# INTRODUCTION TO QUANTUM ANNEALING

Formulating and solve QUBO Problems

*Daniele Ottaviani*

CINECA QUANTUM COMPUTING LAB

# The Quantum Annealing Algorithm

- In today's lesson we will study and learn how to program a particular type of quantum computer, the Quantum Annealer

# The Quantum Annealing Algorithm

- In today's lesson we will study and learn how to program a particular type of quantum computer, the **Quantum Annealer**
- A Quantum Annealer is a **special purpose quantum computer**

# The Quantum Annealing Algorithm

- In today's lesson we will study and learn how to program a particular type of quantum computer, the **Quantum Annealer**
- A Quantum Annealer is a **special purpose quantum computer**
- Its purpose, unlike the quantum computers seen in the previous lessons, called General Purpose Quantum Computers, **is not to be freely programmed** by the user.



2

# The Quantum Annealing Algorithm

- In today's lesson we will study and learn how to program a particular type of quantum computer, the **Quantum Annealer**
- A Quantum Annealer is a **special purpose quantum computer**
- Its purpose, unlike the quantum computers seen in the previous lessons, called General Purpose Quantum Computers, **is not to be freely programmed** by the user.
- A Quantum Annealer is a quantum computer designed and built **to host a single quantum algorithm, Quantum Annealing**

# The Quantum Annealing Algorithm

- In today's lesson we will study and learn how to program a particular type of quantum computer, the **Quantum Annealer**
- A Quantum Annealer is a **special purpose quantum computer**
- Its purpose, unlike the quantum computers seen in the previous lessons, called General Purpose Quantum Computers, **is not to be freely programmed** by the user.
- A Quantum Annealer is a quantum computer designed and built **to host a single quantum algorithm, Quantum Annealing**
- **Quantum Annealing is a quantum algorithm capable of solving optimization problems**



2

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in 1998 with the paper you see in the figure.

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

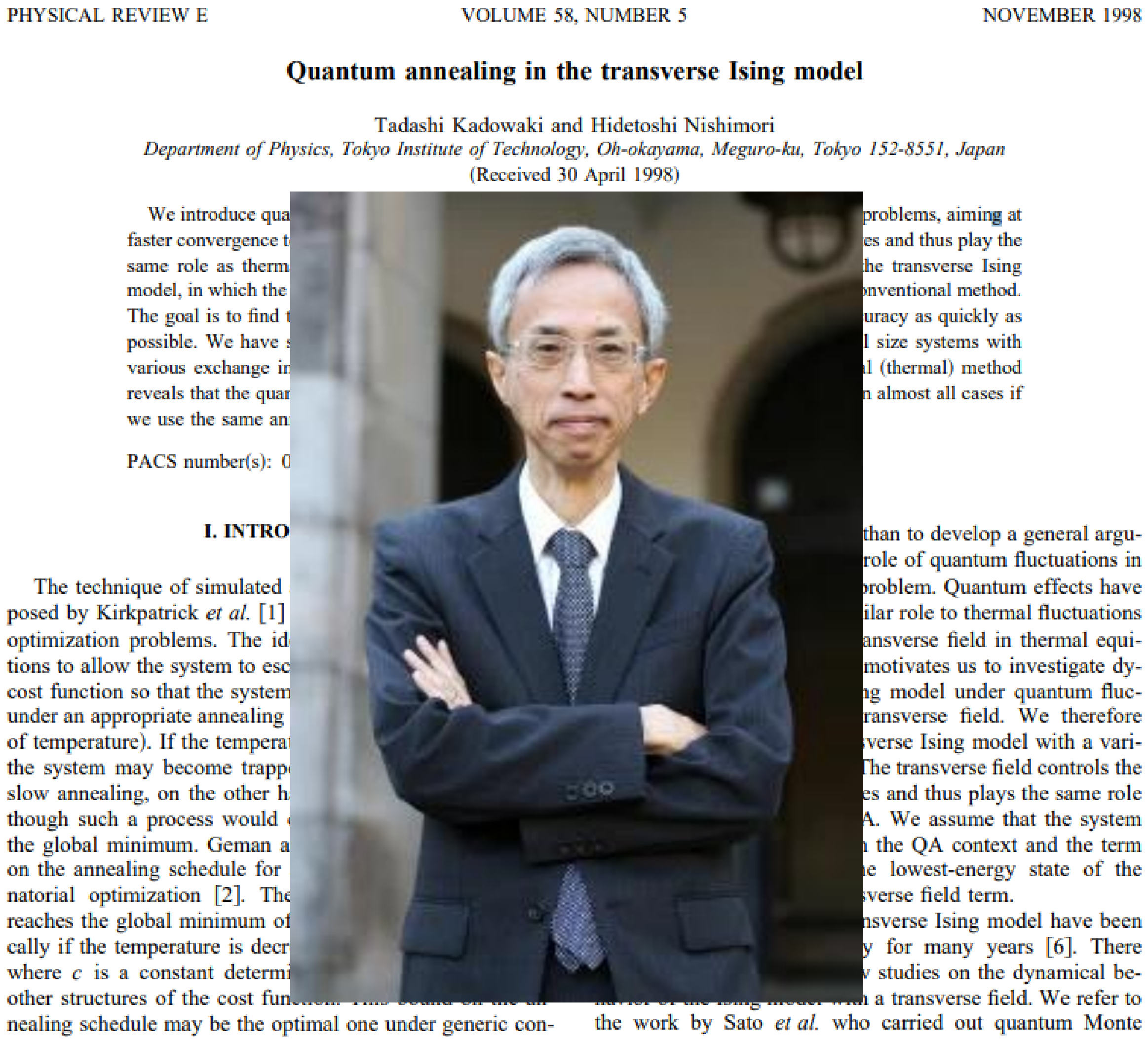
The technique of simulated annealing (SA) was first proposed by Kirkpatrick *et al.* [1] as a general method to solve optimization problems. The idea is to use thermal fluctuations to allow the system to escape from local minima of the cost function so that the system reaches the global minimum under 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 con-

specific 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

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**


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 qua … problems, aiming at faster convergence t … es and thus play the same role as therm … he transverse Ising model, in which the … onventional method. The goal is to find t … uracy as quickly as possible. We have s … l size systems with various exchange in … l (thermal) method reveals that the quan … n almost all cases if we use the same an…

PACS number(s): 0…

I. INTRO…

The technique of simulated … posed by Kirkpatrick *et al.* [1] … optimization problems. The id… tions to allow the system to esc… cost function so that the system … under an appropriate annealing … of temperature). If the tempera… the system may become trapp… slow annealing, on the other h… though such a process would … the global minimum. Geman a… on the annealing schedule for … natorial optimization [2]. The … reaches the global minimum of … cally if the temperature is decr… where *c* is a constant determi… other structures of the cost fun… nealing schedule may be the optimal one under generic con-

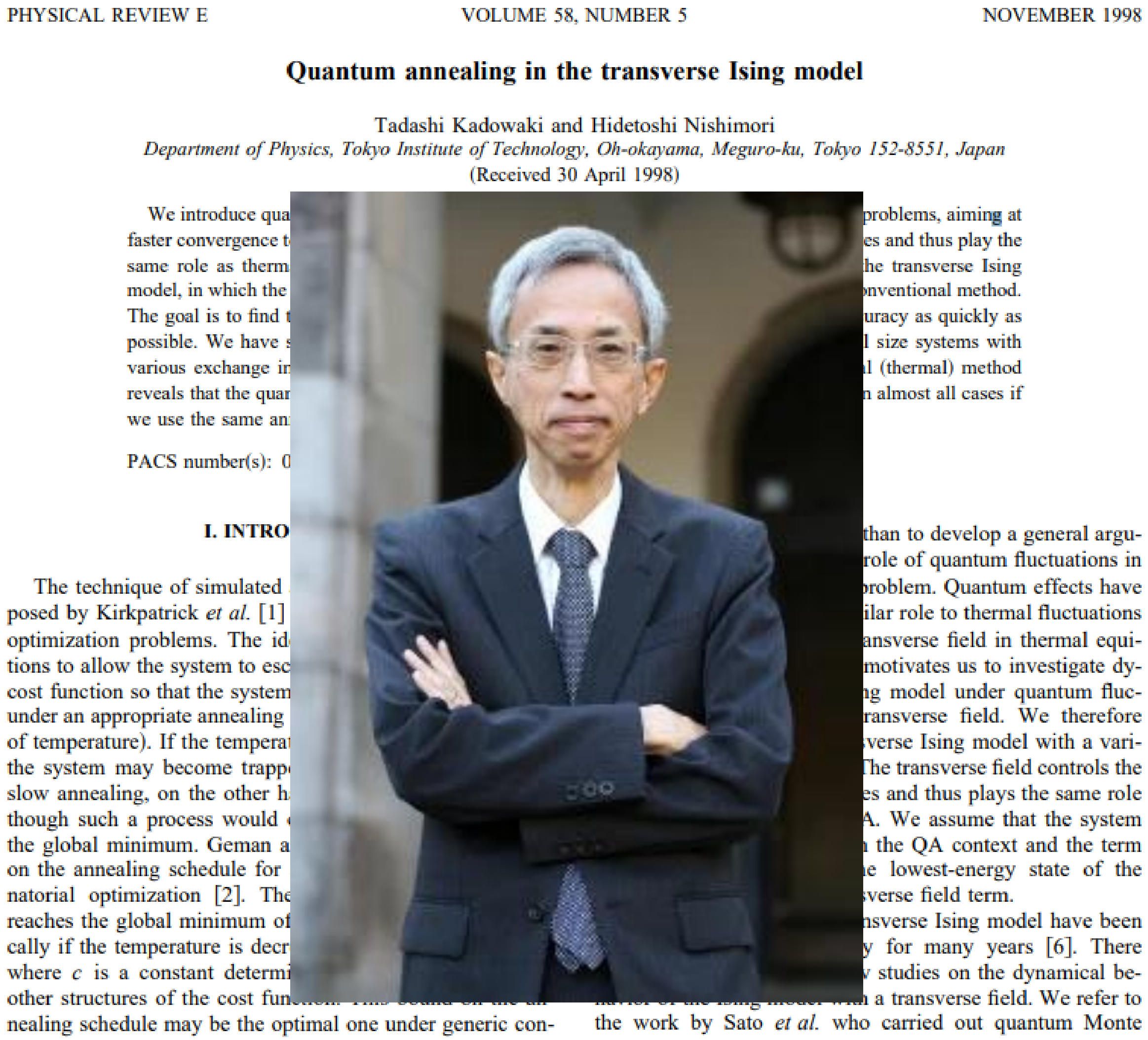
… than to develop a general argu… role of quantum fluctuations in … problem. Quantum effects have … ilar role to thermal fluctuations … ansverse field in thermal equi… motivates us to investigate dy… ng model under quantum fluc… ransverse field. We therefore … verse Ising model with a vari… The transverse field controls the … es and thus plays the same role … A. We assume that the system … n the QA context and the term … e lowest-energy state of the … verse field term. … nsverse Ising model have been … y for many years [6]. There … studies on the dynamical be… a transverse field. We refer to the work by Sato *et al.* who carried out quantum Monte

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**
- Professor Nishimori, now happily retired, used to work as a full professor at the University of Tokyo

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**
- Professor Nishimori, now happily retired, used to work as a full professor at the University of Tokyo
- His studies in this field have opened a **real alternative path** for quantum computing



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 qua… problems, aiming at faster convergence t… es and thus play the same role as therm… he transverse Ising model, in which the … nventional method. The goal is to find t… uracy as quickly as possible. We have s… l size systems with various exchange in… l (thermal) method reveals that the quan… n almost all cases if we use the same an…

PACS number(s): 0…

I. INTRO…

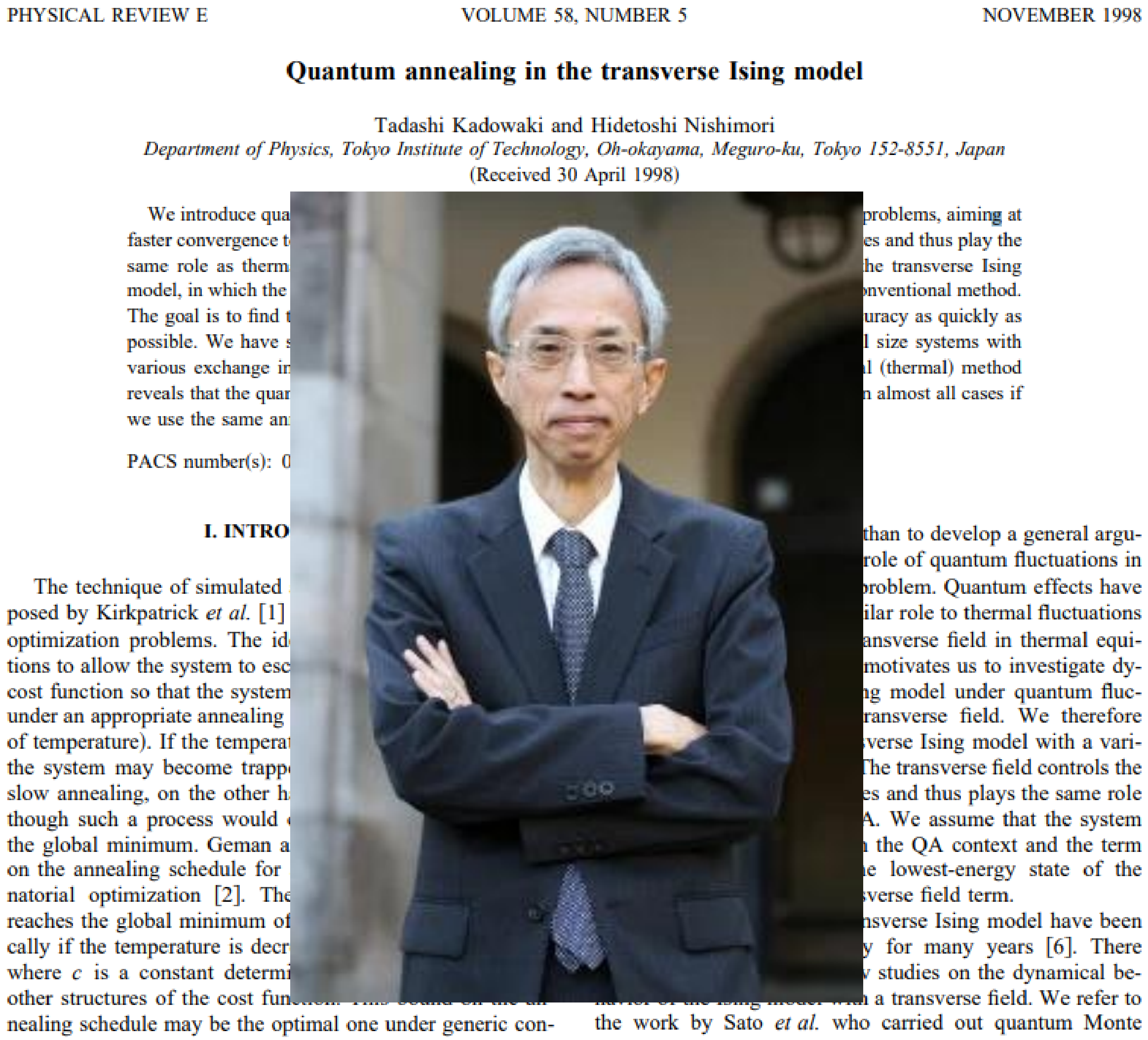
The technique of simulated … posed by Kirkpatrick *et al.* [1] … optimization problems. The id… tions to allow the system to esc… cost function so that the system … under an appropriate annealing … of temperature). If the tempera… the system may become trapp… slow annealing, on the other h… though such a process would … the global minimum. Geman a… on the annealing schedule for … natorial optimization [2]. The … reaches the global minimum of … cally if the temperature is decr… where $c$ is a constant determi… other structures of the cost fun… nealing schedule may be the optimal one under generic con-

… than to develop a general argu… role of quantum fluctuations in … problem. Quantum effects have … ilar role to thermal fluctuations … ansverse field in thermal equi… motivates us to investigate dy… ng model under quantum fluc… ransverse field. We therefore … verse Ising model with a vari… The transverse field controls the … es and thus plays the same role … A. We assume that the system … the QA context and the term … e lowest-energy state of the … verse field term. … nsverse Ising model have been … y for many years [6]. There … studies on the dynamical be… … a transverse field. We refer to the work by Sato *et al.* who carried out quantum Monte

CINECA QUANTUM COMPUTING LAB

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**
- Professor Nishimori, now happily retired, used to work as a full professor at the University of Tokyo
- His studies in this field have opened a **real alternative path** for quantum computing
- 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!**



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 qua problems, aiming at faster convergence t es and thus play the same role as therm he transverse Ising model, in which the onventional method. The goal is to find t uracy as quickly as possible. We have s l size systems with various exchange in l (thermal) method reveals that the quan n almost all cases if we use the same an

PACS number(s): 0

I. INTRO

The technique of simulated posed by Kirkpatrick *et al.* [1] optimization problems. The id tions to allow the system to esc cost function so that the system under an appropriate annealing of temperature). If the temperat the system may become trapp slow annealing, on the other h though such a process would the global minimum. Geman a on the annealing schedule for natorial optimization [2]. The reaches the global minimum of cally if the temperature is decr where $c$ is a constant determi other structures of the cost fun nealing schedule may be the optimal one under generic con-

than to develop a general argu- role of quantum fluctuations in problem. Quantum effects have ilar role to thermal fluctuations ansverse field in thermal equi- motivates us to investigate dy- ng model under quantum fluc- ransverse field. We therefore verse Ising model with a vari- The transverse field controls the es and thus plays the same role A. We assume that the system n the QA context and the term e lowest-energy state of the verse field term. nsverse Ising model have been y for many years [6]. There studies on the dynamical be- a transverse field. We refer to the work by Sato *et al.* who carried out quantum Monte

CINECA QUANTUM COMPUTING LAB

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**
- Professor Nishimori, now happily retired, used to work as a full professor at the University of Tokyo
- His studies in this field have opened a **real alternative path** for quantum computing
- 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!**
- The first Quantum Annealer model from **D-Wave**, in fact, came out 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 qua … problems, aiming at faster convergence t … es and thus play the same role as therm … he transverse Ising model, in which the … onventional method. The goal is to find t … uracy as quickly as possible. We have s … l size systems with various exchange in … l (thermal) method reveals that the quan … n almost all cases if we use the same an …

