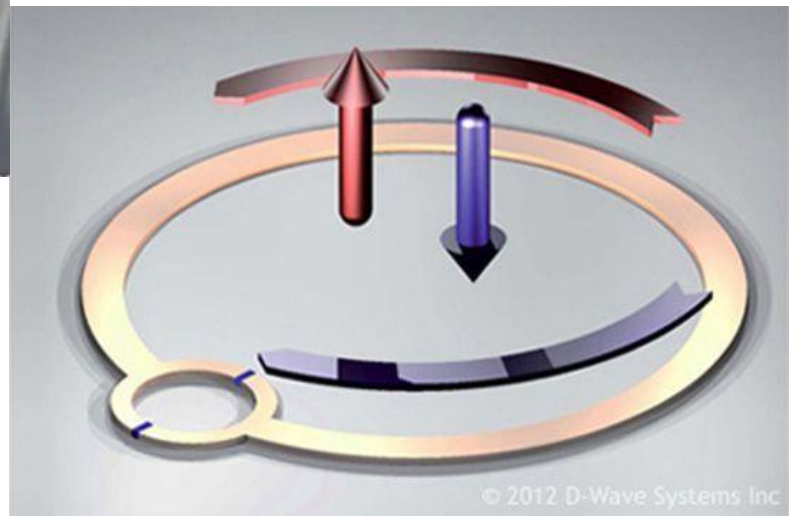


D-Wave Quantum Annealer



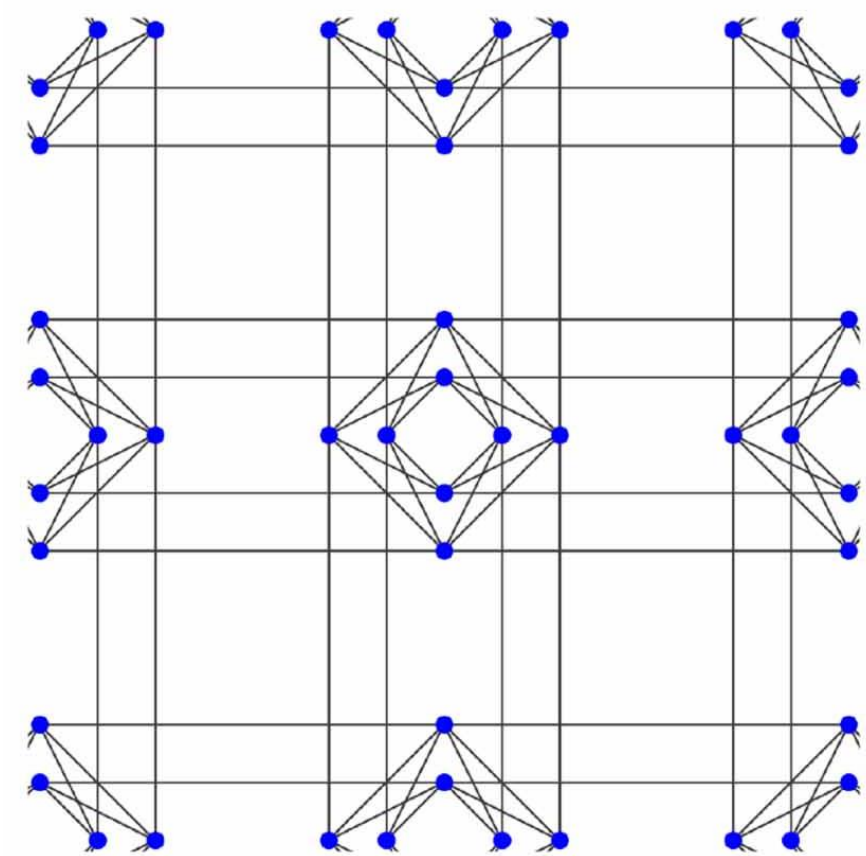
Two Annealers devices:
- 2000 qubits
- 5000 qubits

Superconducting Circuits

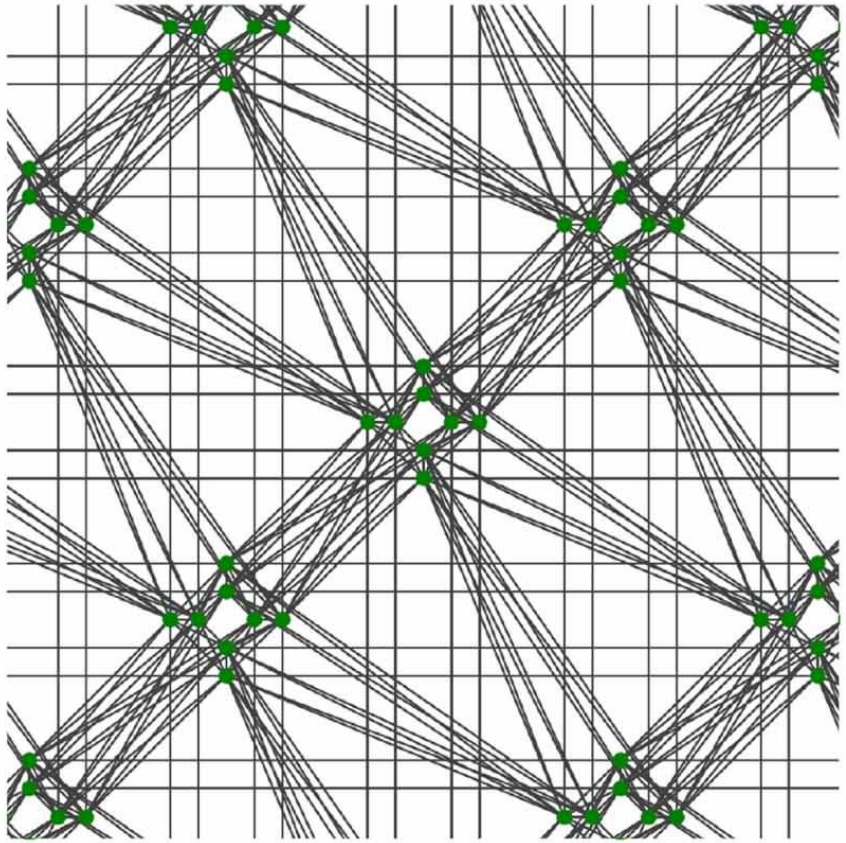


D-Wave Architecture

The building blocks of the D-wave architecture are the **Chimera or Pegasus Graphs**



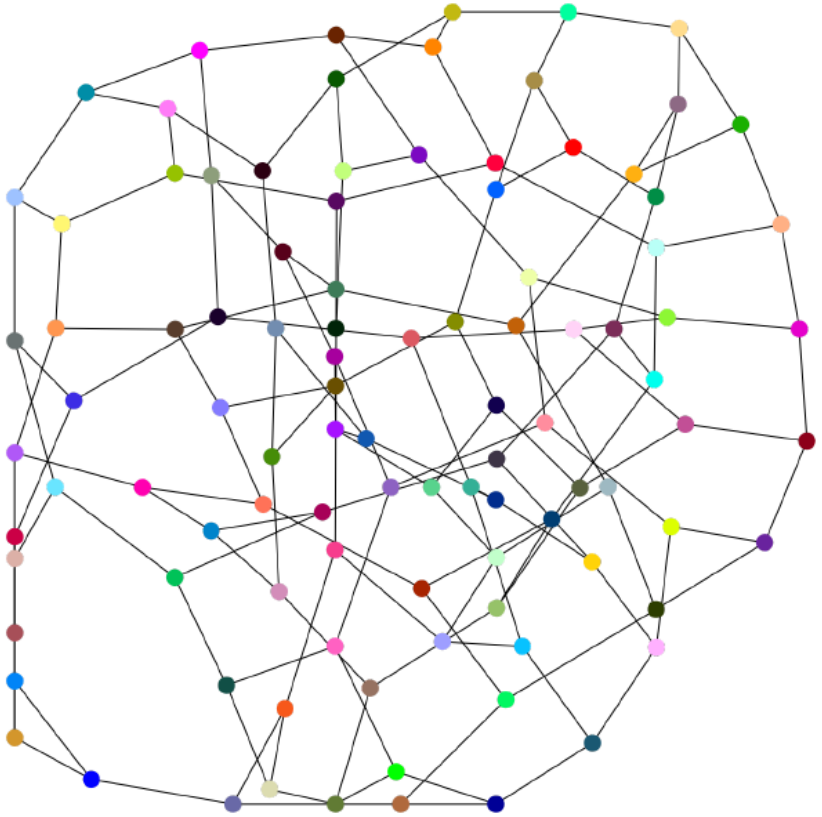
(c) Chimera cells.



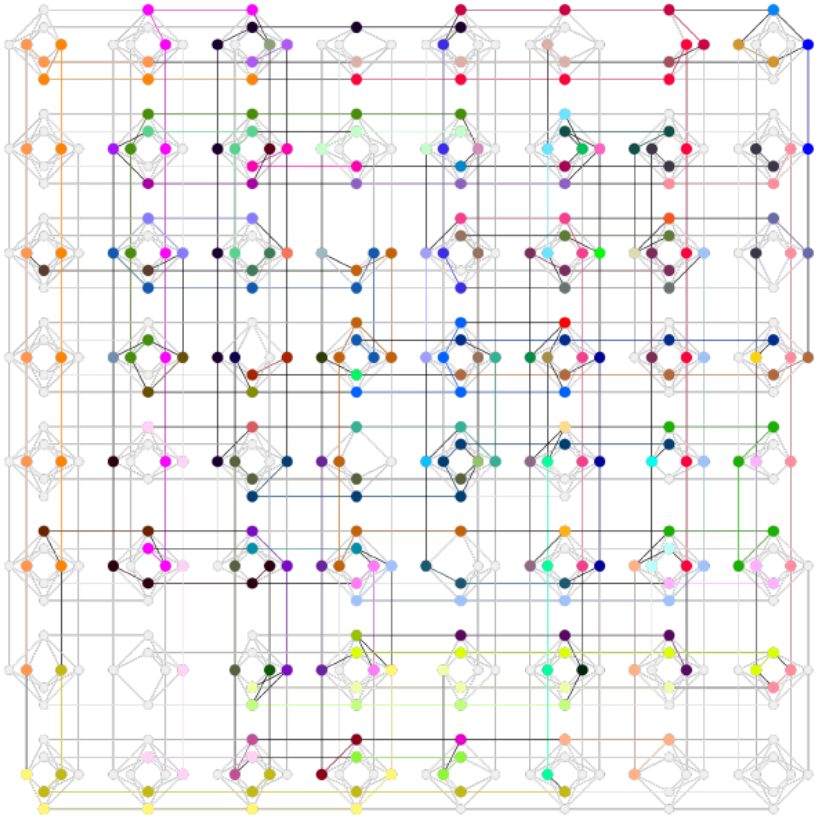
(d) Pegasus cells.

Minor Embedding

Logical Ising (or QUBO) problem



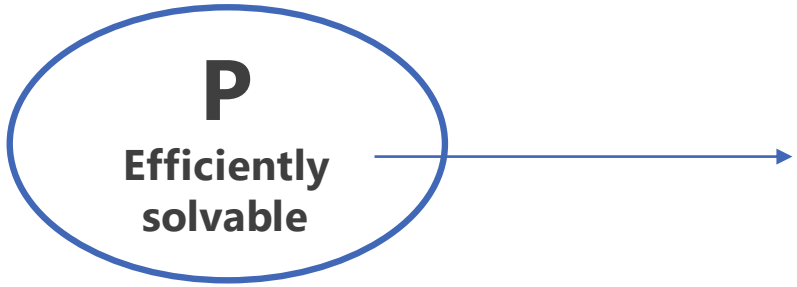
Embedded problem



Minor-embedding a general graph G into a hardware working graph H.