PACS number(s): 0 …

I. INTRO …

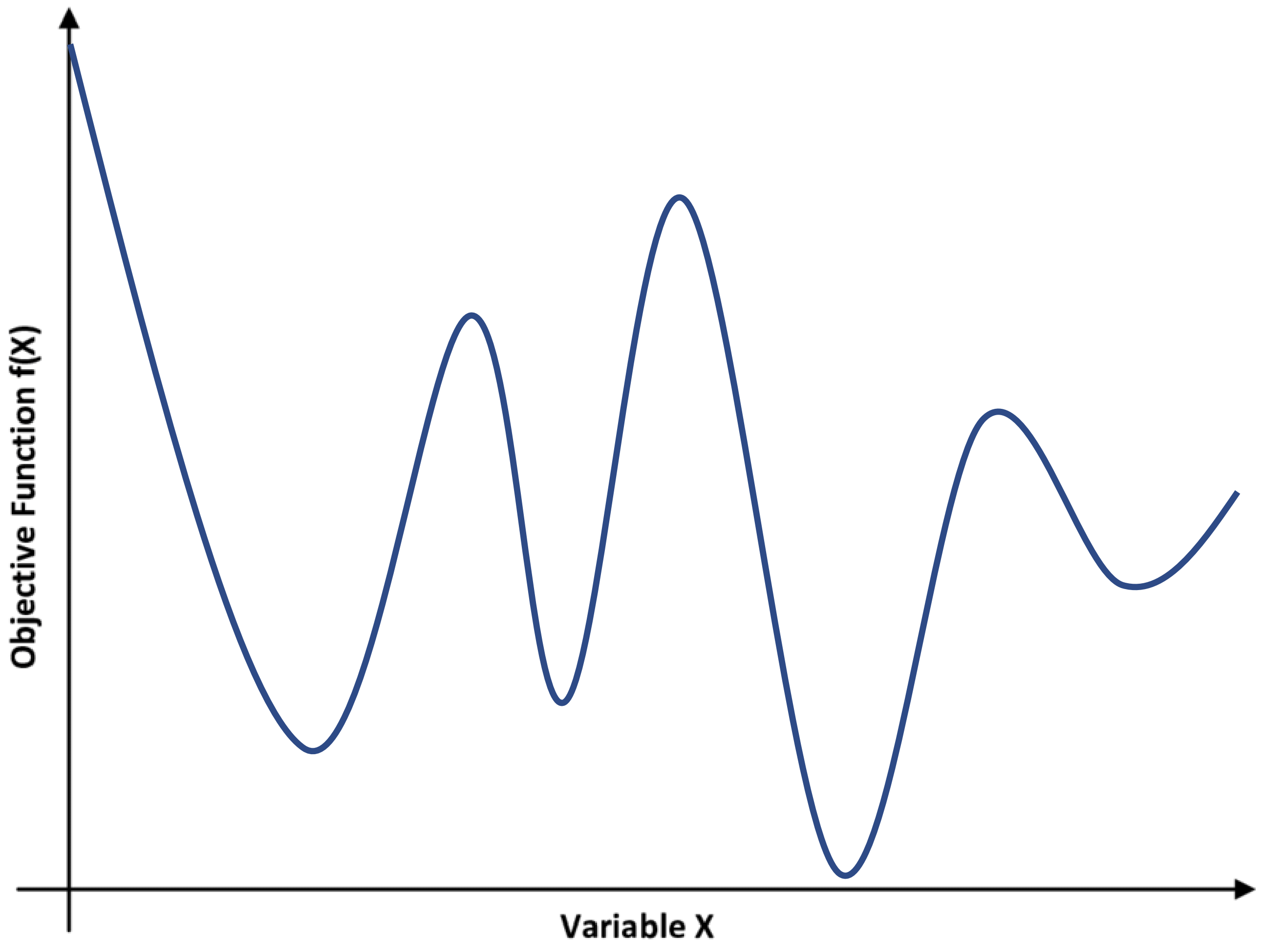
The technique of simulated … than to develop a general argu- posed by Kirkpatrick *et al.* [1] … role of quantum fluctuations in optimization problems. The id … problem. Quantum effects have tions to allow the system to esc … ilar role to thermal fluctuations cost function so that the system … ansverse field in thermal equi- under an appropriate annealing … motivates us to investigate dy- of temperature). If the temperat … ng model under quantum fluc- the system may become trapp … ransverse field. We therefore slow annealing, on the other h … verse Ising model with a vari- though such a process would … The transverse field controls the the global minimum. Geman a … es and thus plays the same role on the annealing schedule for … A. We assume that the system natorial optimization [2]. The … n the QA context and the term reaches the global minimum of … e lowest-energy state of the cally if the temperature is decr … sverse field term. where $c$ is a constant determi … nsverse Ising model have been other structures of the cost fun … y for many years [6]. There nealing schedule may be the optimal one under generic con- … studies on the dynamical be- … a transverse field. We refer to the work by Sato *et al.* who carried out quantum Monte

# The Quantum Annealing Algorithm

- The quantum annealing algorithm was proposed for the first time in **1998** with the paper you see in the figure.
- The author of the paper, as well as the theorist of the algorithm, is **Professor Hidetoshi Nishimori**
- Professor Nishimori, now happily retired, used to work as a full professor at the University of Tokyo
- His studies in this field have opened a **real alternative path** for quantum computing
- 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!**
- The first Quantum Annealer model from **D-Wave**, in fact, came out in **2012**
- **In 2018 I had a beer with him!**

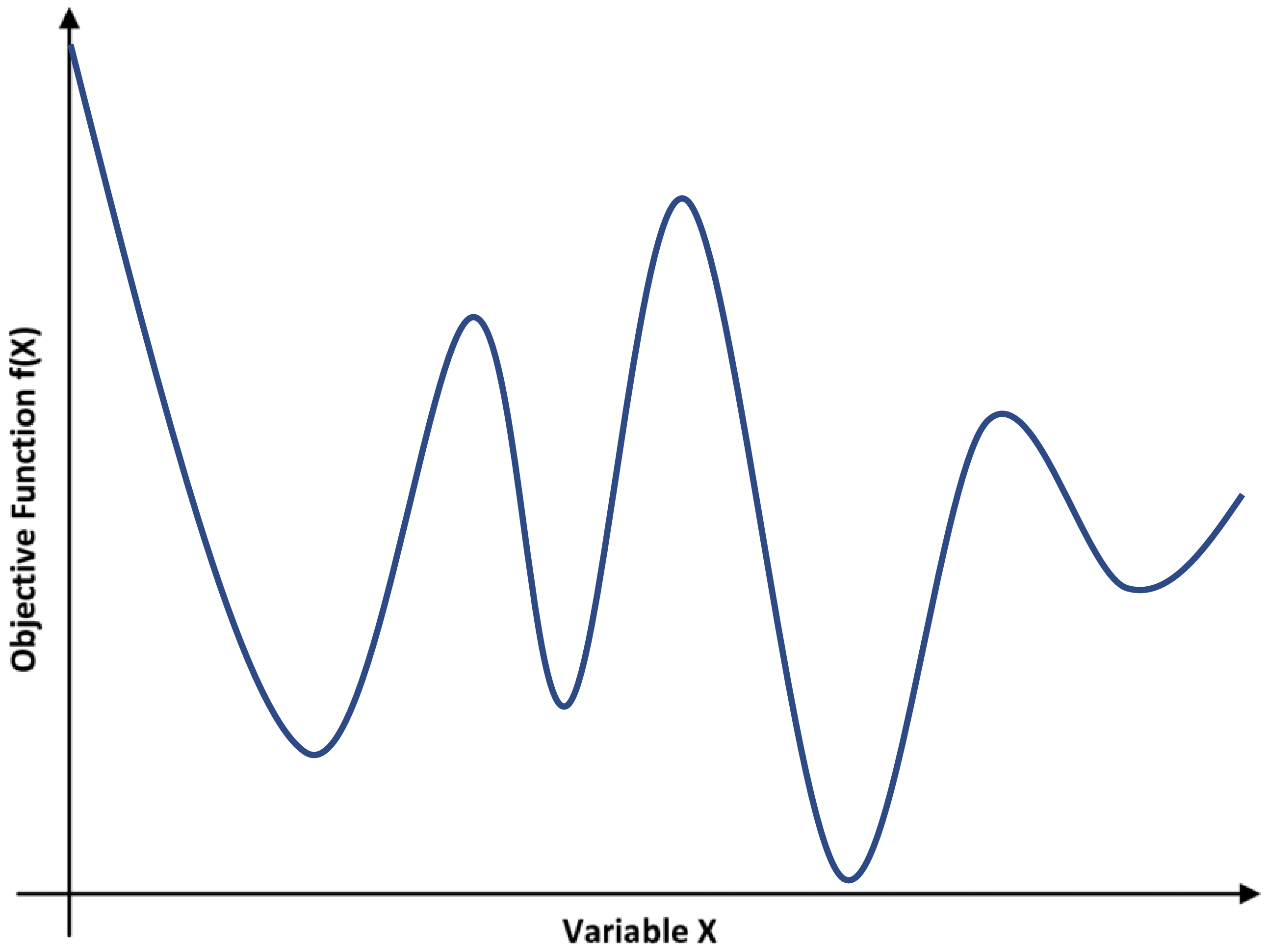
# Annealing Algorithms

- Suppose we have an optimization problem, for example a minimization problem, whose objective function (i.e. the function to be minimized) is known and computable using a finite set of variables.



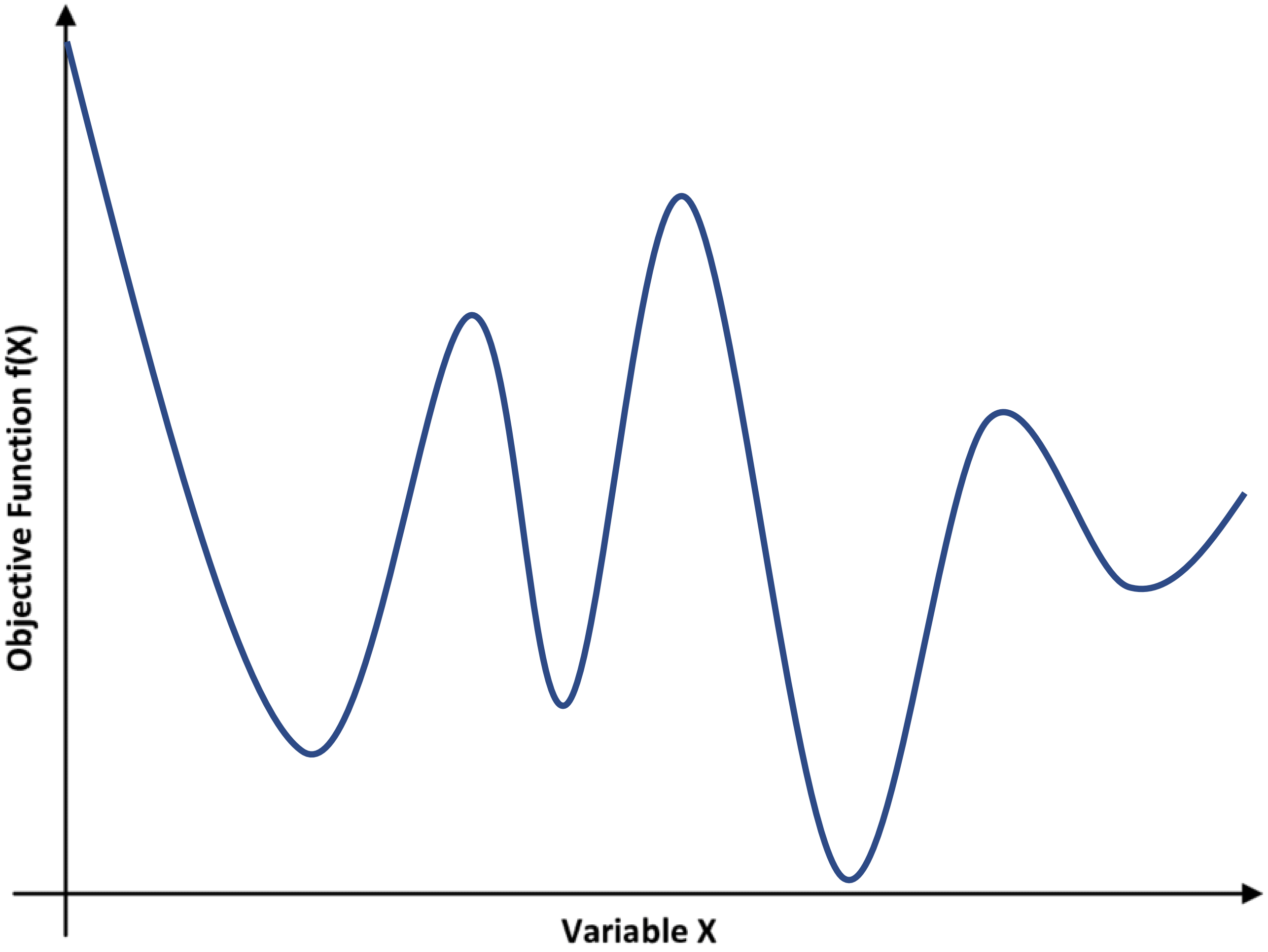
# Annealing Algorithms

- Suppose we have an optimization problem, for example a minimization problem, whose objective function (i.e. the function to be minimized) is known and computable using a finite set of variables.
- The best way to solve a problem of this type is undoubtedly the so-called **brute force approach**: we calculate all the values of the objective function for all possible inputs and consider the smallest



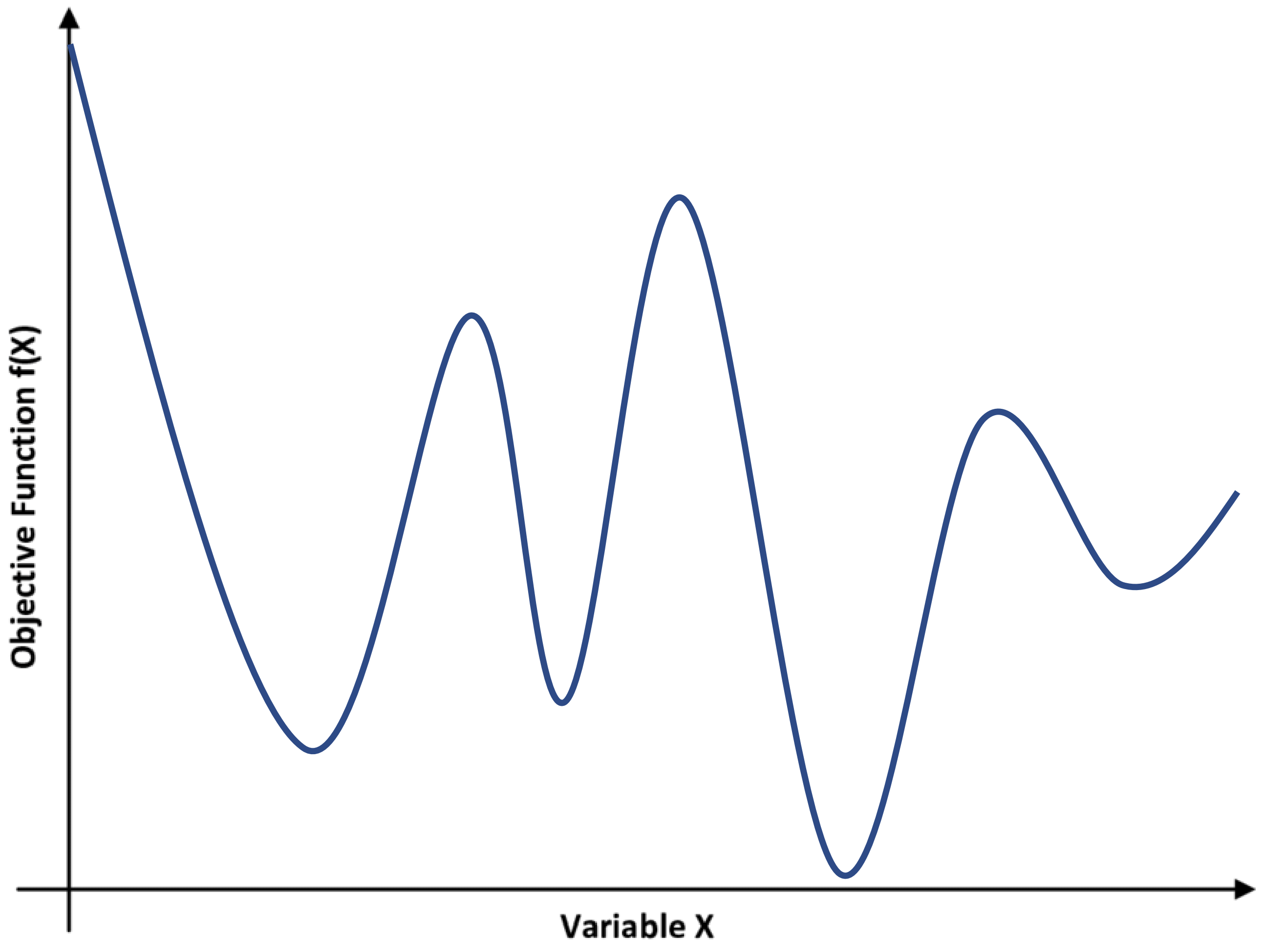
# Annealing Algorithms

- Suppose we have an optimization problem, for example a minimization problem, whose objective function (i.e. the function to be minimized) is known and computable using a finite set of variables.
- The best way to solve a problem of this type is undoubtedly the so-called **brute force approach**: we calculate all the values of the objective function for all possible inputs and consider the smallest
- This approach, although undeniably functional, is **unfortunately not always practicable**. Sometimes the inputs with which to calculate the value of the objective function are not few ...