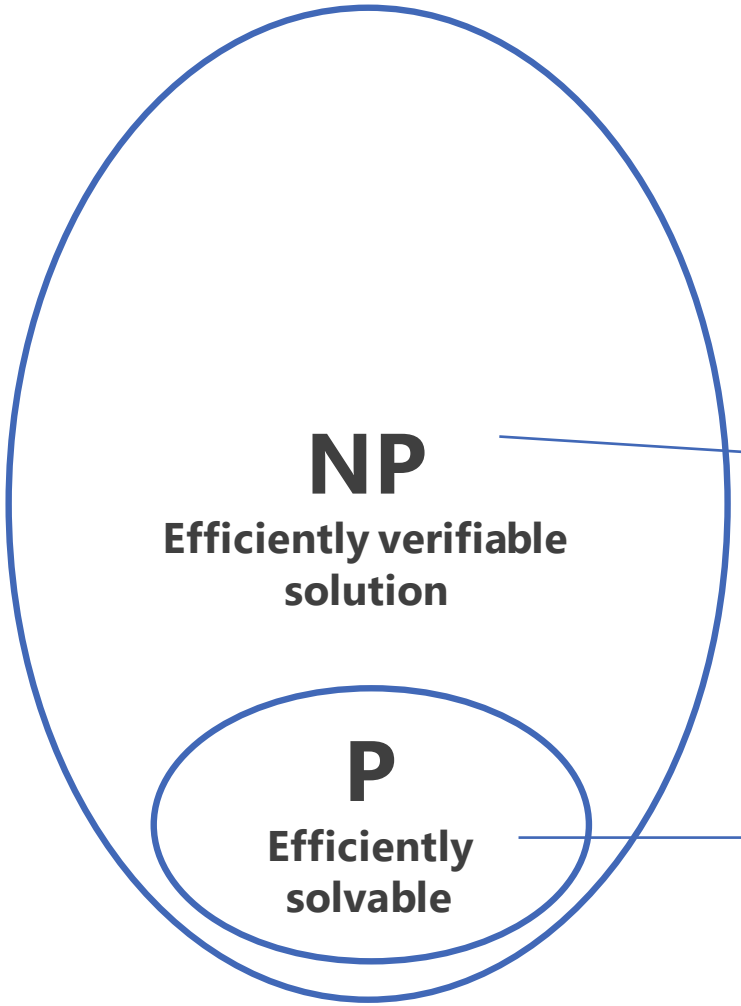
1. NP-Hard Problems

NP-Hard problems



- Sorting
- Multiplication
- …

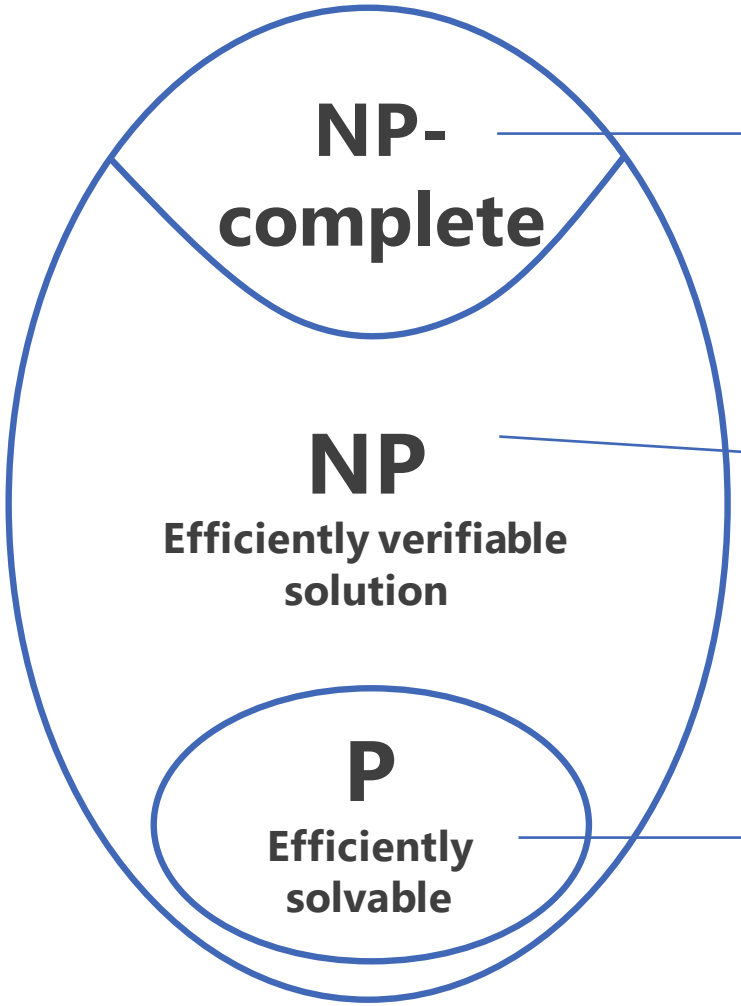
NP-Hard problems



- Factorization
- Graph isomorphism
- …

- Sorting
- Multiplication
- …

NP-Hard problems

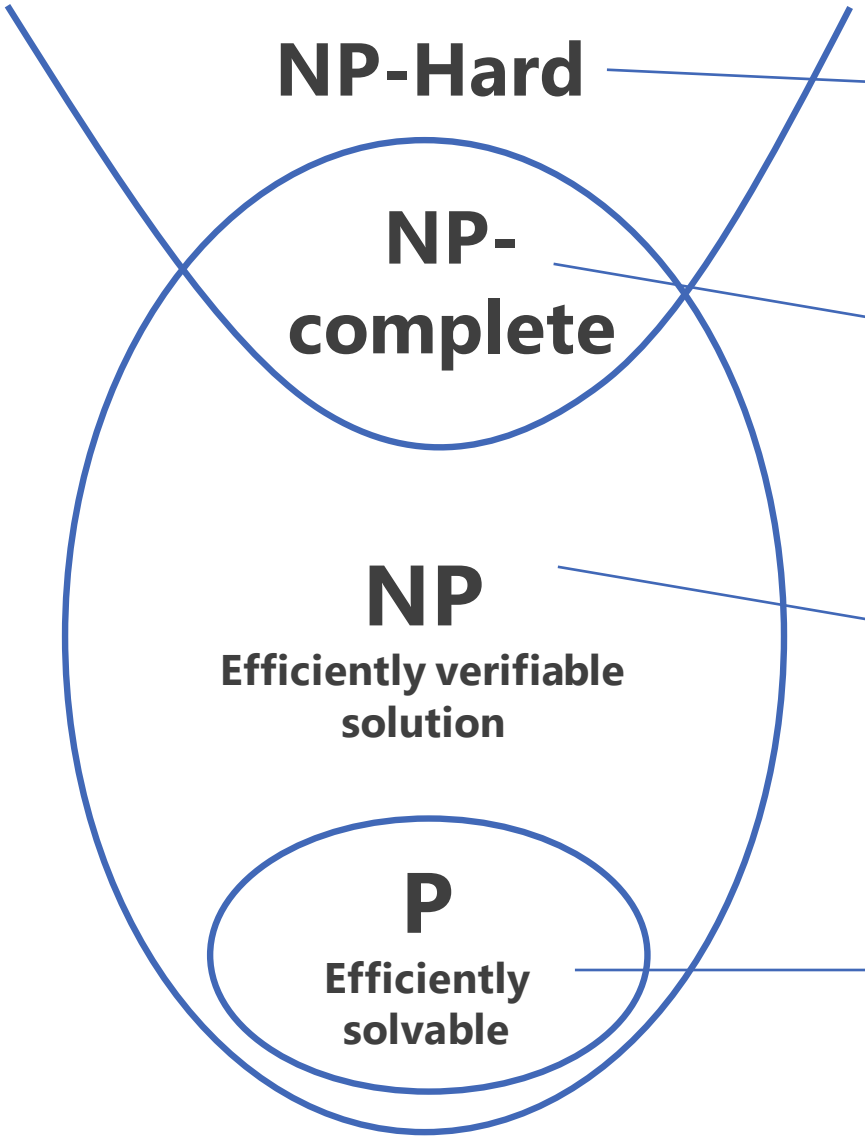


- Travelling Salesman Problem
- NxN Sudoku
- SAT
- …

- Factorization
- Graph isomorphism
- …

- Sorting
- Multiplication
- …

NP-Hard problems



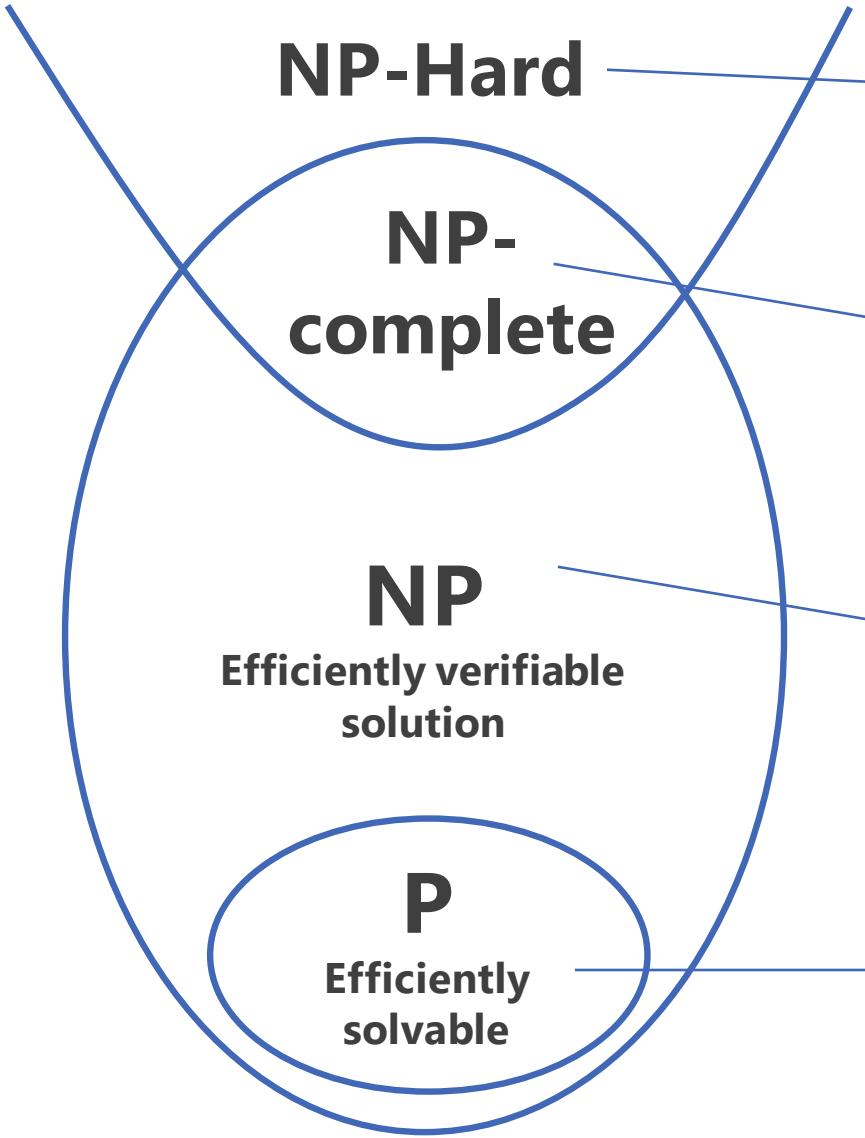
- Optimization
- Some ML problems
- ..

- Travelling Salesman Problem
- NxN Sudoku
- SAT
- …

- Factorization
- Graph isomorphism
- …

- Sorting
- Multiplication
- …

NP-Hard problems



- Optimization
- Some ML problems
- ..

- Travelling Salesman Problem
- NxN Sudoku
- SAT
- …

- Factorization
- Graph isomorphism
- …

- Sorting
- Multiplication
- …

NP-Hard problems formulation

NP-Hard Problem

map

QUBO or Ising Problem

$$\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

NP-Hard problems formulation

NP-Hard Problem → map → **QUBO or Ising Problem**

$$\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

Not always easy..

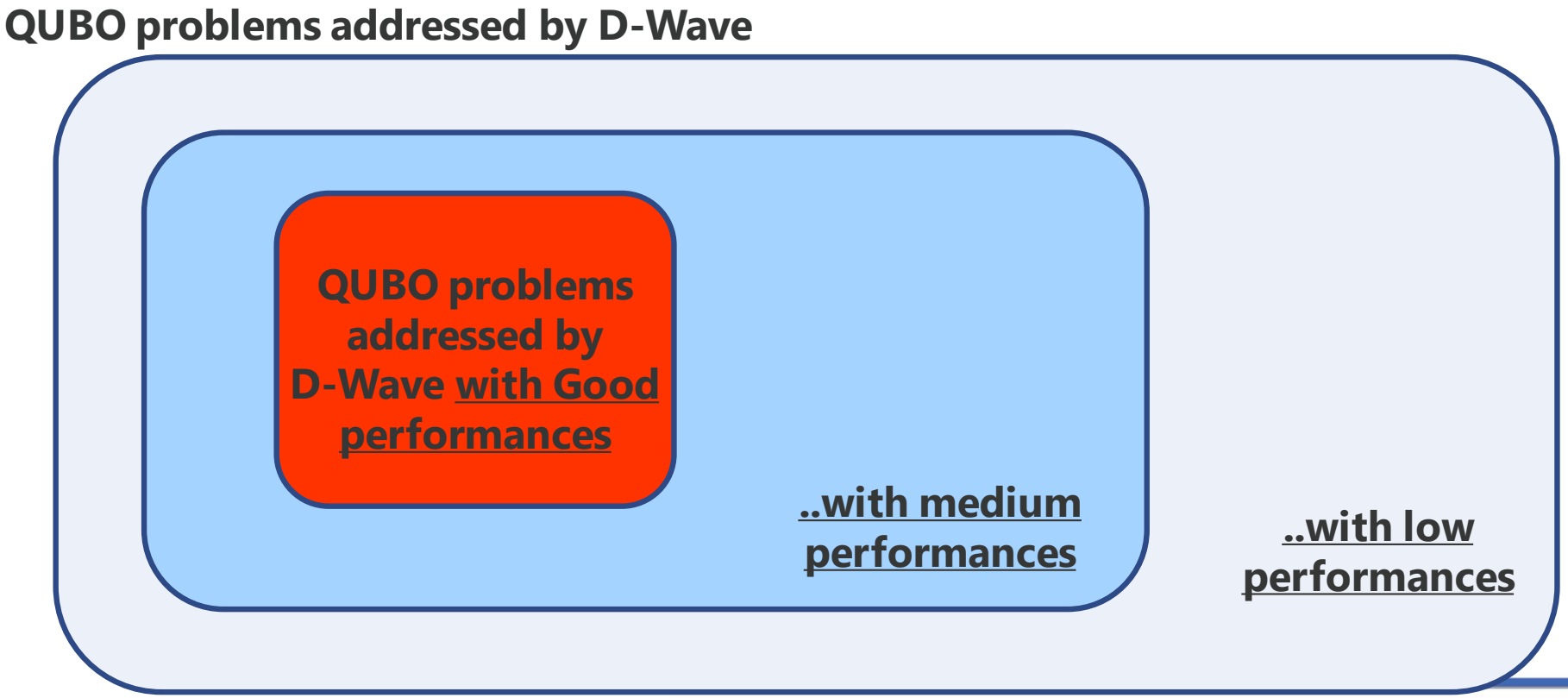
NP-Hard Problem → map → **HUBO Problem (High Order Binary Optimization)** → map → **QUBO or Ising Problem**

$$\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

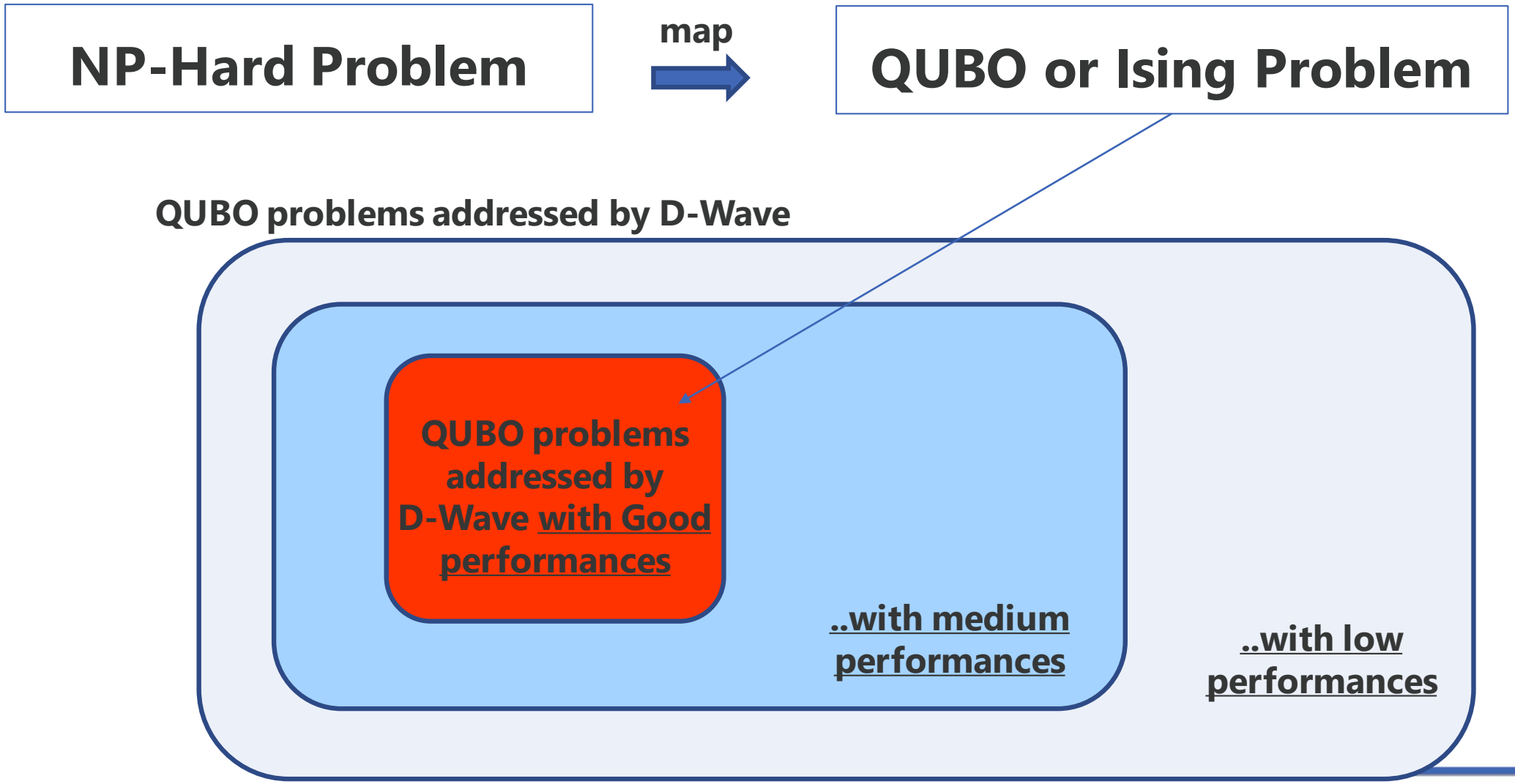
Overhead number of variables and quadratic terms

NP-Hard problems formulation + Hardware limitations

NP-Hard Problem → map → **QUBO or Ising Problem**



NP-Hard problems formulation + Hardware limitations



NP-Hard problems formulation + Hardware limitations

QUBO problems addressed by D-Wave with Good performances

$$\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

- **Number of resources (variables) grow polynomially wrt problem size**
- **Short Chains of physical qubits representing single locial qubit**
- **Limited range of coefficients:** $h_i, J_{ij} \in [-1,1]$
- **Limited precision:** when $h_j \neq h_i$ and $J_{kl} \neq J_{ij}$

$$\Delta h = \min_{ij}(h_i - h_j) \geq 0.01$$
$$\Delta J = \min_{ijkl}(J_{ij} - J_{kl}) \geq 0.01$$

NP-Hard problems formulation + Hardware limitations

QUBO problems addressed by D-Wave with Good performances

$$\sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$$

Best default usage

- **Annealing schedule (Time per each anneal cycle)**
  $T_a \in [1\ \mu sec, 200 \mu sec]$
  Optimal $1\ \mu sec \leq T_a \leq 10$ µsec

- **Annealing cycles** 10.000 up to $1.000.000$

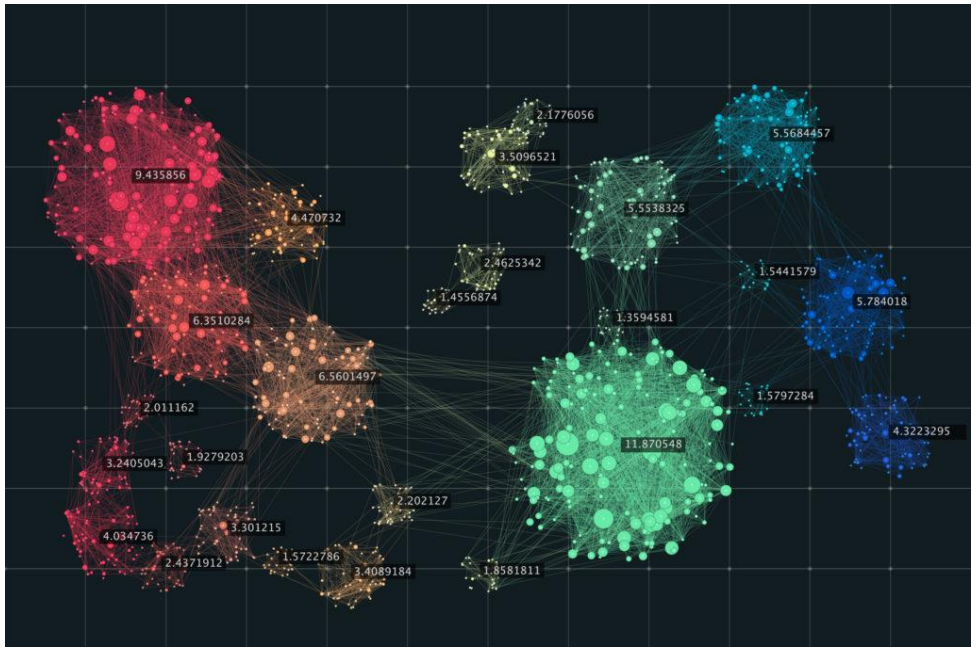
- **Extended J range ($J_{kl} = -2$) for chain of physical qubits representing single logical qubit (resolved with majority voting)**