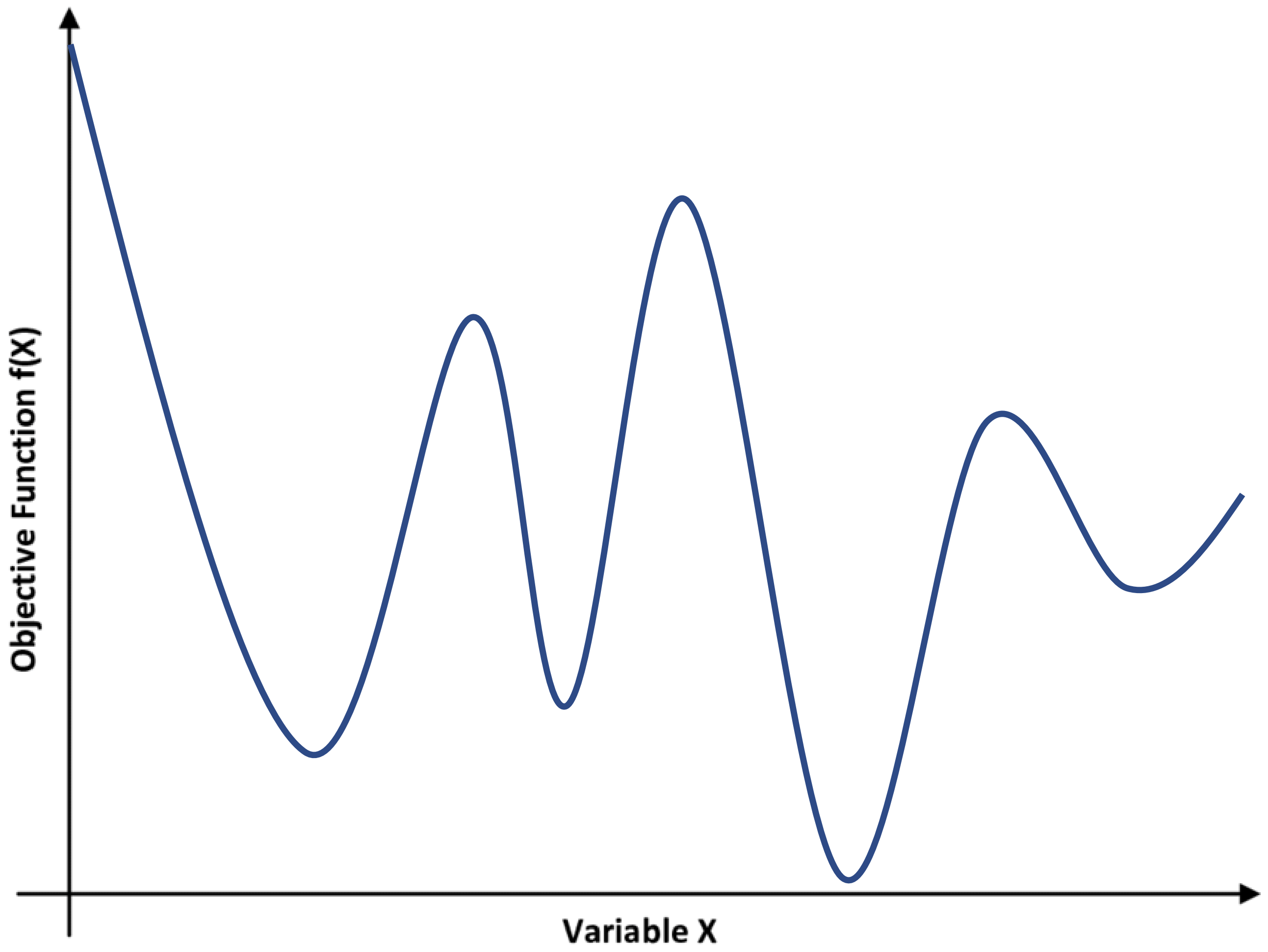
# Annealing Algorithms

- Suppose we have an optimization problem, for example a minimization problem, whose objective function (i.e. the function to be minimized) is known and computable using a finite set of variables.
- The best way to solve a problem of this type is undoubtedly the so-called **brute force approach**: we calculate all the values of the objective function for all possible inputs and consider the smallest
- This approach, although undeniably functional, is **unfortunately not always practicable**. Sometimes the inputs with which to calculate the value of the objective function are not few ...
- Let's imagine for example the case of a function with **N** binary variables: the number of possible combinations is $2^N$...



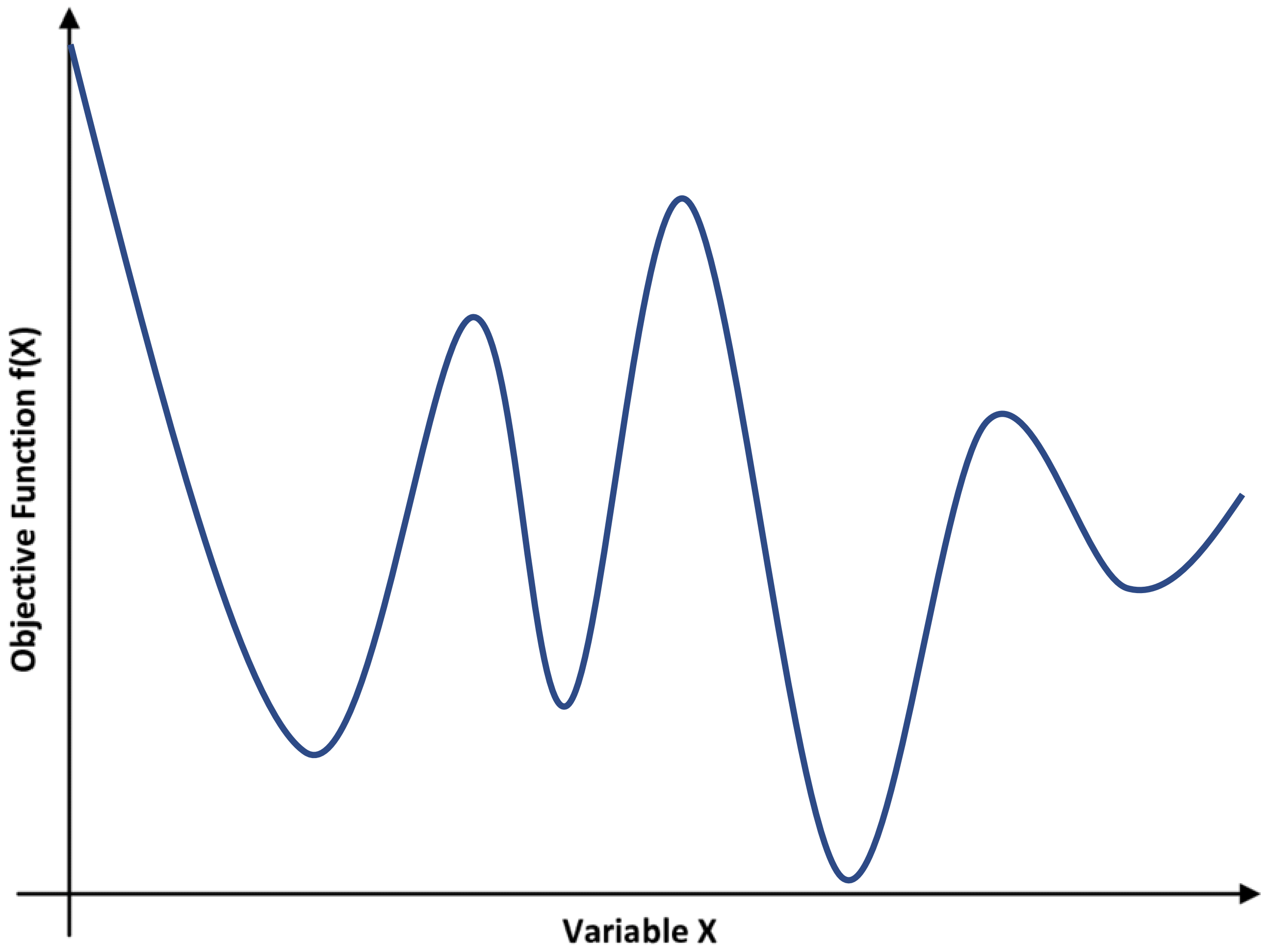
# Annealing Algorithms

- Fortunately, brute force is **not** the only algorithm for solving problems of this type.