- **Advance methods (reverse annealing, pause, quench) (increase performance but also runtime)**

2. Unsupervised Learning

Unsupervised Learning

The algorithm is provided with an **unlabelled dataset** with the goal of finding **patterns and structures without any prior experience of the problem**.



A standard examples of unsupervised learning task is **clustering** where **data-points** in the form of multidimensional vector are **assigned labels and grouped in such a way to minimize within-group distance** while **maximizing the margin between different classes.**

K-means Clustering

K-Means Clustering

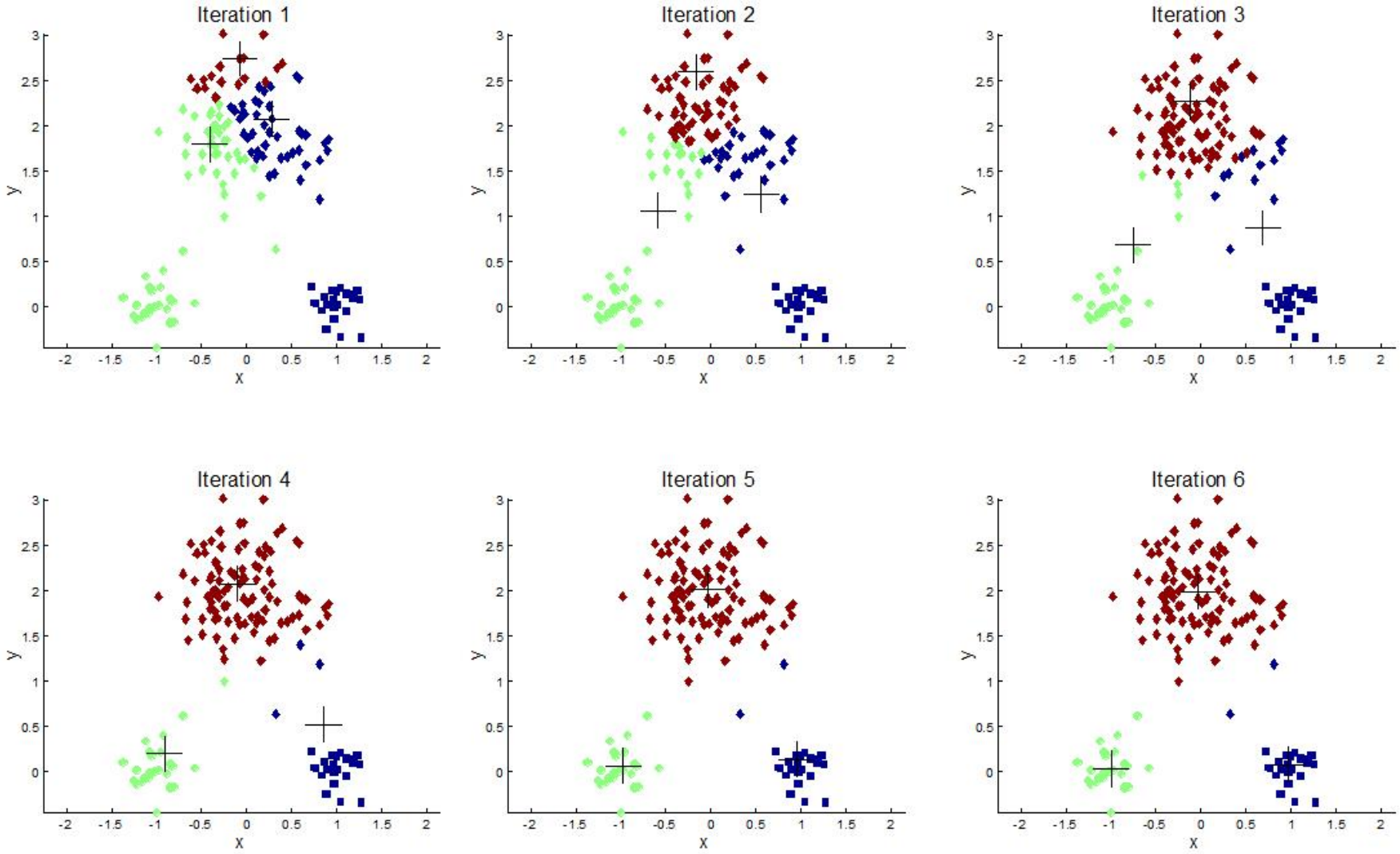
More in detail, consider an unlabelled dataset $\mathcal{D}$ containing M feature vectors $\vec{x} \in \mathbb{R}^N$, the k-means algorithm aims to partition the dataset into $k \leq M$ sets $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ in order to minimize the within-cluster variance. The objective function is the following

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{\vec{x} \in C_i} \|\vec{x} - \vec{\mu}_i\|^2 \tag{1.12}$$

where $\vec{\mu}_i$ is the centroid or mean vector calculated among those points belonging to the same class C_i

$$\vec{\mu}_i = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}. \tag{1.13}$$

K-Means Clustering



K-Means Clustering

The algorithm at first **randomly select k centroids** and then follows an iterative procedure that alternates between two steps.

Repeated until convergence

- An **assignment step** where each vector is assigned to the cluster whose centroid has the least squared Euclidean distance.
- An **updating step** where it is calculated the new centroids of the vectors in the new clusters.

The algorithm **does not guarantee a convergence** to a global optimum **but** towards a **local minimum**
Moreover the overall performance is largely affected by the initial choice of the centroids.

From the complexity point of view, finding the **optimal solution for the k-means clustering problem is NP-hard in the Euclidean space**

Let's try to formulate the clustering problem as a combinatorial optimization problem that we can address with Quantum Annealing

Quantum Annealing for Clustering

Quantum Annealing for Clustering

The idea is the following: consider a dataset $\mathcal{D}$ containing M feature vectors $\vec{x}_i \in \mathbb{R}^N$ that we want to partition into $k \leq M$ sets $\mathbf{C} = \{C_1, C_2, ..., C_K\}$ in order to minimize the within-cluster variance. We can start defining a binary variable $z_i^{(k)}$ that is 1 only when the vector $\vec{x}_i$ belongs to class C_i, in other words

$$z_i^{(k)} = \begin{cases} 1 \text{ iff } \vec{x}_i \in C_k \\ 0 \text{ iff } \vec{x}_i \notin C_k \end{cases}$$

Since every vector $\vec{x_i}$ must be associated to a single class, this means that

$$\forall \vec{x}_i \qquad \sum_{k=1}^{K} z_i^{(k)} = 1$$

Quantum Annealing for Clustering

$$z_i^{(k)} = \begin{cases} 1 \text{ iff } \vec{x}_i \in C_i \\ 0 \text{ iff } \vec{x}_i \notin C_i \end{cases} \quad + \quad \forall \vec{x}_i \quad \sum_{k=1}^{K} z_i^{(k)} = 1$$

This condition can be rephrased as an objective function as follows

$$f_1(z) = \sum_{i=1}^{M} \left(\sum_{k=1}^{K} z_i^{(k)} - 1 \right)^2$$

Quantum Annealing for Clustering

Moreover, since we want the distance between vectors $D_{i,j} = |\vec{x}_i - \vec{x}_j|$ to define the clusters, i.e we want to group together vectors that are closer, it is necessary to add the penalty term

$$f_2(z) = \sum_{k=1}^{K} \sum_{i \neq j} D_{i,j} z_i^{(k)} z_j^{(k)}$$

The **QUBO** associated to the clustering problem is therefore

$$O(z) = \lambda \sum_{i=1}^{M} \left(\sum_{k=1}^{K} z_i^{(k)} - 1 \right)^2 + \sum_{k=1}^{K} \sum_{i \neq j} D_{i,j} z_i^{(k)} z_j^{(k)}$$

where parameter λ was inserted to modulate the strength of **the first term**, which represent the **hard constraint** of the problem.

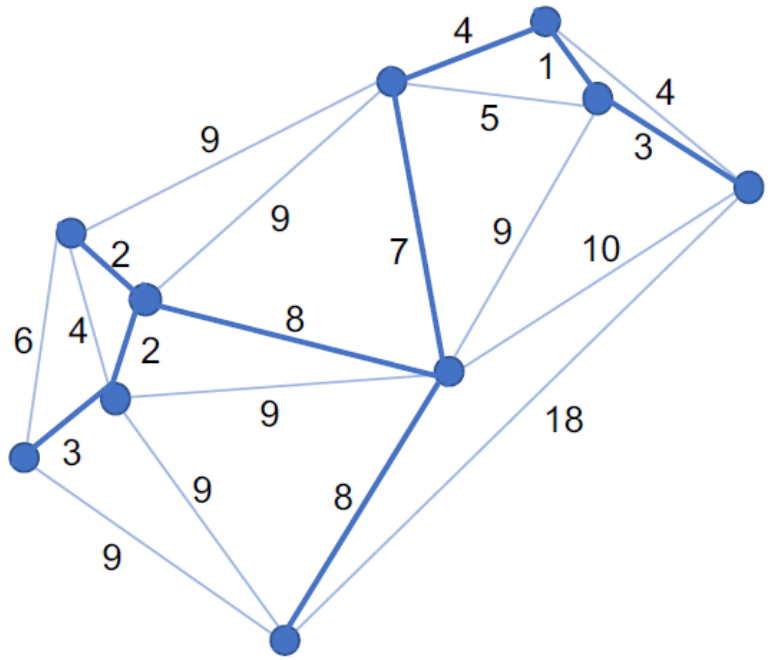
Minimum Spanning Tree Clustering

Minimum Spanning Tree Clustering

Intro to the Minimum Spanning Tree problem

Consider a weighted graph $G = (V, E, w)$ composed of a set V of n vertices, a set E of edges and a function $w : E \to \mathbb{R}$ that associates a weight $w_{v,v'}$ to each edge $(v, v') \in E$.

The **Minimum Spanning Tree (MST)** problem aims at **finding a subgraph T of G such that**: it does not contain cycles (i.e. T is a tree), it touches all the n vertices of G and it **minimizes the sum overall edge weights.**



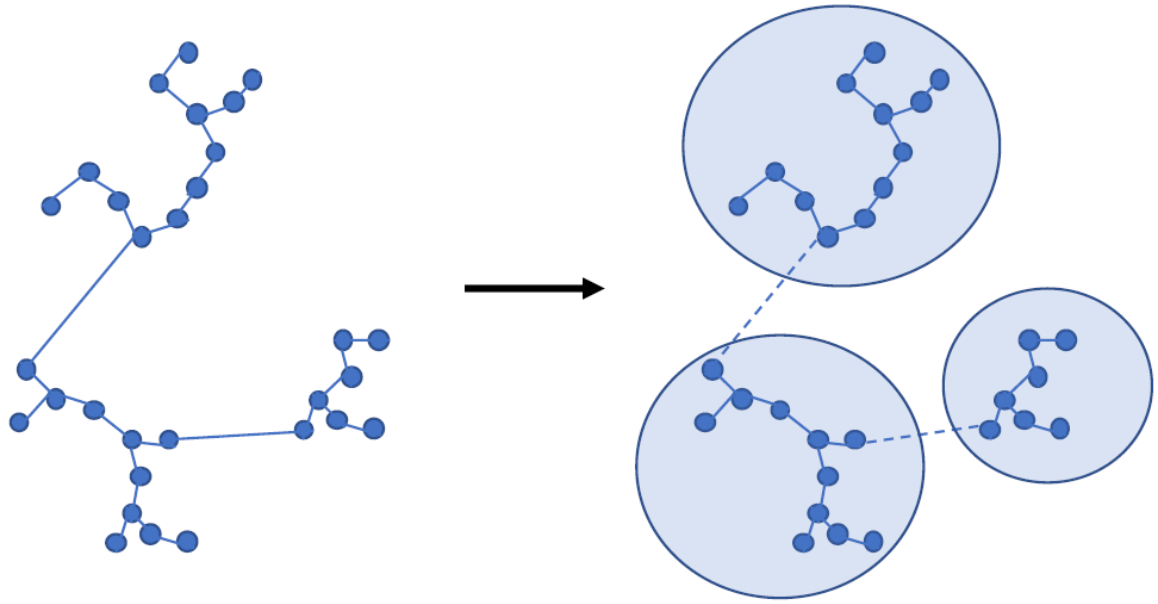
Minimum Spanning Tree Clustering

Given an unlabelled dataset $\mathcal{D}$ containing n feature vectors $\vec{x_i} \in \mathbb{R}^N$, the MST based clustering algorithm considers the n points $\vec{x_i}$ as nodes of a complete weighted graph G, where weights $w_{i,j}$ represent the euclidean distance between any pair of points $\vec{x_i}$ and $\vec{x_j}$.

$$w_{i,j} = d(\vec{x_i}, \vec{x_j}) \tag{1.14}$$

The algorithm computes the MST of the weighted graph G.

If the **number of clusters k is known in advance**, the algorithm **sorts the edges** of the MST in descending order and **remove the (k-1) edges with heaviest weights**



Minimum Spanning Tree Clustering

Delta Degree Minimum Spanning is NP-hard

In general, finding the MST of a given graph is not a hard task for a classical computer. In fact, given a graph where e is the number of edges and n is the number of vertices, the MST problem can be exactly solved in $O(e \log n)$ time. However, with an additional constraint called Δ-Degree constraint (i.e. each vertex in the MST must have degree less or equal than Δ), where $2 \leq \Delta < n - 1$, the problem becomes NP-hard and the corresponding decision problem NP-complete [9]. Finally, note that if the initial graph is complete, then a MST with Δ-Degree constraint is ensured to exist.

Let's try to formulate the Delta Degree Minimum Spanning Tree clustering problem as a combinatorial optimization problem that we can address with Quantum Annealing

Quantum Annealing for MST Clustering

Quantum Annealing for MST Clustering

Consider an ordering of the vertices, i.e. a bijective map that associates to each vertex only one indexed $i \in \{1, 2, ..., n\}$ and vice versa. Define the binary variables $x_{v,i}$ and $y_{p,i}$ such that

$$x_{v,i} = \begin{cases} 1 \text{ iff } v \text{ is in position } i \\ 0 \text{ otherwise} \end{cases}$$

and

$$y_{p,i} = \begin{cases} 1 \text{ iff } p \text{ is parent of vertex in position } i \\ 0 \text{ otherwise} \end{cases}$$

Because the root has no parent, $y_{p,1}$ must be zero for every vertex. Moreover, since the root has been set from the beginning, it is already known that $y_{p,2} = 1$ only when p is the root. Hence, we can safely restrict the variable $y_{p,i}$ to the cases where $p \in V$ and $i \geq 3$.

Quantum Annealing for MST Clustering

Let's now introduce the hard constraints these two variables have to satisfy in order to realize the MST. Since we considered a total ordering of the vertices, the following two QUBO terms should be considered:

$$\sum_{v \in V \setminus \{r\}} \left(\sum_{i=2}^{n} x_{v,i} - 1 \right)^2 + \sum_{i=2}^{n} \left(\sum_{v \in V \setminus \{r\}} x_{v,i} - 1 \right)^2$$

Where the first term ensures that only one i is associated to every v while the second term guarantees the converse.

Moreover, a necessary property to obtain a tree requires that each vertex has only one parent. Such property is expressed with the QUBO constraint

$$\sum_{i=3}^{n} \left(\sum_{p \in V} y_{p,i} - 1 \right)^2$$

Quantum Annealing for MST Clustering

Now we need to secure that, if p is in the parent of the vertex in position i, then p shouldn't be at greater or equal position. In QUBO form, such condition becomes

$$\sum_{p \in V \setminus \{r\}} \sum_{i=3}^{n} \sum_{j \geq i} y_{p,i} x_{p,j}$$

Finally, in order to make sure that parents and children are indeed neighbours, the following term that penalizes non-neighbours should be taken into account

$$\sum_{p \in V} \sum_{v \notin N(p), v \neq r} \sum_{i=3}^{n} x_{v,i} y_{p,i}$$

Note that this term does not give any contribution when the graph is complete, since all vertices are neighbours. Anyway we decided to insert this term form completeness, since it becomes relevant when the input graph is not complete.