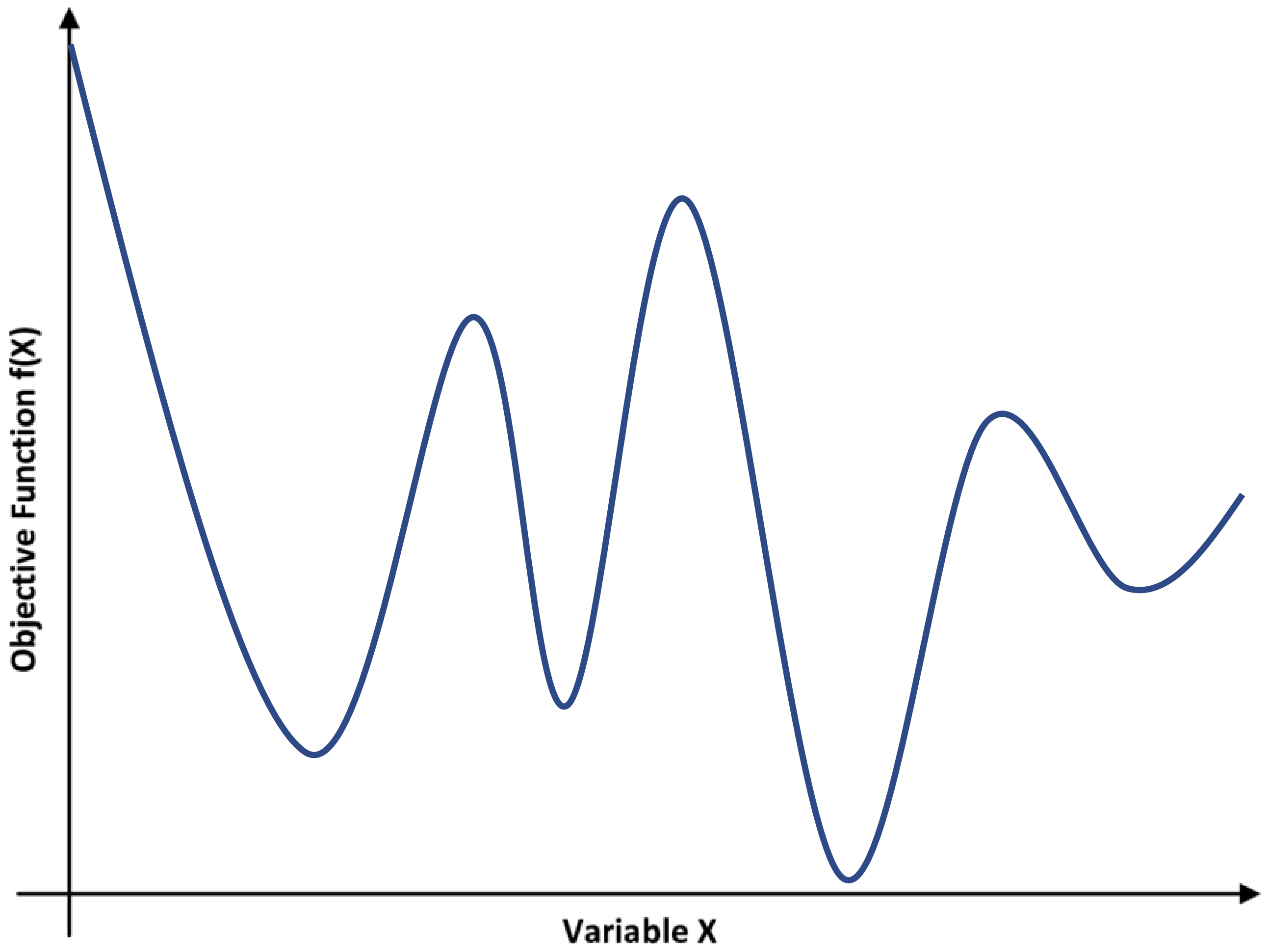
# Annealing Algorithms

- 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



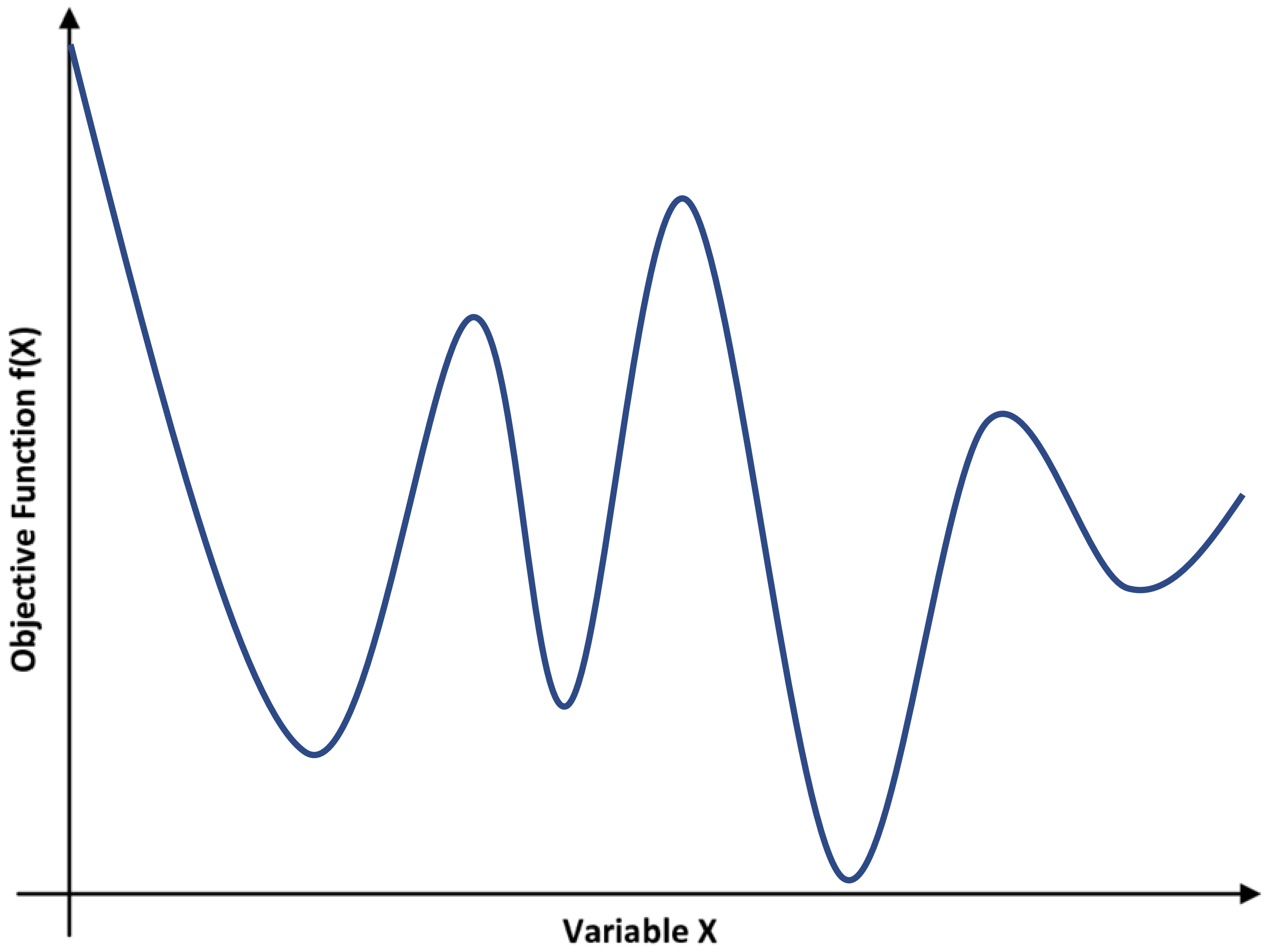
# Annealing Algorithms

- 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



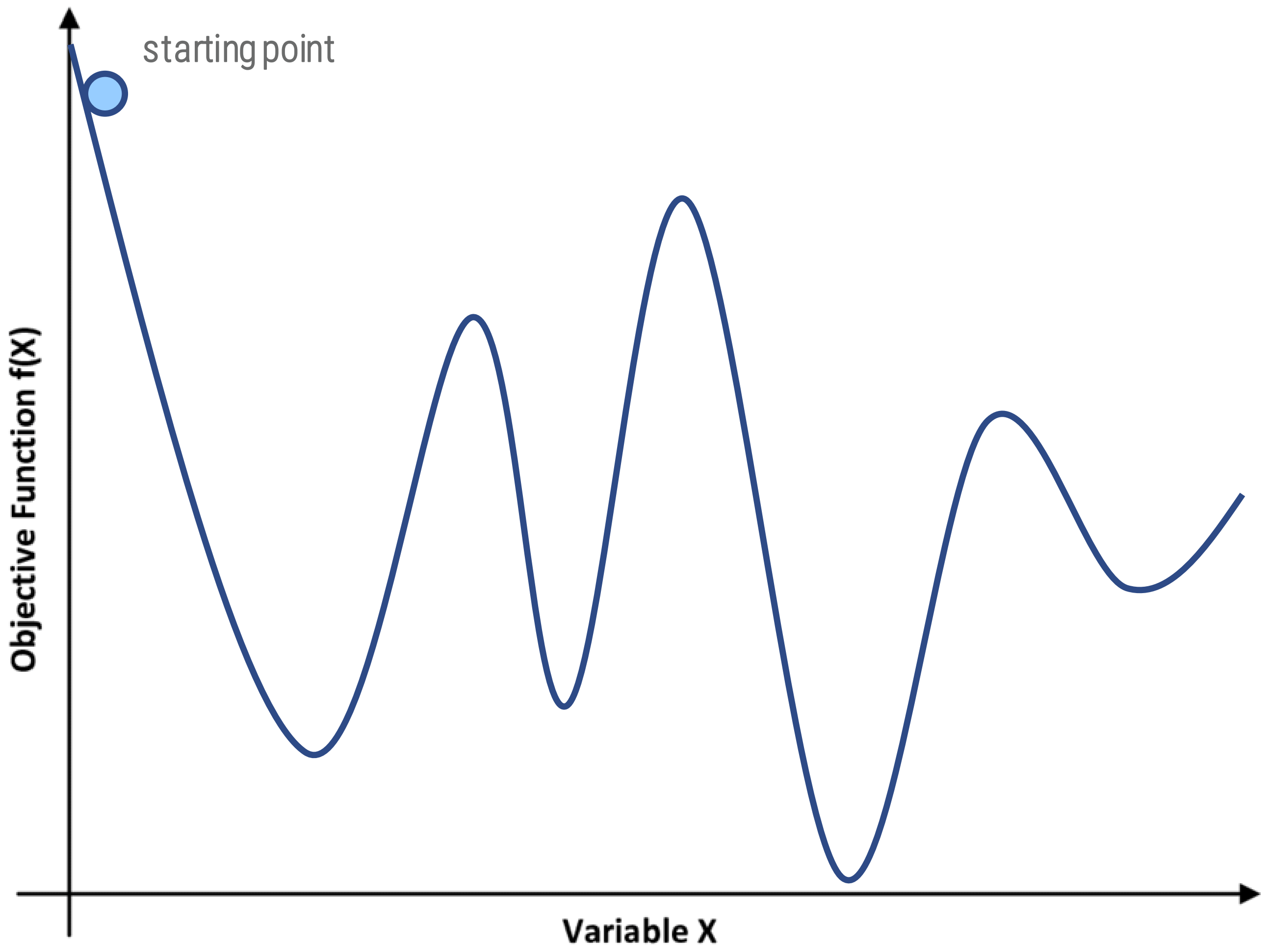
# Annealing Algorithms

- 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



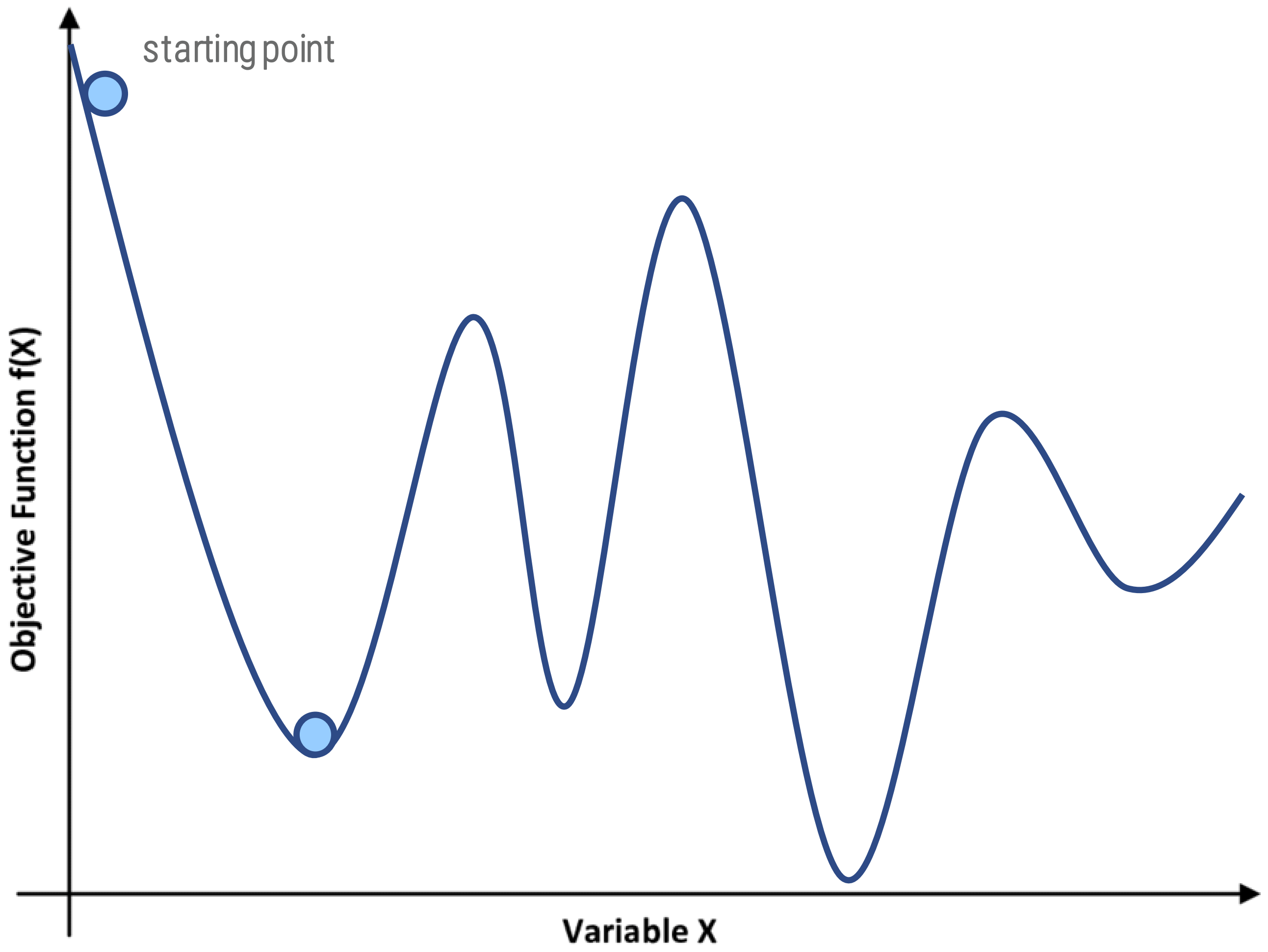
# Annealing Algorithms

- 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.**



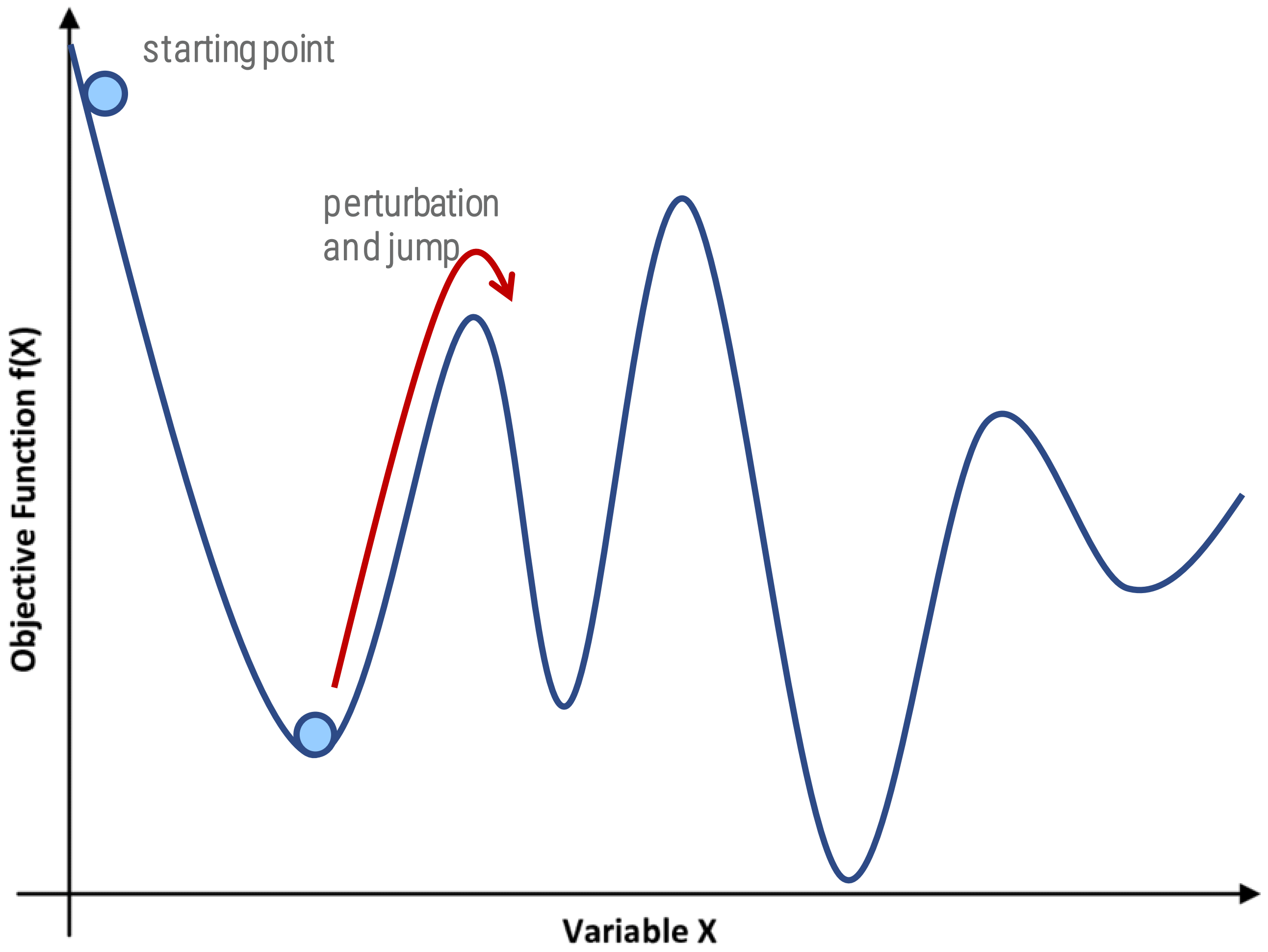
# Annealing Algorithms

- 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.**



# Annealing Algorithms

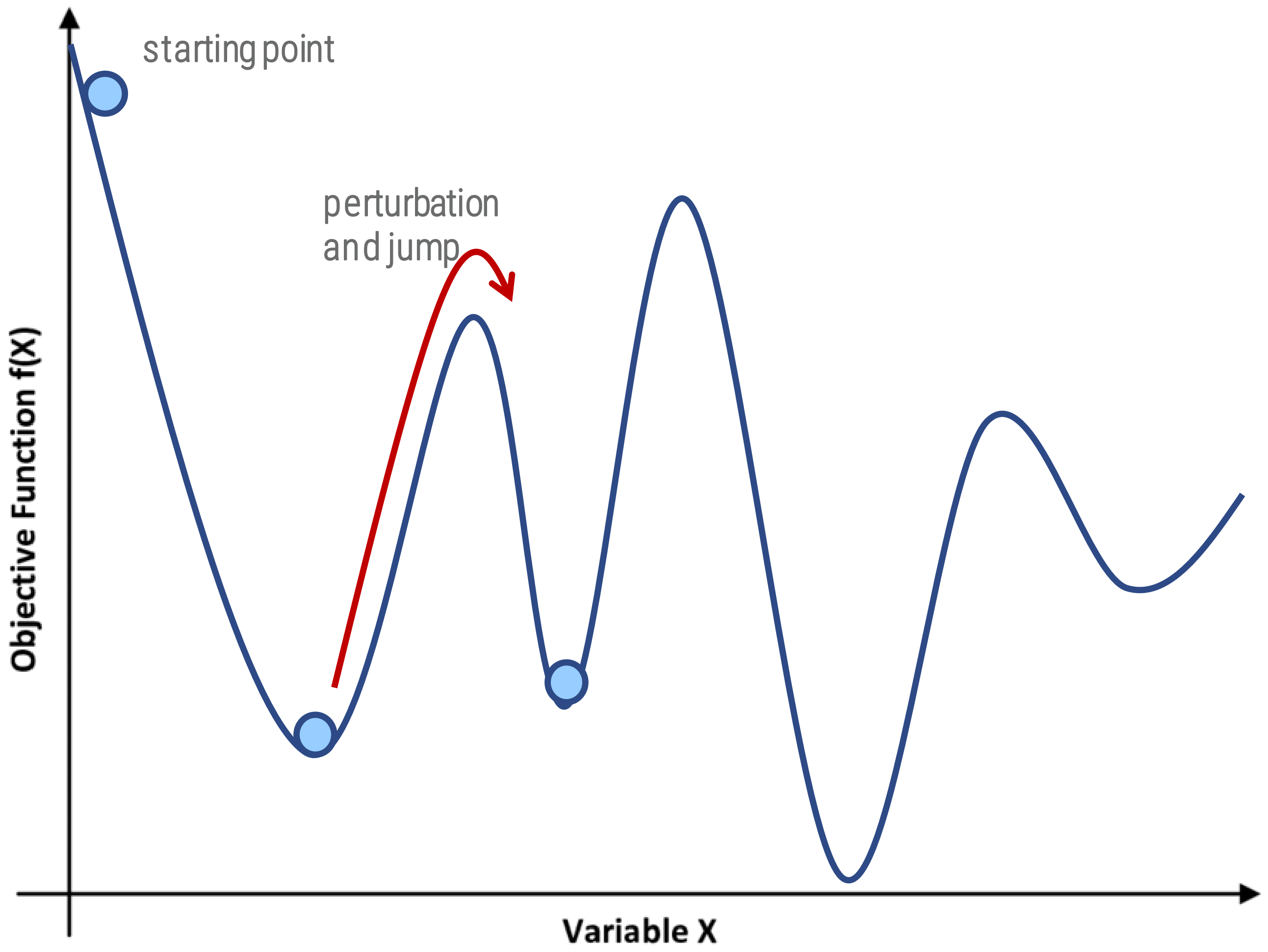
- Fortunately, brute force is **not** the only algorithm for solving problems of this type.
- There are many algorithms capable of identifying the optimal point of an objective function without having to analyze each of its points
- One of these is known as **Simulated Annealing**
- Simulated annealing 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.




CINECA QUANTUM COMPUTING LAB

# Annealing Algorithms

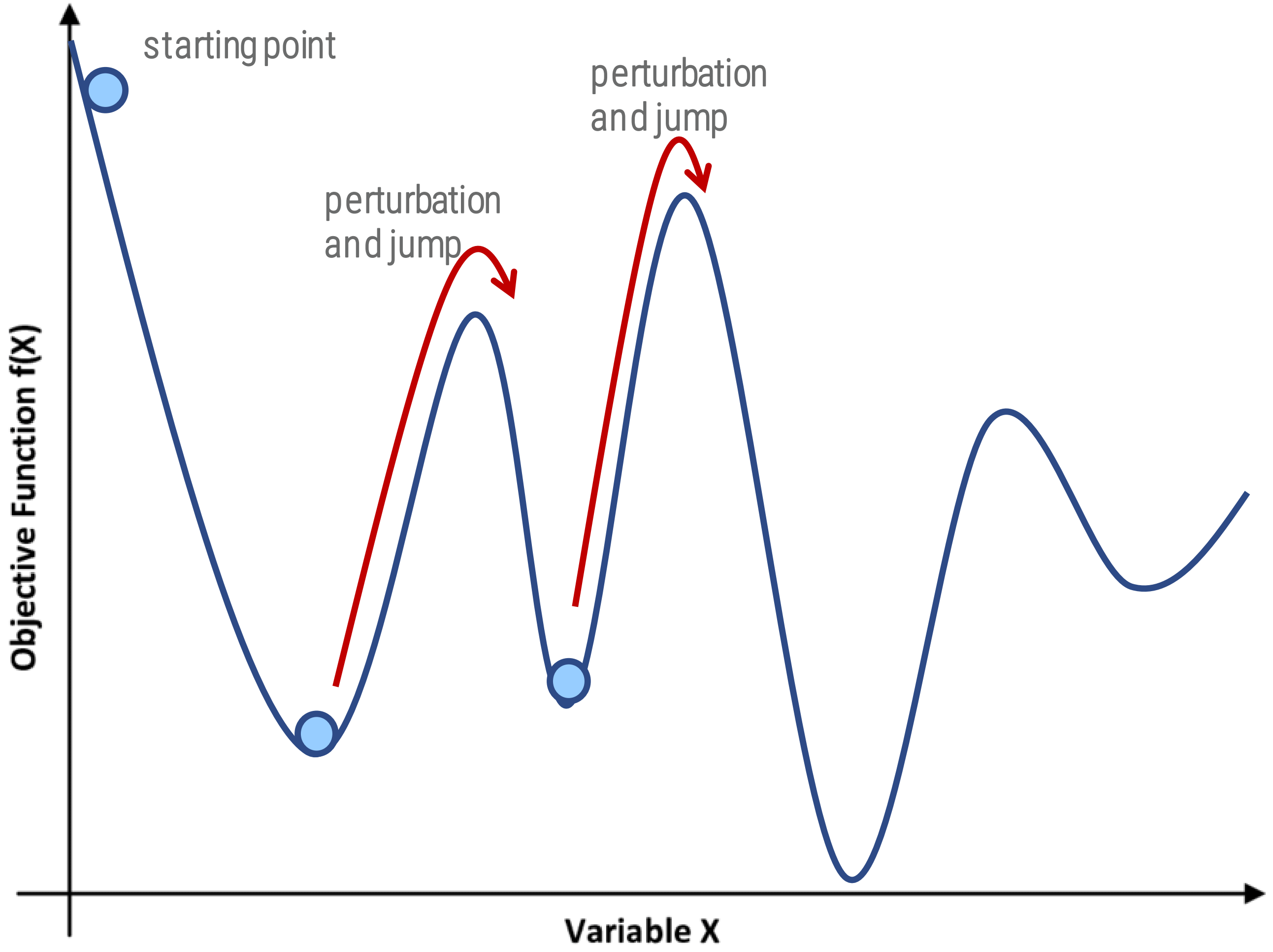
- Fortunately, brute force is **not** the only algorithm for solving problems of this type.
- There are many algorithms capable of identifying the optimal point of an objective function without having to analyze each of its points
- One of these is known as **Simulated Annealing**
- Simulated annealing 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.




CINECA QUANTUM COMPUTING LAB

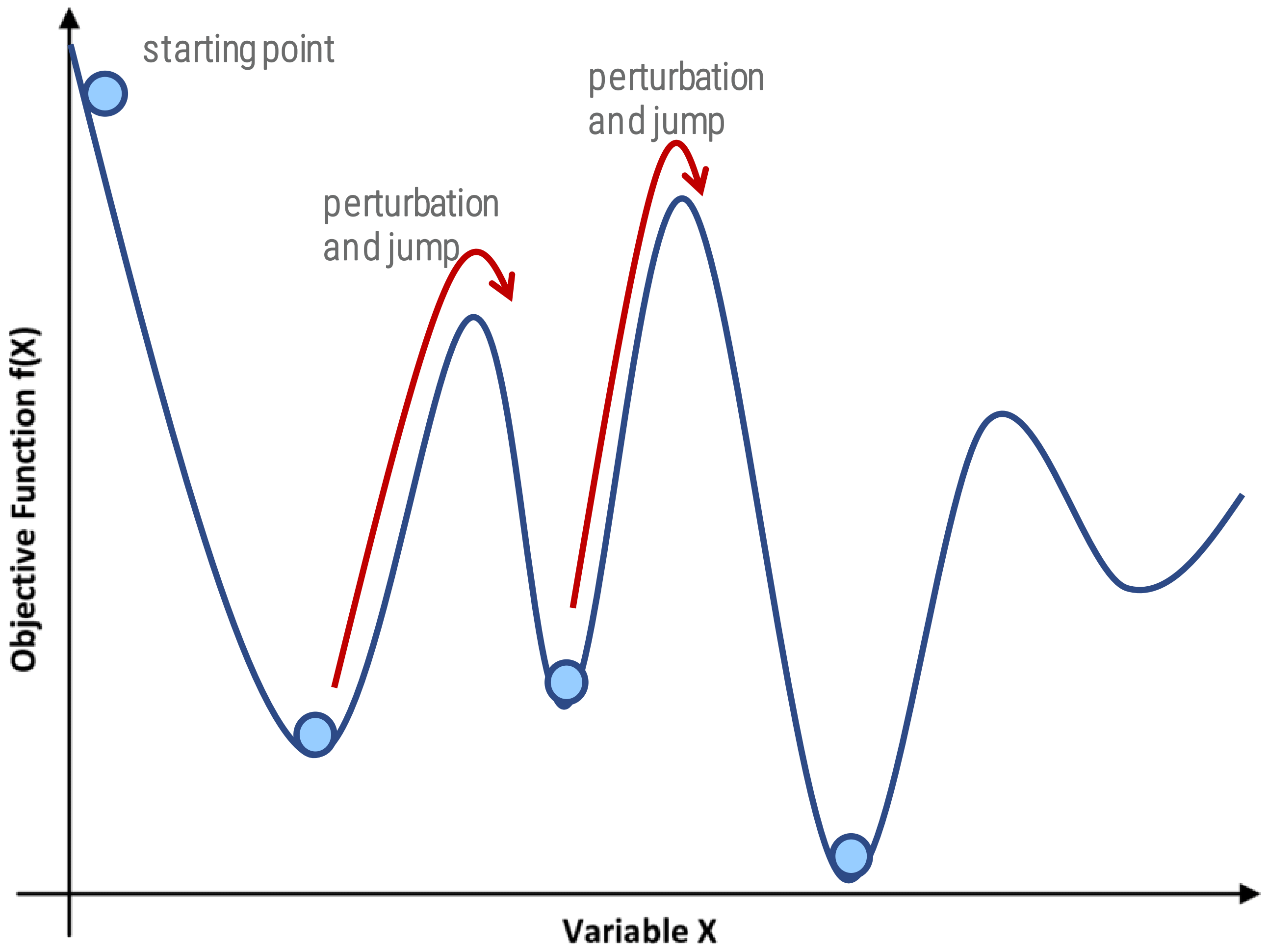
# Annealing Algorithms

- 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.



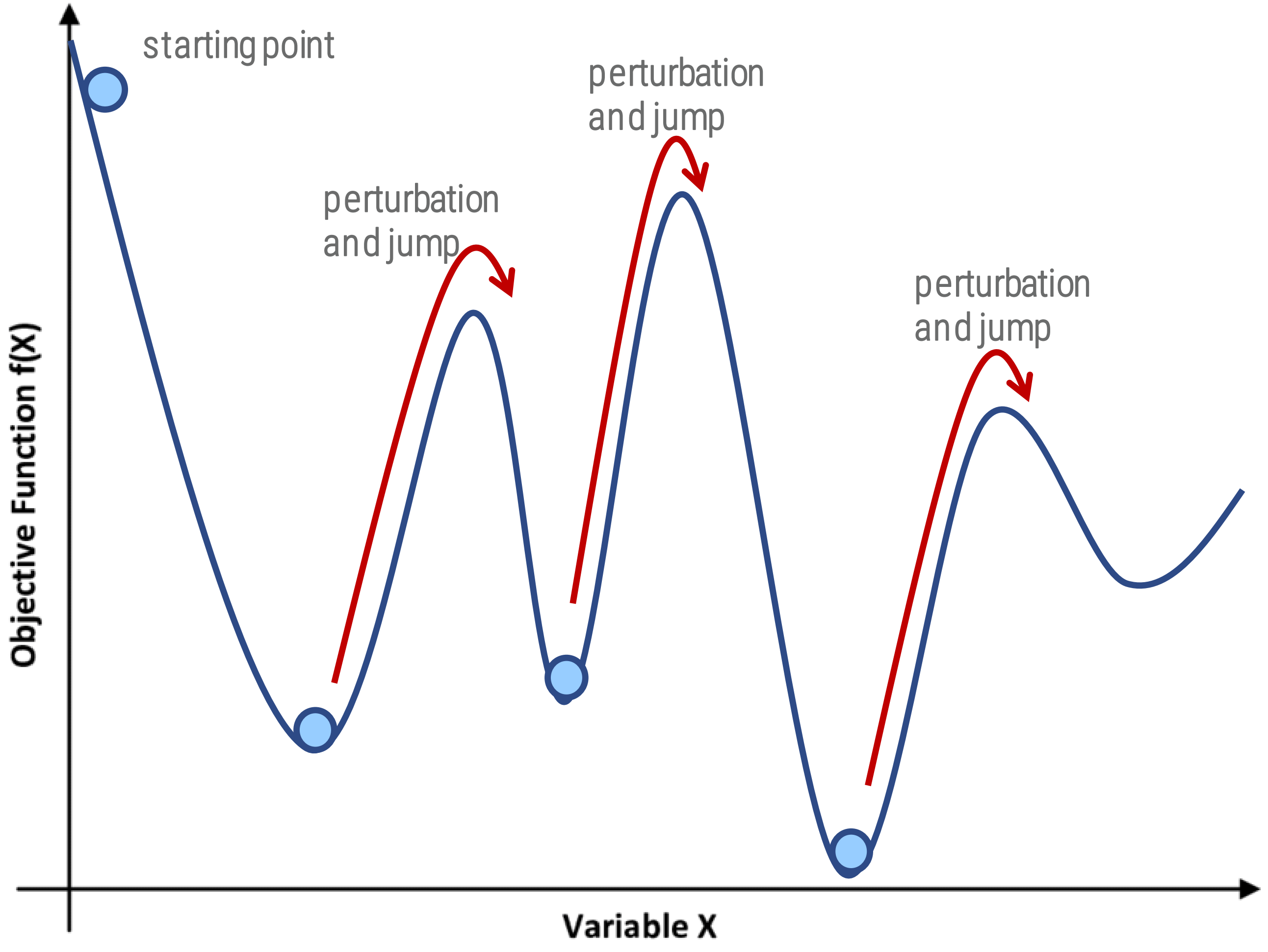
# Annealing Algorithms

- 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.



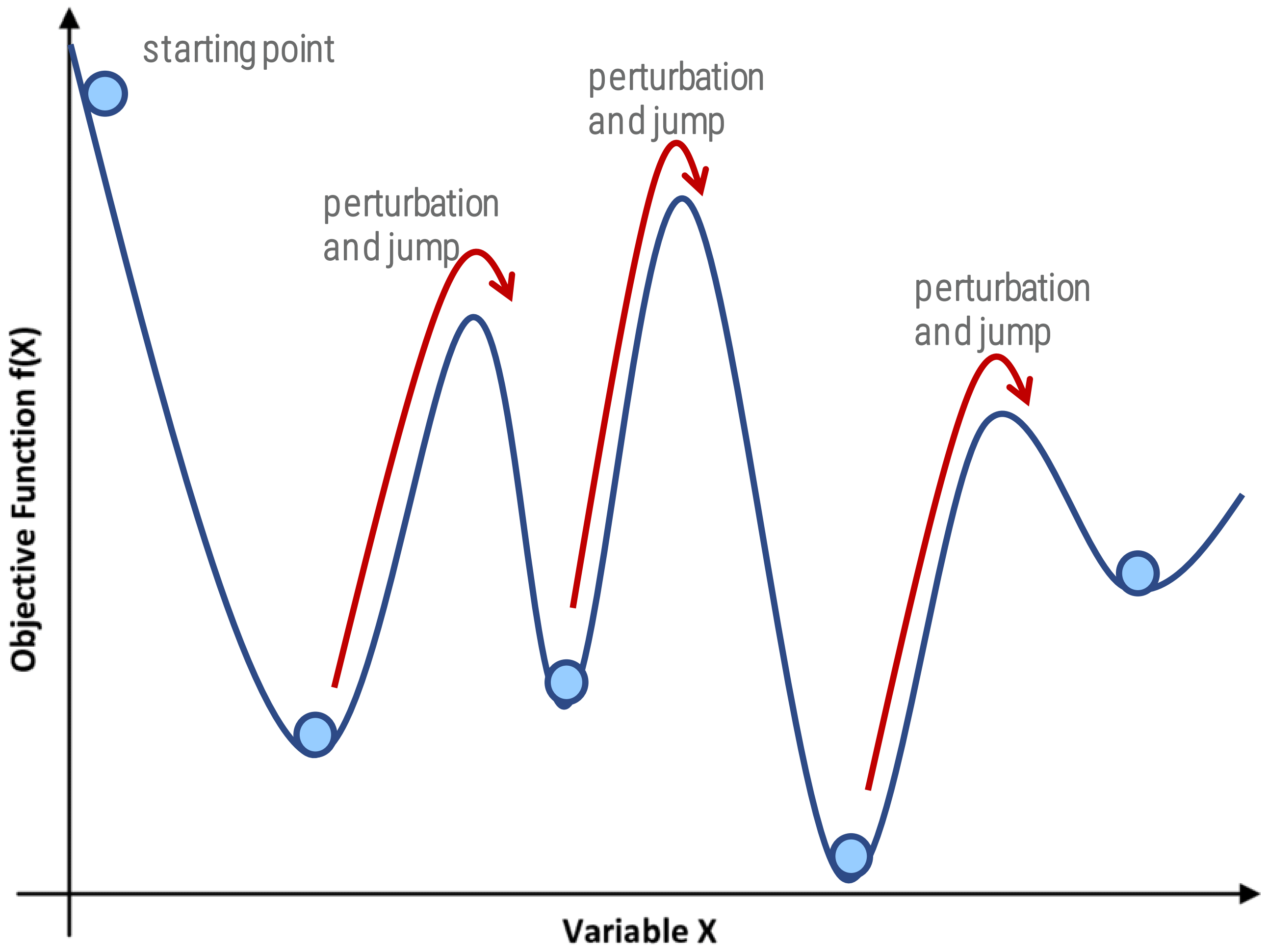
# Annealing Algorithms

- 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.



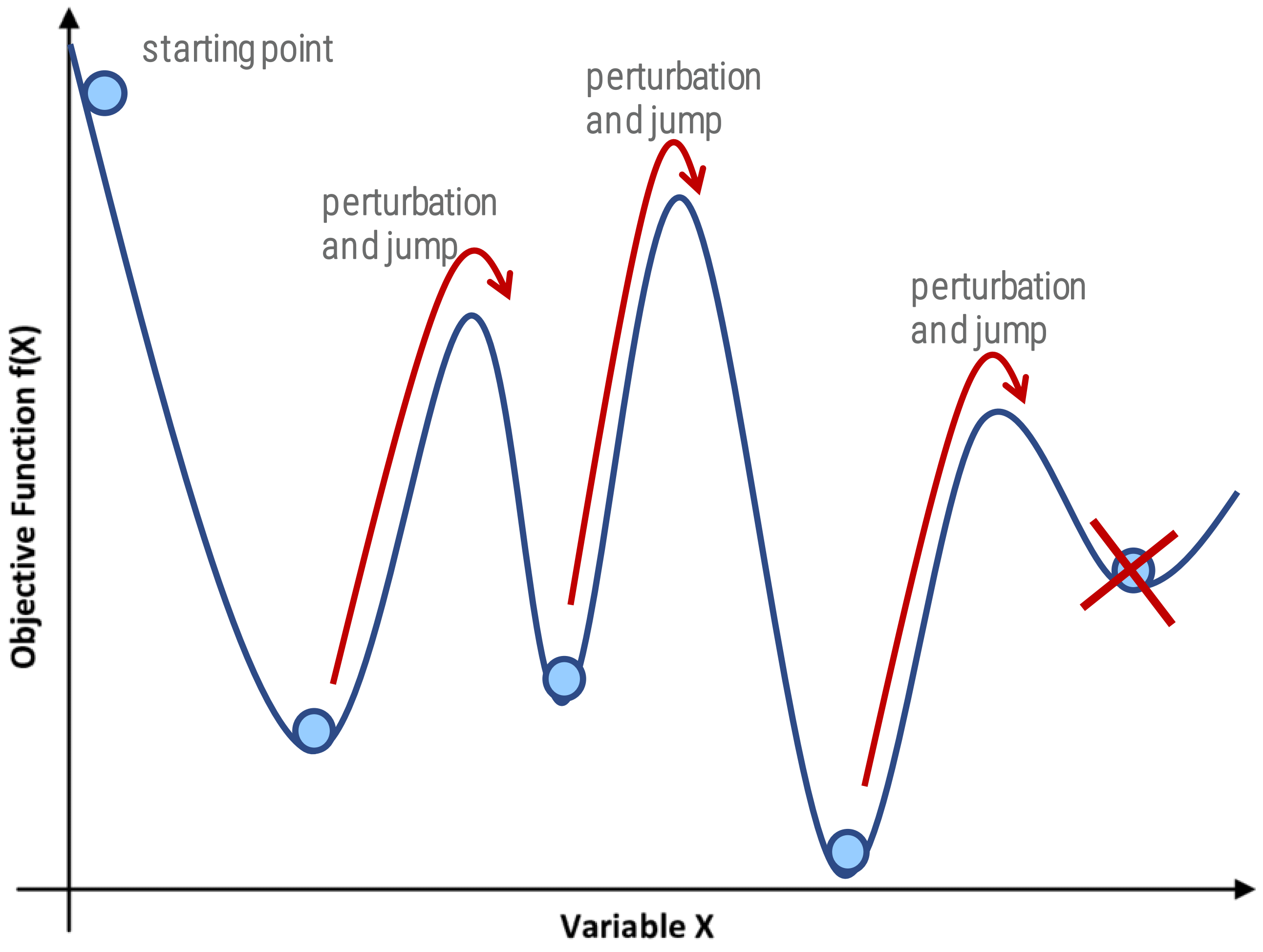
# Annealing Algorithms

- 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.



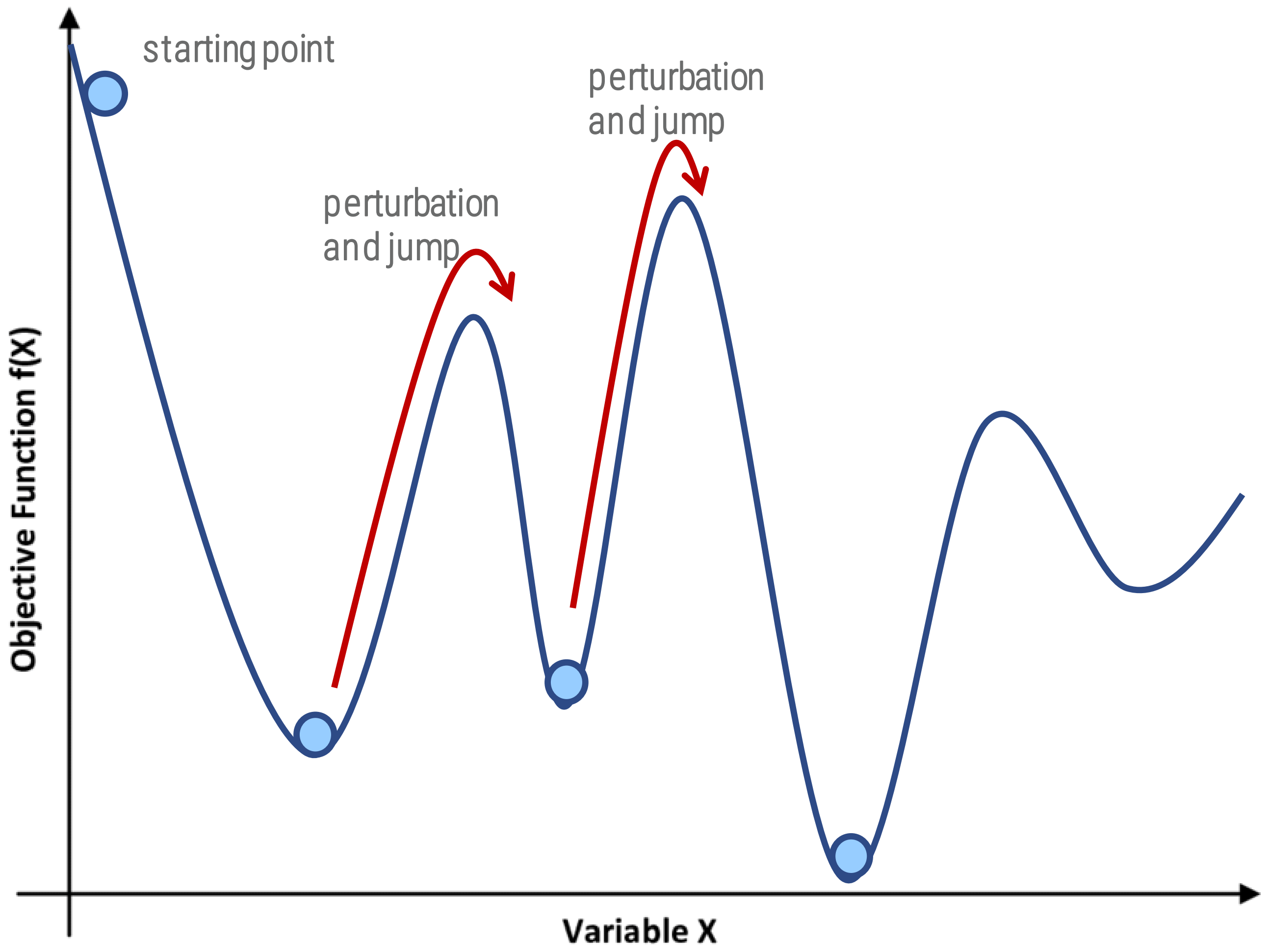
# Annealing Algorithms

- 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.



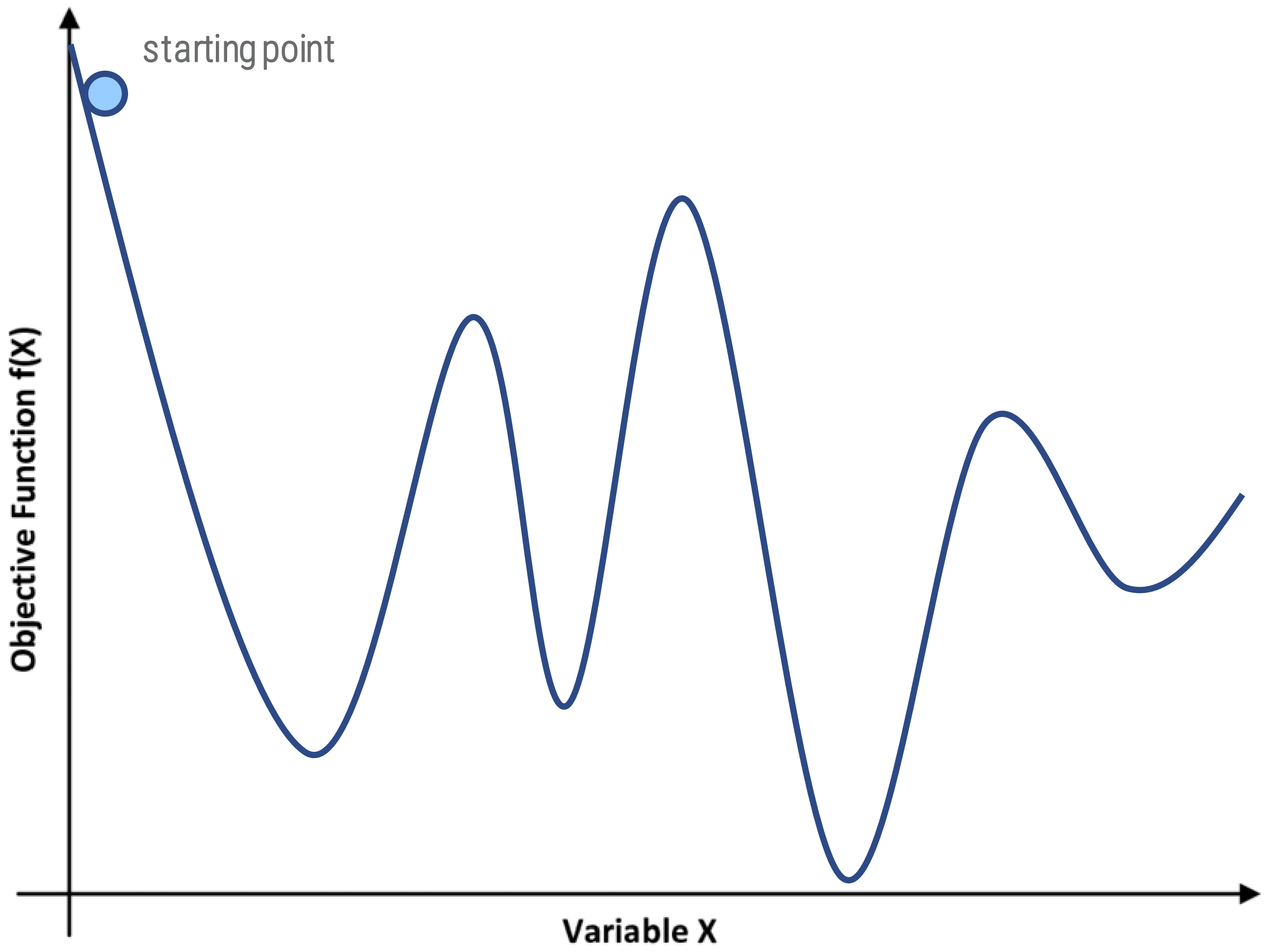
# Annealing Algorithms

- 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.