Quantum Annealing for MST Clustering

The final QUBO form for the MST problem with Δ-degree constrain is

$$
\begin{aligned}
& A \sum_{v \in V \setminus \{r\}} \left(\sum_{i=2}^{n} x_{v,i} - 1 \right)^2 + A \sum_{i=2}^{n} \left(\sum_{v \in V \setminus \{r\}} x_{v,i} - 1 \right)^2 + \\
& + A \sum_{i=3}^{n} \left(\sum_{p \in V} y_{p,i} - 1 \right)^2 + A \sum_{p \in V \setminus \{r\}} \sum_{i=3}^{n} \sum_{j \geq i} y_{p,i} x_{p,j} + \\
& + A \left(\sum_{p \in V} \sum_{v \notin N(p), v \neq r} \sum_{i=3}^{n} x_{v,i} y_{p,i} \right) + A \sum_{p \in V} \left(\sum_{i=3}^{n} y_{p,i} - \sum_{j=1}^{\Delta - 1} z_{p,j} \right)^2 + \\
& + B \left(\sum_{v \in N(r)} x_{v,2} w_{r,v} + \sum_{v \in V \setminus \{r\}} \sum_{p \in V} \sum_{i=3}^{n} y_{p,i} x_{v,i} w_{p,v} \right)
\end{aligned}
\tag{7.7}
$$

where the last term minimizes the sum of edge weights w.

Quantum Annealing for MST Clustering

Delta Degree constraint

In order to realize the Δ-Degree constraint, a number $\Delta - 1$ of binary slack variables $z_{p,j}$ are introduced for each vertex. The QUBO term for this constrain is

$$\sum_{p \in V} \left(\sum_{i=3}^{n} y_{p,i} - \sum_{j=1}^{\Delta-1} z_{p,j} \right)^2$$

Parameter Setting

As we can see from Eq. 7.7, parameter A have been introduced to define the strength of the hard constraints with respect to the cost term, multiplied instead by parameter B. In general, $A >> B$ to ensure that the problem is well defined. In our case, the minimum value of A for which the problem is well defined is given by the relation

$$\frac{A}{B} > \max_{v,v' \in V} \{w_{v,v'}\}$$

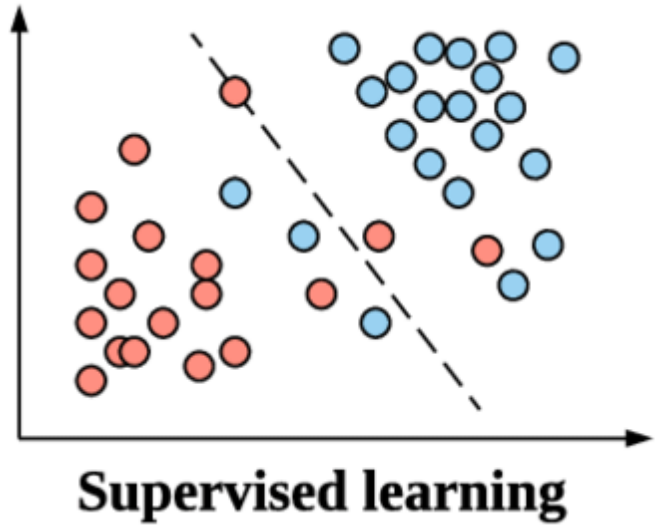
3. Supervised Learning

Supervised Learning

It is given an **ensemble of labelled data points** usually called **training set**.

The learning algorithm has the task **to induce a classifier from these labelled samples**.

The **classifier is a function that allocate labels to inputs**, including those that are out of the training set which have never been previously analyzed by the algorithm.

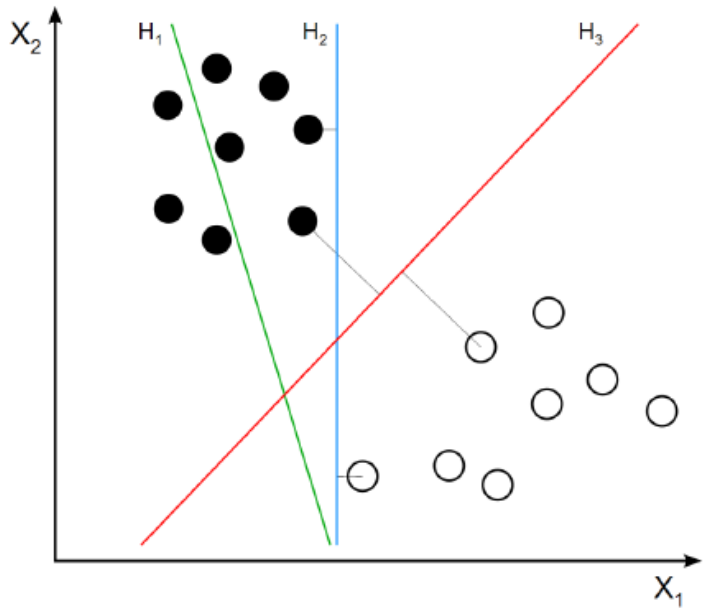


- **Classification**
- **Regression**

Kernel methods and SVMs

Support Vector Machine (SVM)

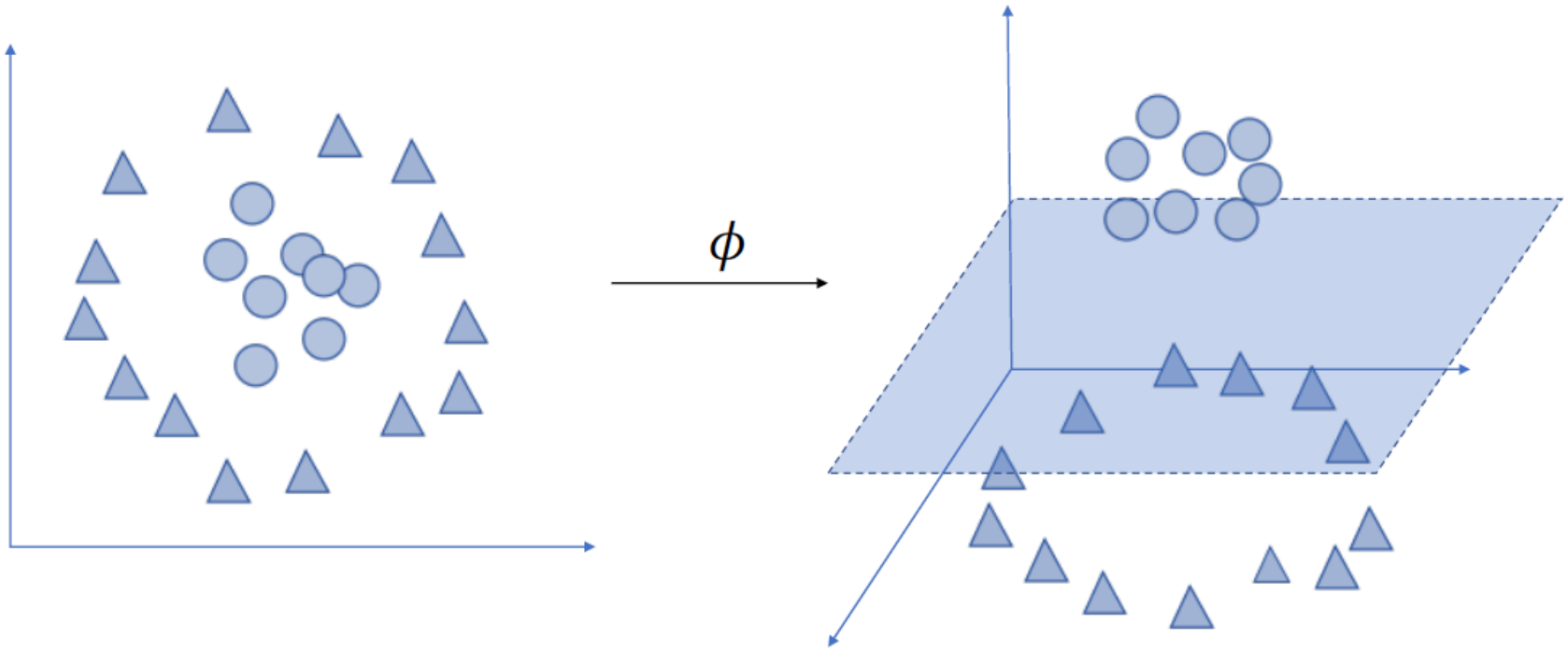
The task is to find a **hyperplane** that is the best discrimination between two class regions and serves as a decision boundary for future classification tasks.



input: M training vectors $\{(\vec{x}, y) \mid \vec{x} \in \mathbb{R}^N,\ y \in \{-1, +1\}\}$

Kernel Methods

Kernel methods [115] are classification algorithms in ML that use a kernel function K in order to map data points, living in the input space V, to a higher dimensional feature space V', where separability between classes of data becomes clearer. Kernel methods avoid the calculation of points in the new coordinates but rather perform the so called kernel trick that allow to work in the feature space V' simply computing the kernel of pairs of data points [115] (see. Fig.1.2).