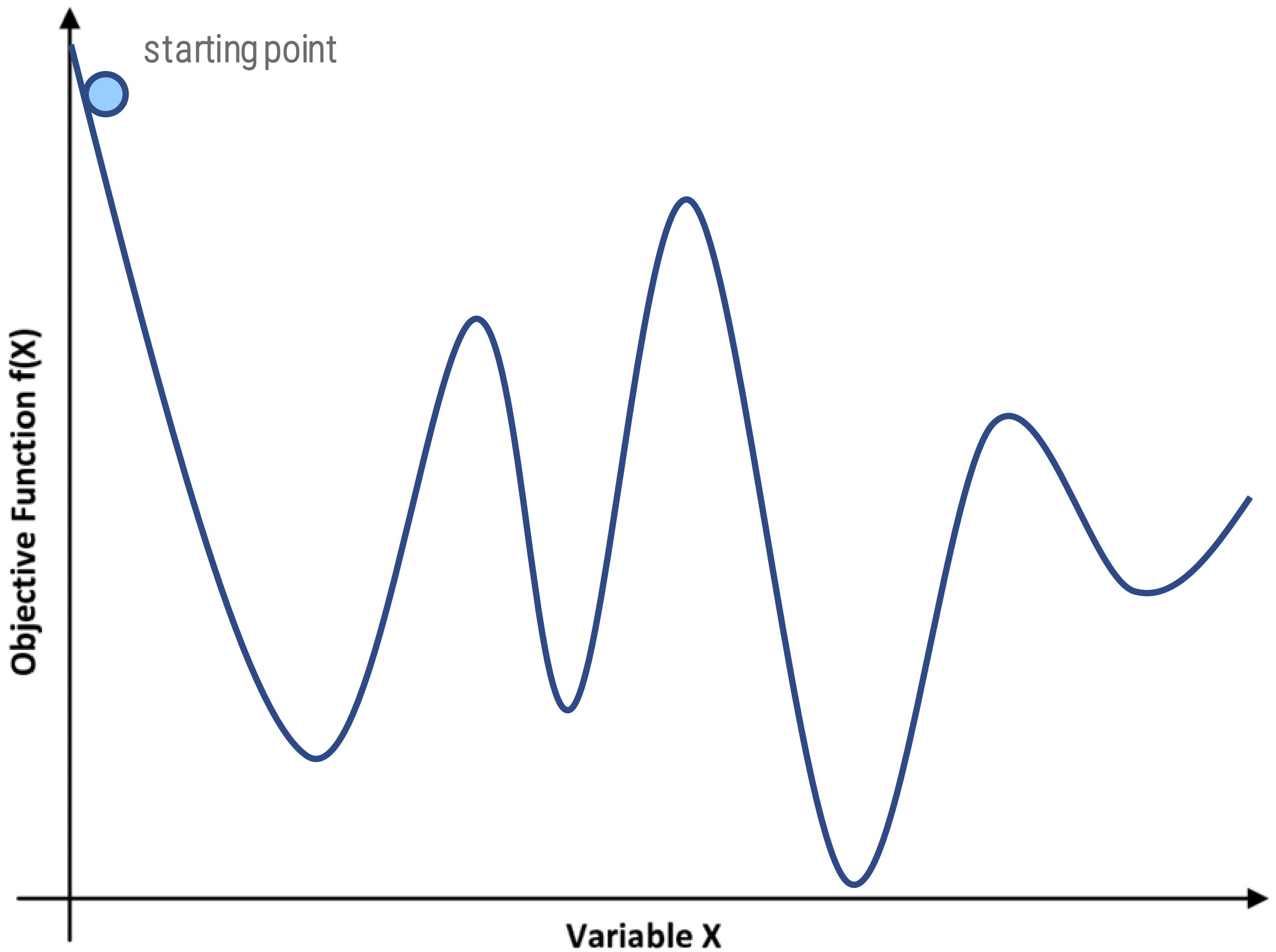
# Quantum Annealing

- **Quantum Annealing** is the quantum version of simulated annealing



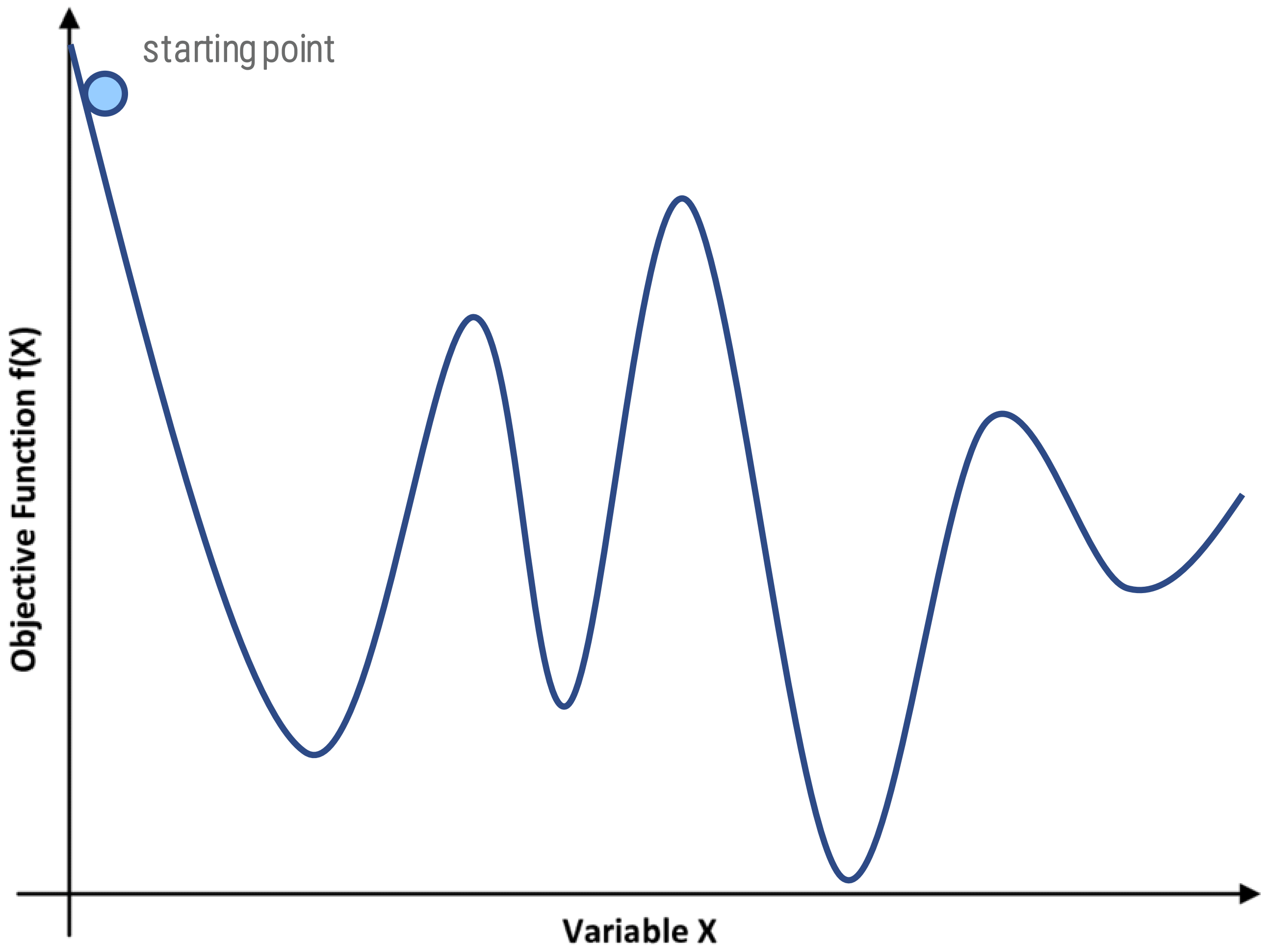
# Quantum Annealing

- **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**



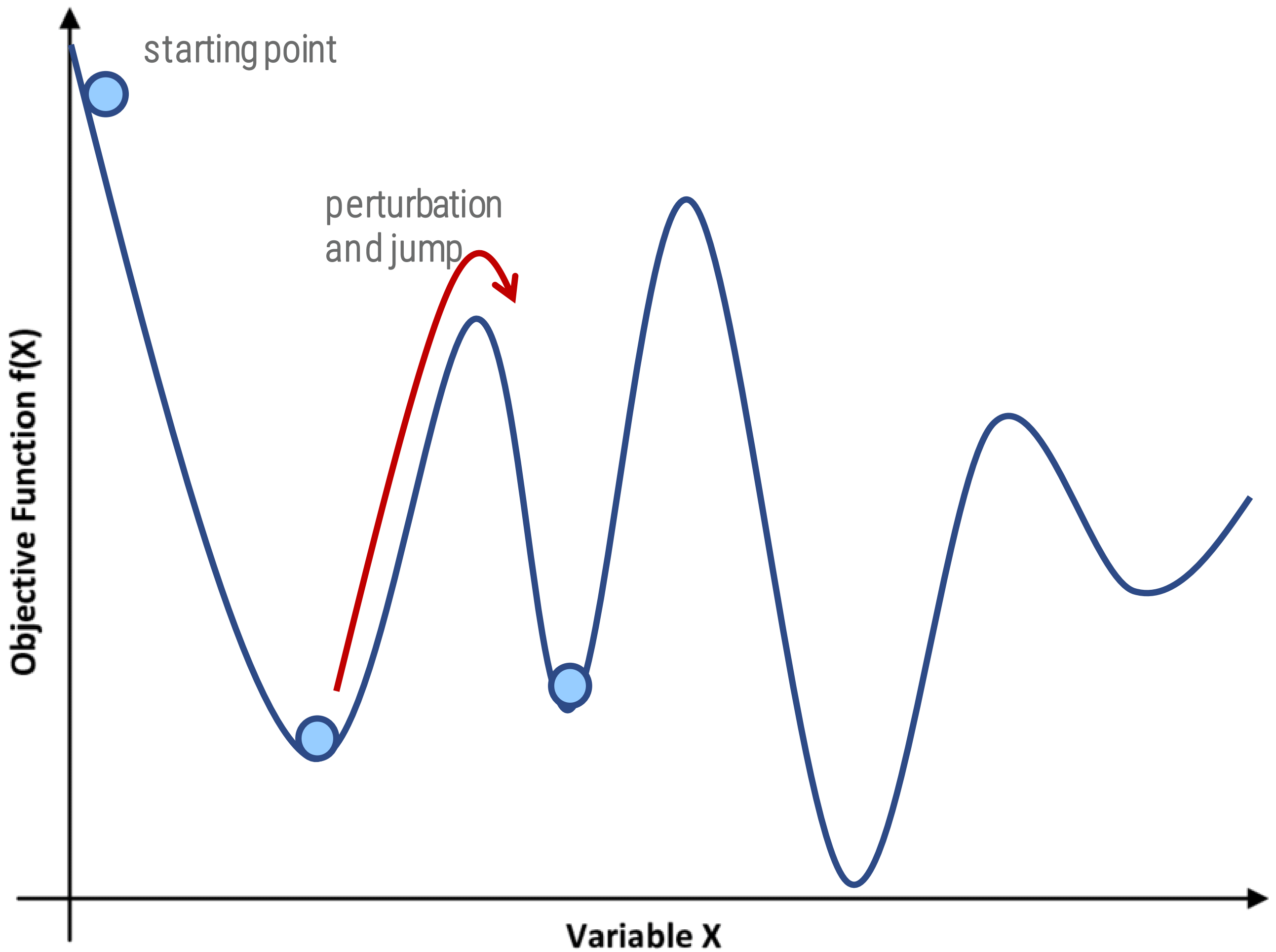
# Quantum Annealing

- **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.



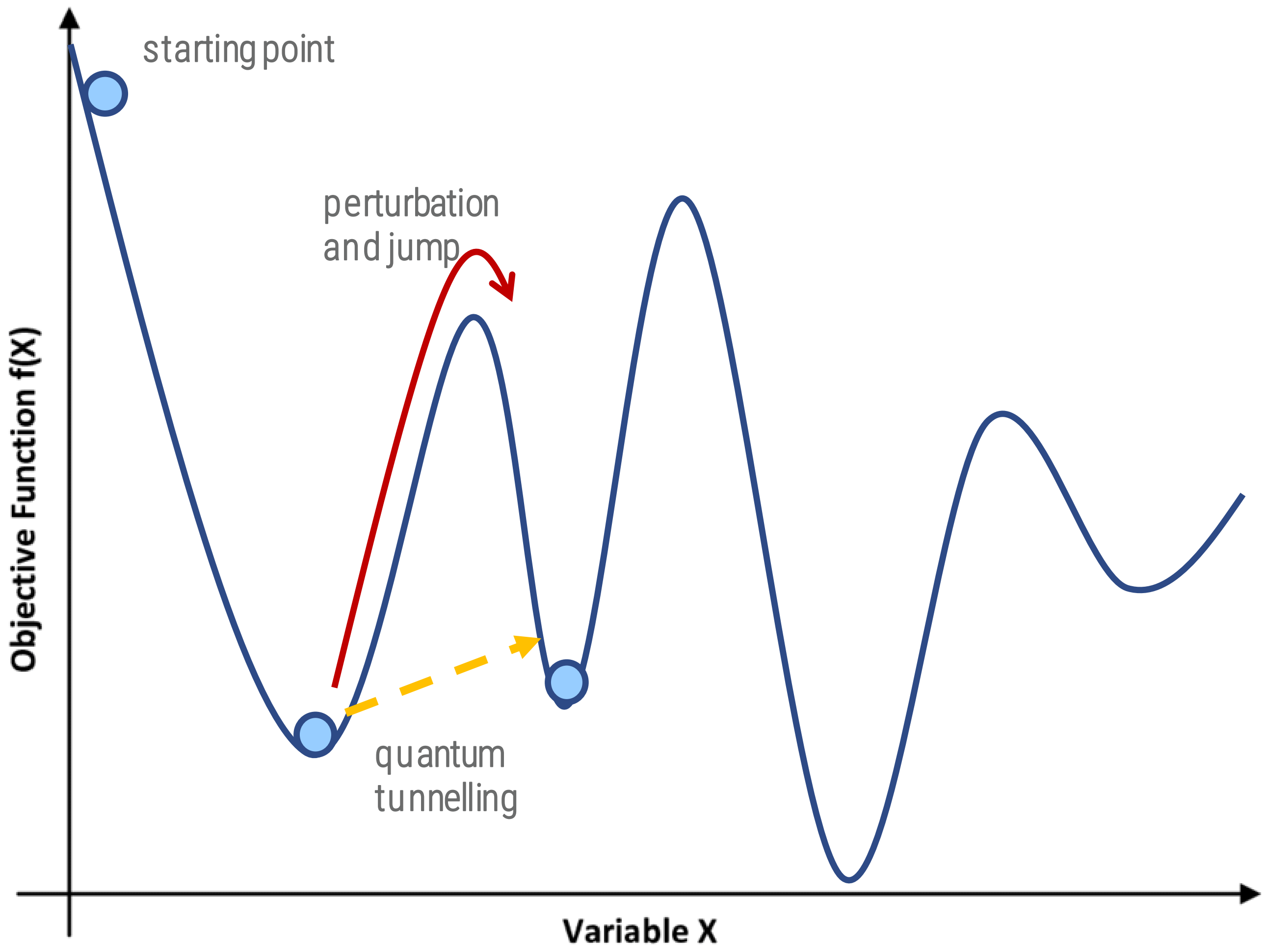
# Quantum Annealing

- **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.



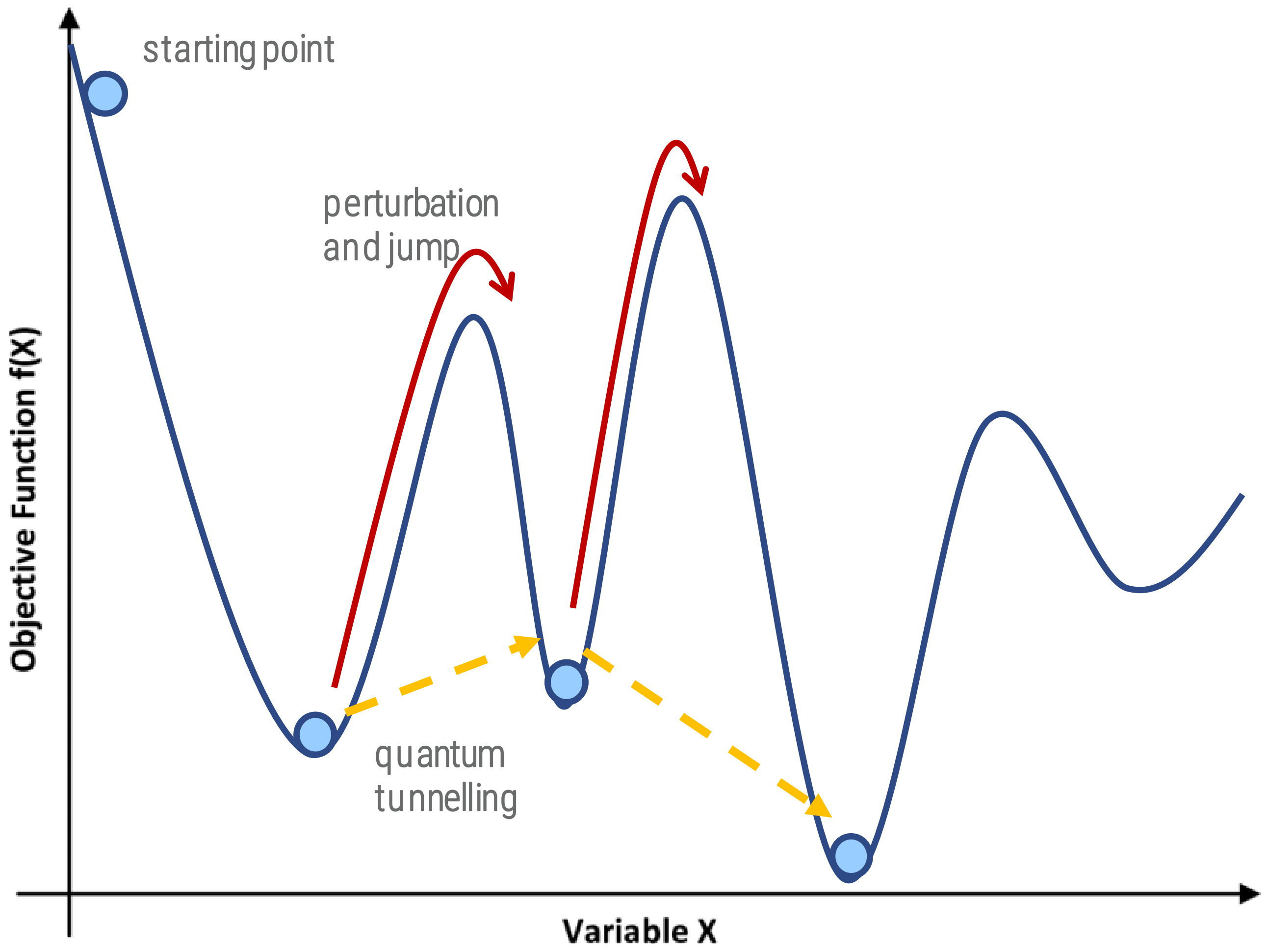
# Quantum Annealing

- **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.



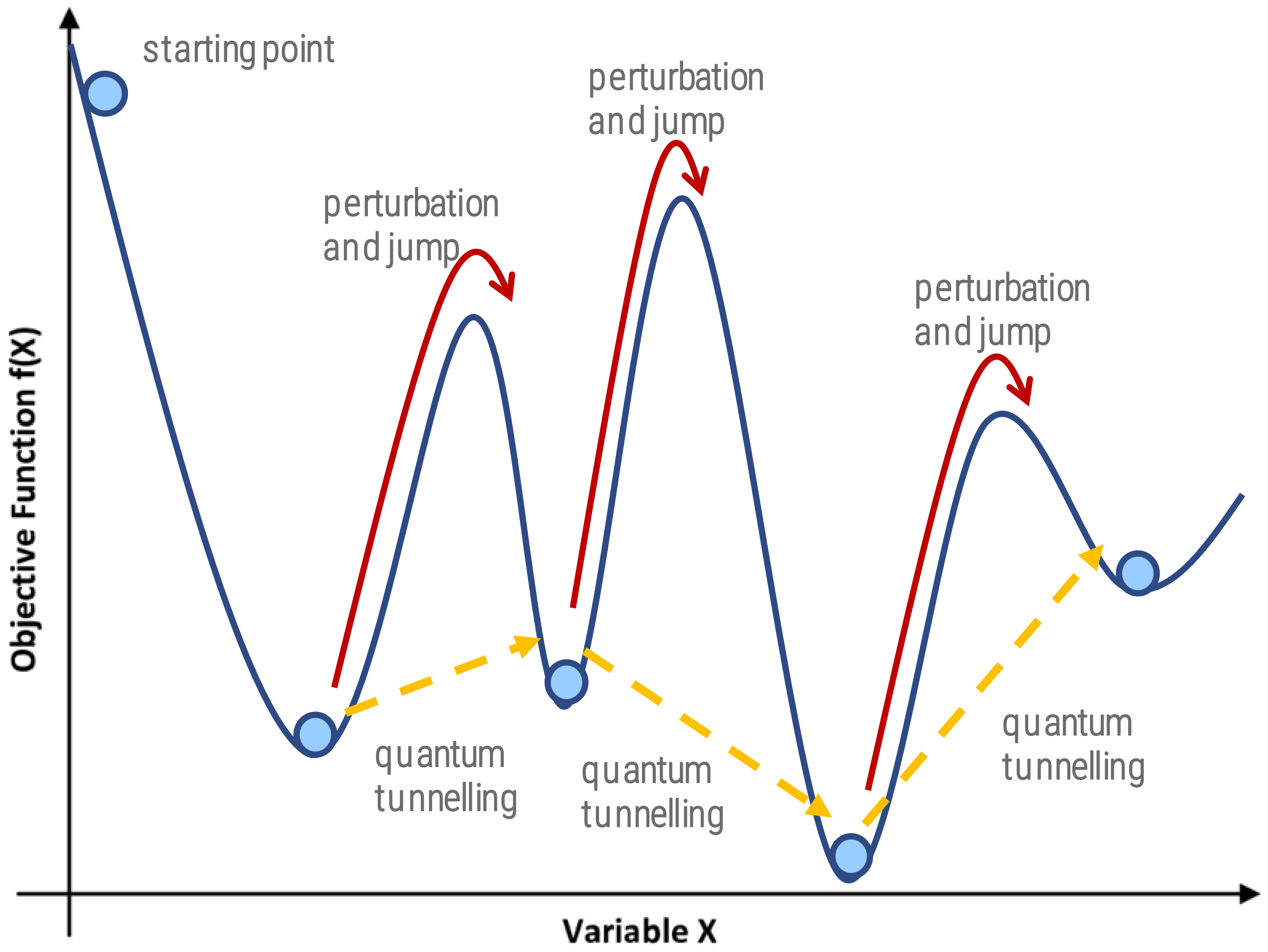
# Quantum Annealing

- **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.



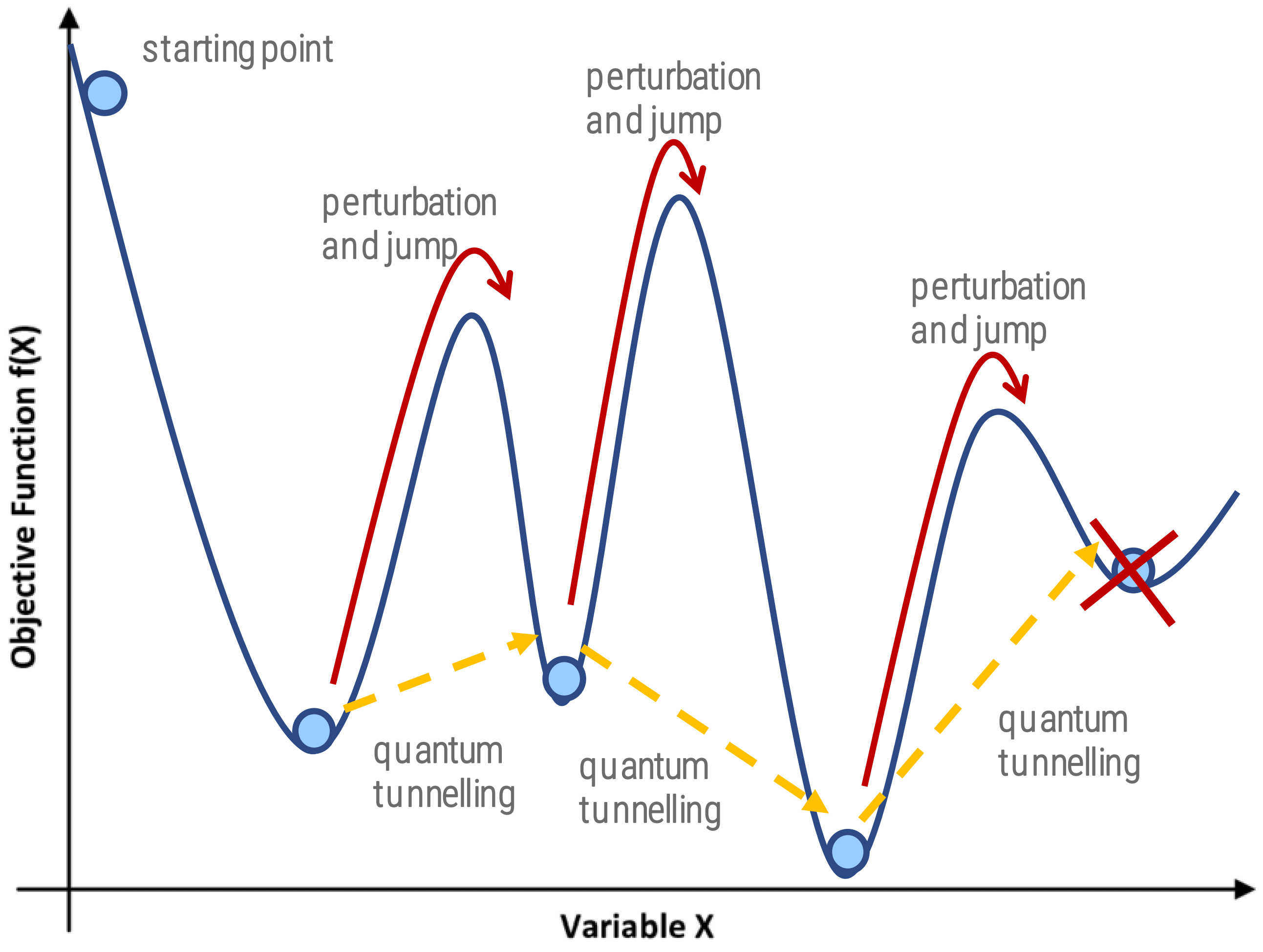
# Quantum Annealing

- **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.



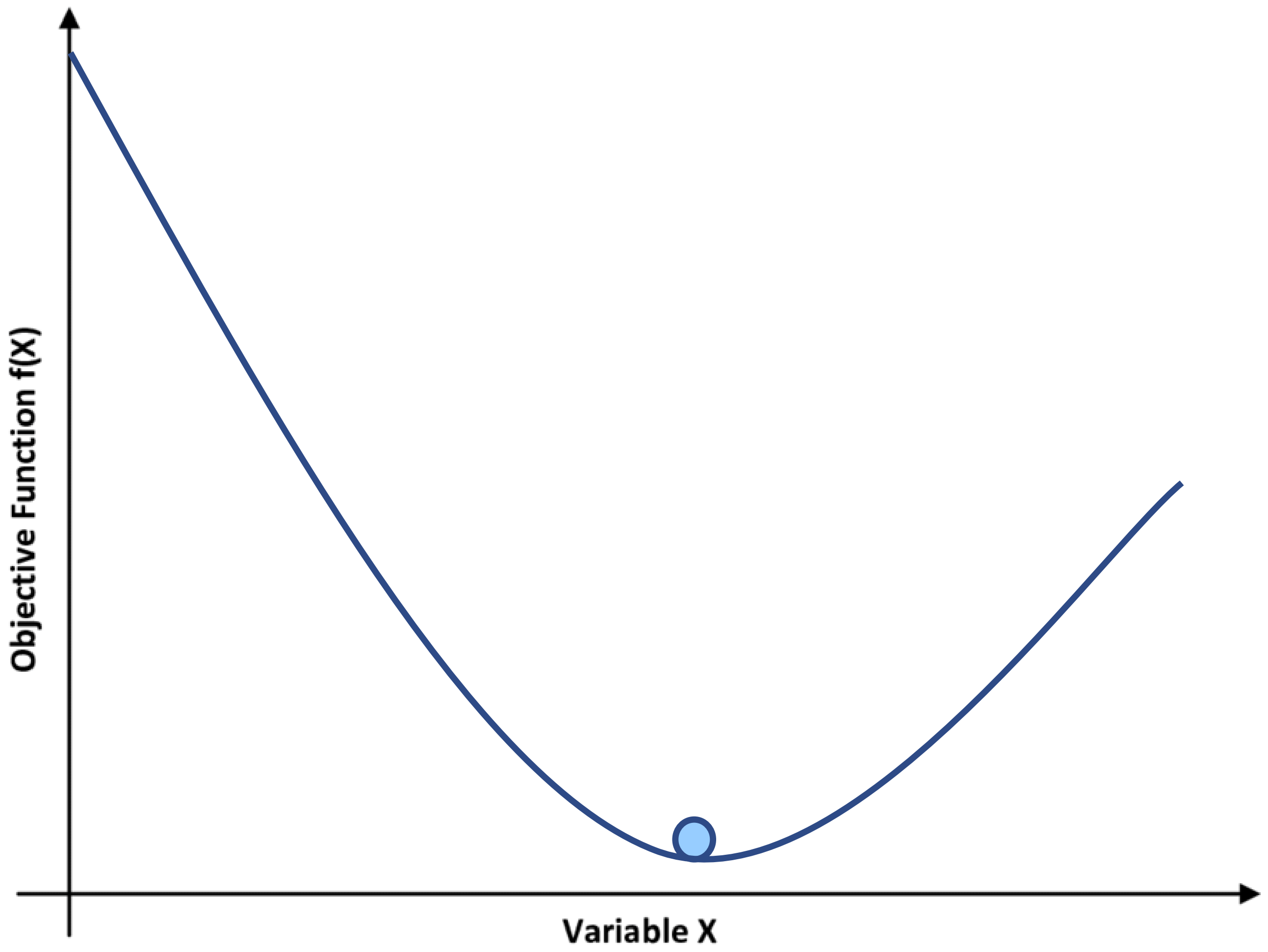
# Quantum Annealing

- **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*



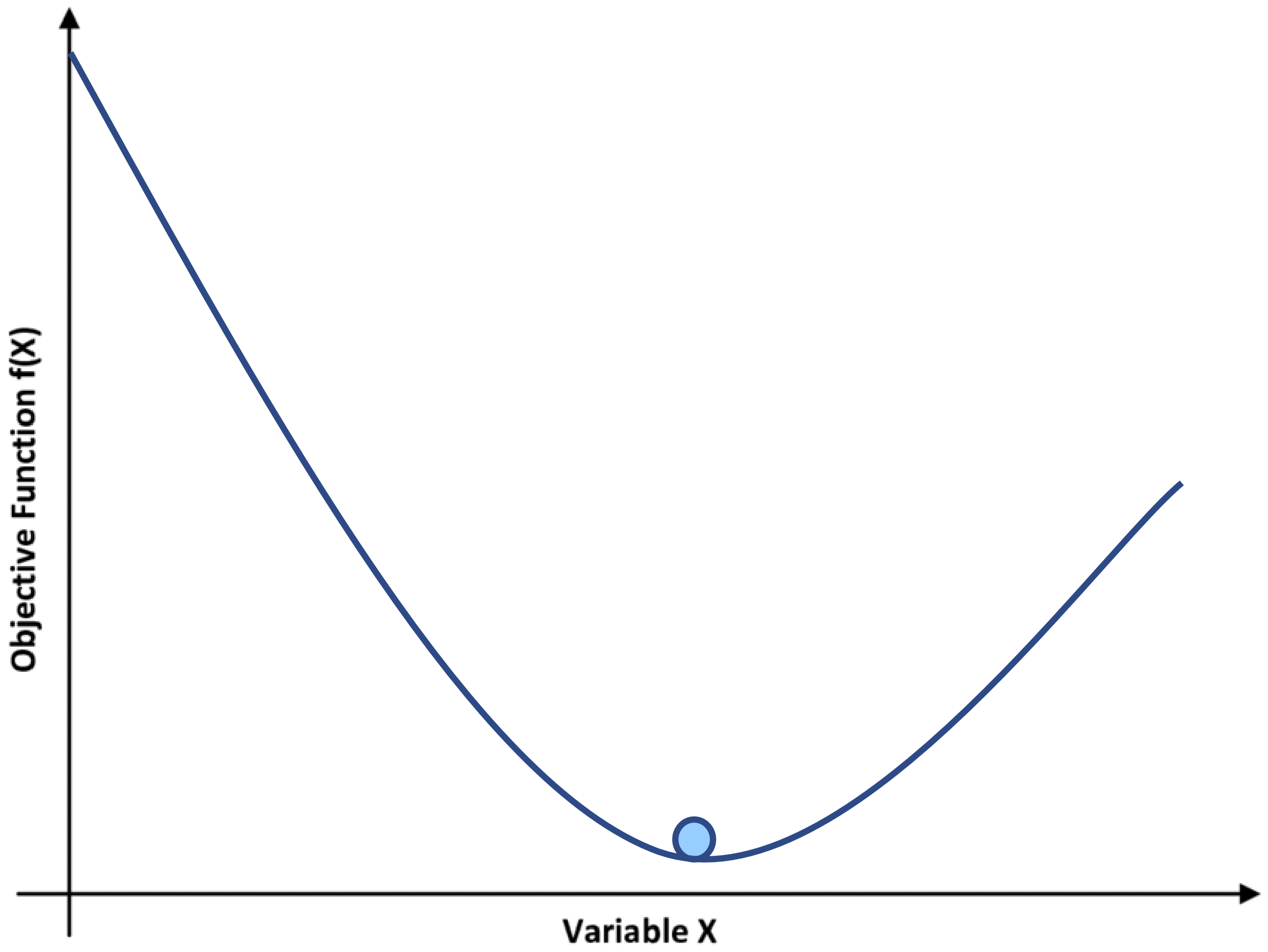
# Quantum Annealing

- Optimization through quantum annealing begins with choosing an objective function **different** than the one you want to optimize.



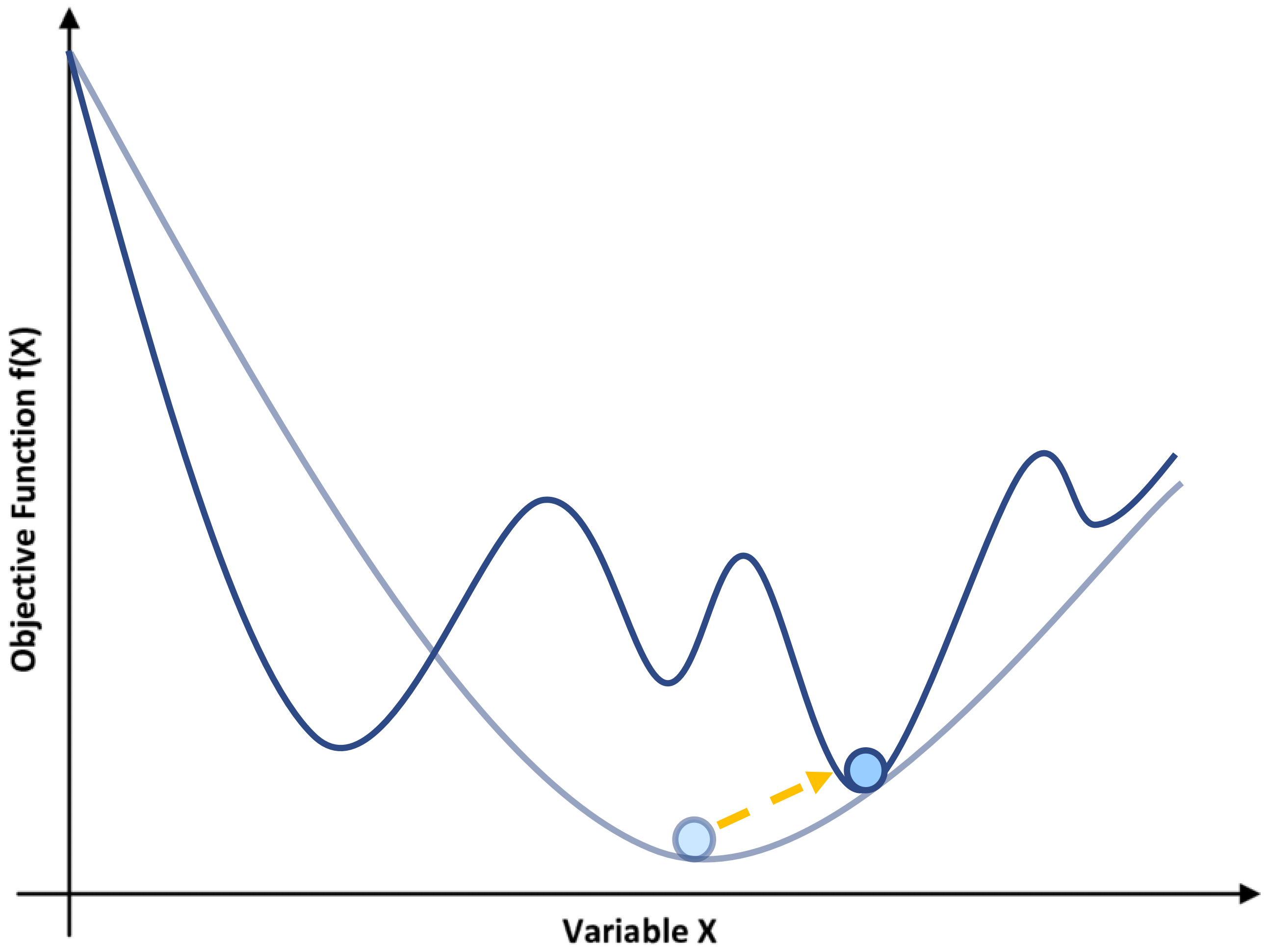
# Quantum Annealing

- 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).