Graph Edit Distance

Graph Edit Distance

The **GED between two graphs** is defined as

$$GED(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$$

where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming g_1 into (a graph isomorphic to) g_2 and $c(e) \geq 0$ is the cost of an edit operation e.

Consider a **kernel** for SVM **based on the GED**

$$K(g_1, g_2) = \mathcal{F}(GED(g_1, g_2))$$

The problem of **computing GED** is **NP-hard**.

Quantum Annealing for GED

Quantum Annealing for Graph Edit Distance

Let's try to calculate GED as the output of a combinatorial optimization problem that we can address with Quantum Annealing

Given **two graphs** $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with the **same number of nodes**, we define a binary variable $x_{v,i}$ as follows

$$x_{v,i} = \begin{cases} 1 & \text{if vertex } v \in V_2 \text{ gets mapped to vertex } i \in V_1 \\ 0 & \text{otherwise} \end{cases}$$

Quantum Annealing for Graph Edit Distance

We are now introducing the initial Hamiltonian, i.e. the hard constraints, which must necessarily be satisfied in order to obtain a solution. To guarantee such bijective mapping it is necessary to formulate $\mathcal{H}_A$ as follows:

$$\mathcal{H}_A = A\sum_v (1 - \sum_i x_{v,i})^2 + A\sum_i (1 - \sum_v x_{v,i})^2$$

As said before the number of nodes of the two graphs are the same, in this case $\mathcal{H}_A$ must be zero in the optimal case. At this stage we also need a second term $\mathcal{H}_B$ which penalize a bad mapping i.e. when two vertices i and j in G_1 that are connected by and edge, $(i, j) \in E_1$, are mapped into vertices u and v in G_2 which are not connected, $(u, v) \notin E_2$ and vice versa.

$$\mathcal{H}_B = B \sum_{i,j \notin E_1} \sum_{u,v \in E_2} x_{u,i}x_{v,j} + B \sum_{i,j \in E_1} \sum_{u,v \notin E_2} x_{u,i}x_{v,j}$$

Quantum Annealing for Graph Edit Distance

The complete QUBO formulation is:

$$\begin{aligned}\mathcal{H}_{Qubo} =& A\sum_{v}(1-\sum_{i}x_{v,i})^2 + A\sum_{i}(1-\sum_{v}x_{v,i})^2 + \\ &+ B\sum_{i,j\notin E_1}\sum_{u,v\in E_2}x_{u,i}x_{v,j} + B\sum_{i,j\in E_1}\sum_{u,v\notin E_2}x_{u,i}x_{v,j}\end{aligned} \tag{7.13}$$

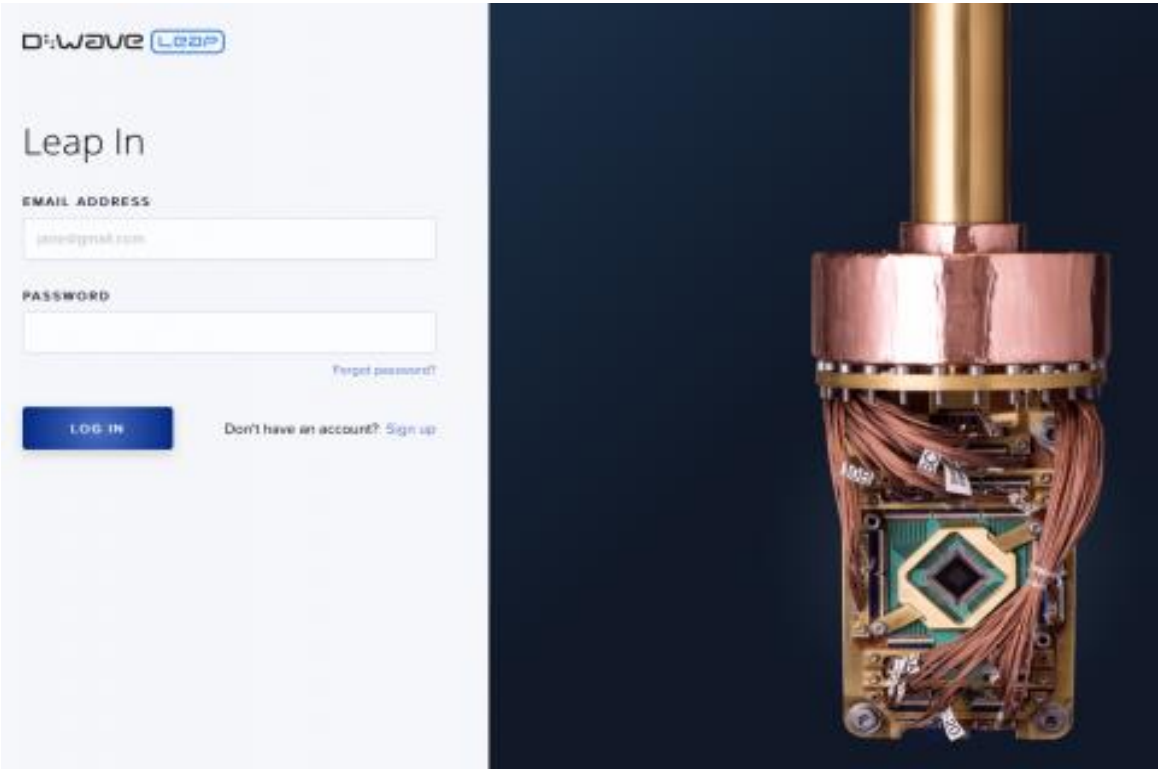
So if the result of Eq. 7.13 is equal to 0, then the two graphs are isomorphic and their GED is 0. If the result is greater than 0, the minimum of $\mathcal{H}_{Qubo}$ will be also the minimum distance between the two graphs, i.e. $\mathcal{H}_{Qubo}$ will return the GED.

Since $\mathcal{H}_A$ is a hard constraint, parameter A should be much higher than B if we want $\mathcal{H}_A$ to be surely satisfied. However, since in the D-Wave quantum annealer all the QUBO coefficients (both linear, h_i, and quadratic, J_{ij}) get normalized in the range $[-1, 1]$, it is important to chose A and B in such a way that the normalized coefficients do not become too small. For this reason, a good choice for parameters A and B is the one where

$$\min_{i,j,k}(h_i^A, J_{jk}^A) \geq \max_{i,j,k}(h_i^B, J_{jk}^B)$$

where h_i^A and J_{jk}^A are respectively the linear and quadratic coefficients appearing in $\mathcal{H}_A$ while h_i^B and J_{jk}^B are linear and quadratic coefficients of $\mathcal{H}_B$.

Try on the real D-Wave hardware



D-Wave access:
https://cloud.dwavesys.com/leap/
(free 1 min / month)

Solver Name	Status	Description
Hybrid		
hybrid_binary_quadratic_model_version2	Online	Hybrid solver for general BQM problems, version 2.0
hybrid_discrete_quadratic_model_version1	Online	Hybrid solver for general DQM problems, version 1.0
QPU		
Advantage_system1.1	Online	Advantage system
DW_2000Q_6	Online	D-Wave 2000Q lower-noise system

D-Wave Software Documentation:
https://docs.ocean.dwavesys.com/en/stable/index.html

Quantum Computing @ CINECA

CINECA: Italian HPC center

CINECA Quantum Computing Lab:

- Research with Universities, Industries and QC startups

- Internship programs, Courses and Conference (HPCQC)

https://www.quantumcomputinglab.cineca.it

r.mengoni@cineca.it