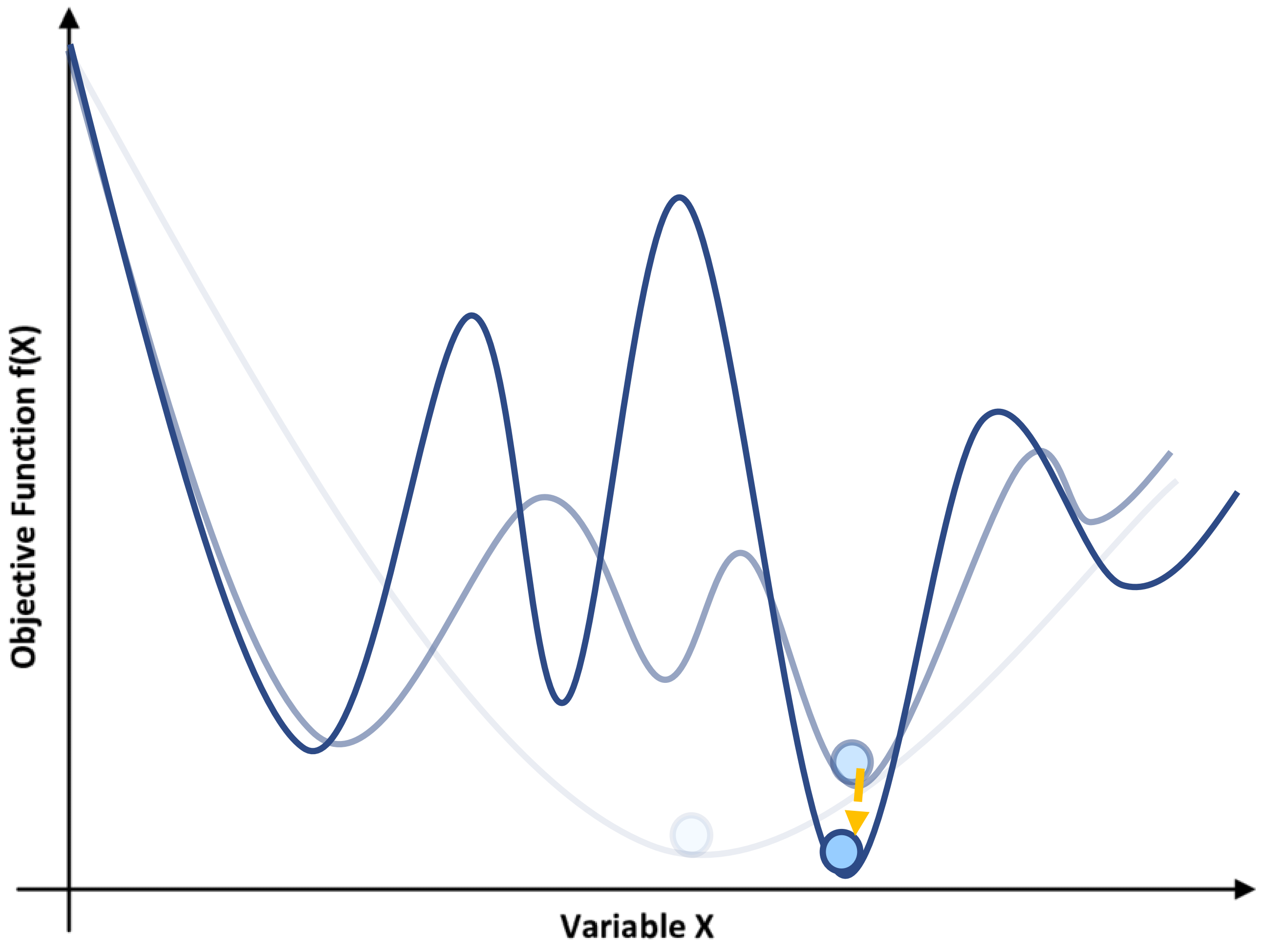
# Quantum Annealing

- 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



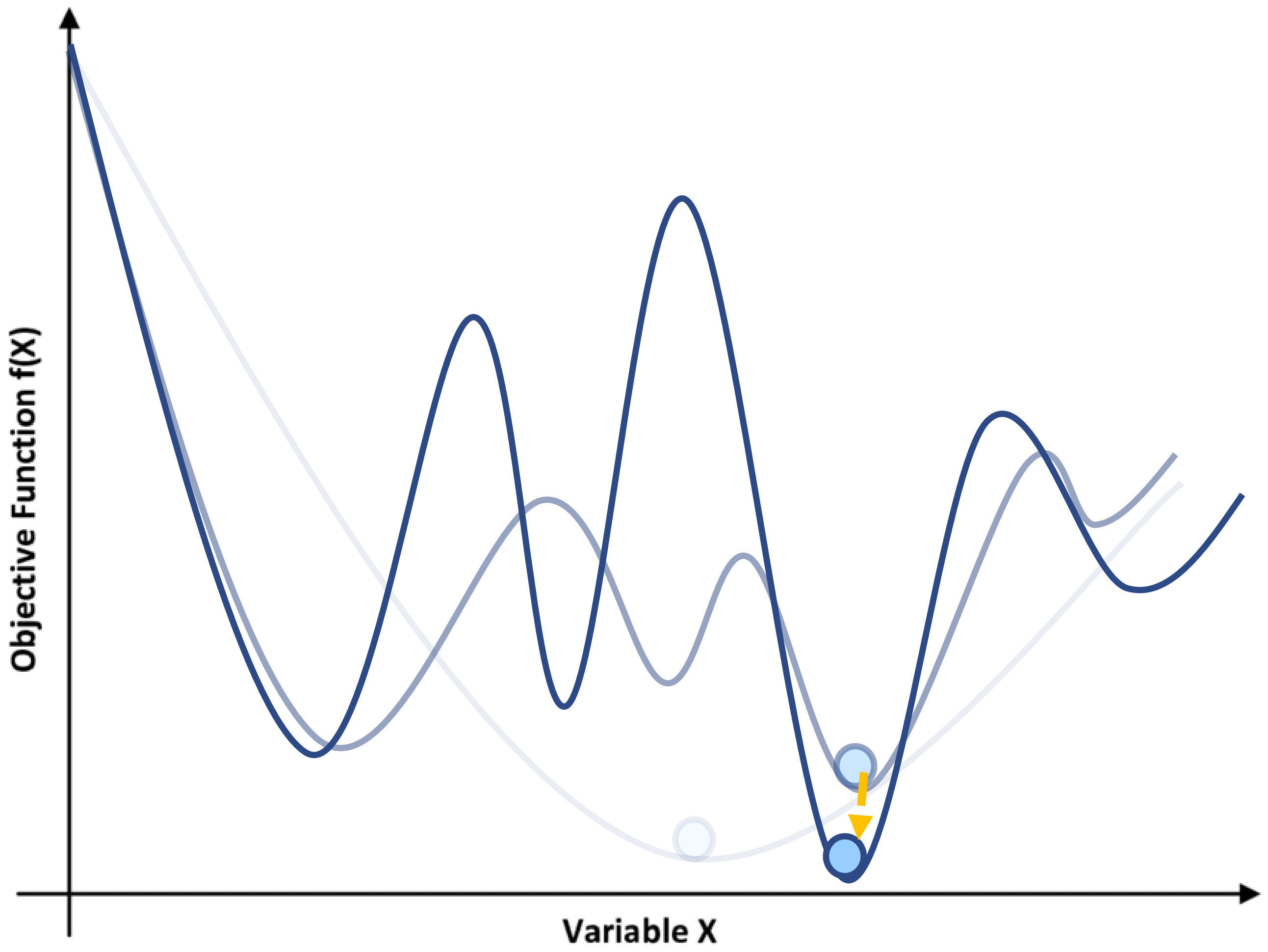
# Quantum Annealing

- 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



# Quantum Annealing

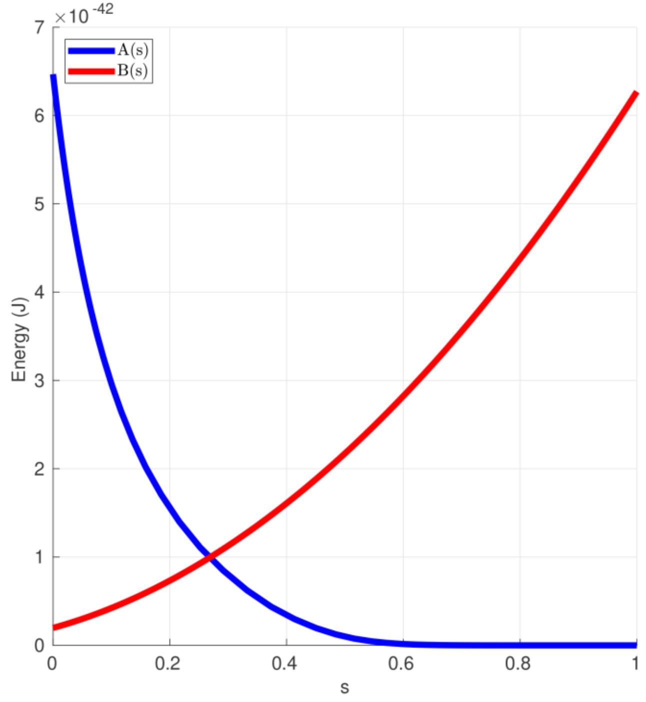
- Optimization through quantum annealing begins with choosing an objective function **different** than the one you want to optimize.
- The choice always falls on a **simple function, of which the global minimum is known** (for example).
- The annealing process consists in **slowly modifying** the objective 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**

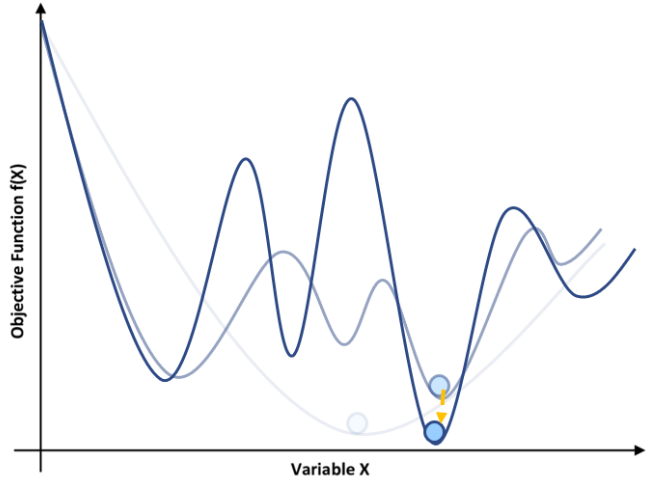


# Quantum Annealing

- 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**

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



# 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 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**

# 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

# 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

# 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:

# 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

# 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)

# The Quantum Annealer

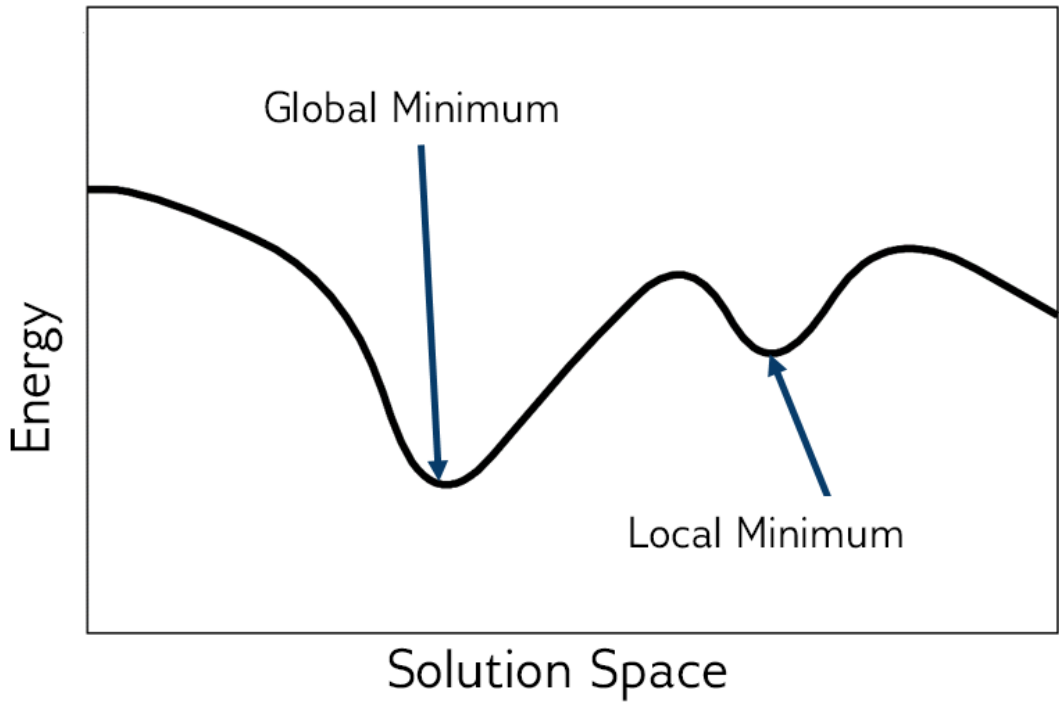
- Currently, we can talk about **The** quantum annealer and not about **a** generic quantum annealer **since today there is only one manufacturer for this type of device.**
- The company in question is called **D-Wave**
- At the moment the latest quantum annealer model has more than 5000 qubits and about 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)

# 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

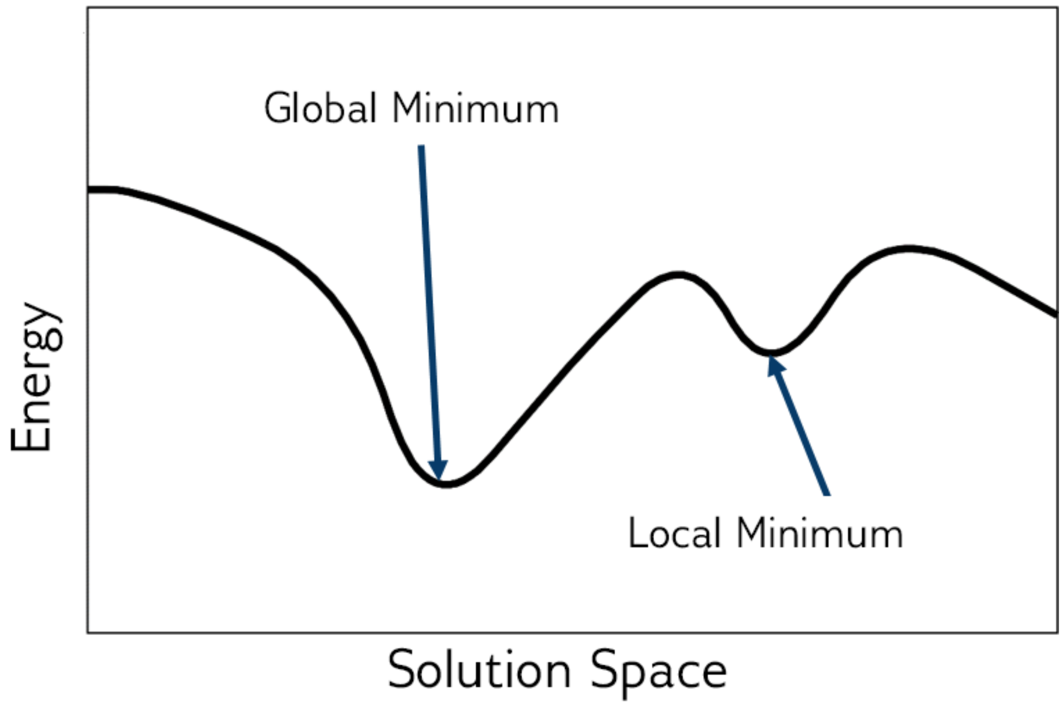
# Objective Function

- To express a problem in a form that allows its resolution through quantum annealing, we need first of all an objective function



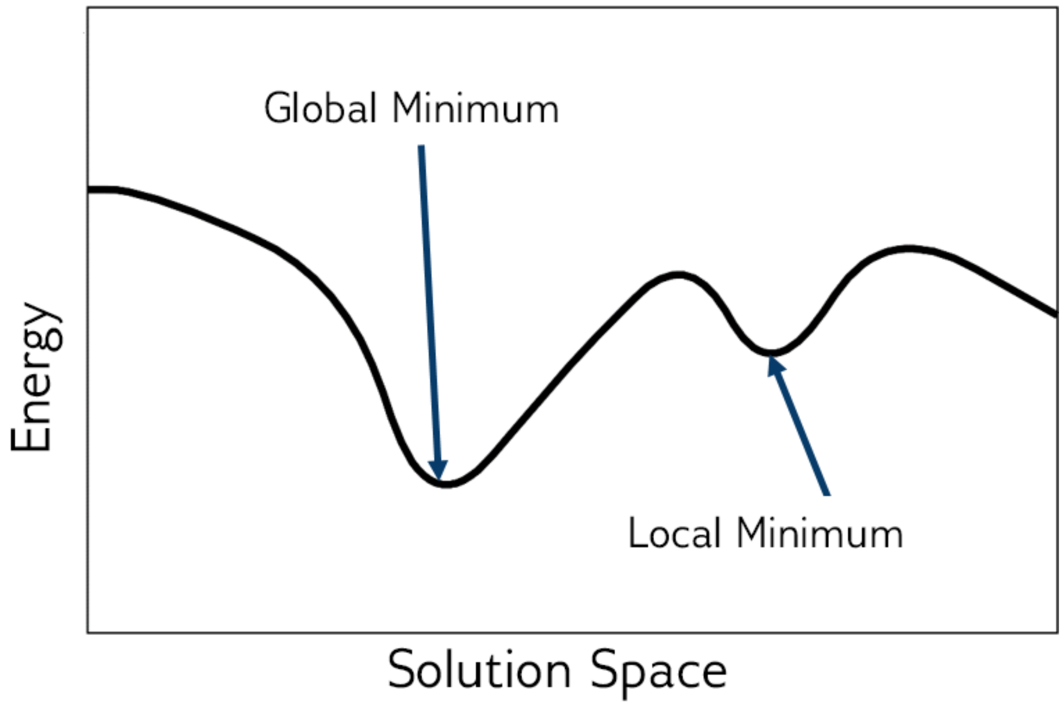
# Objective Function

- To express a problem in a form that allows its resolution through quantum annealing, we need first of all 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



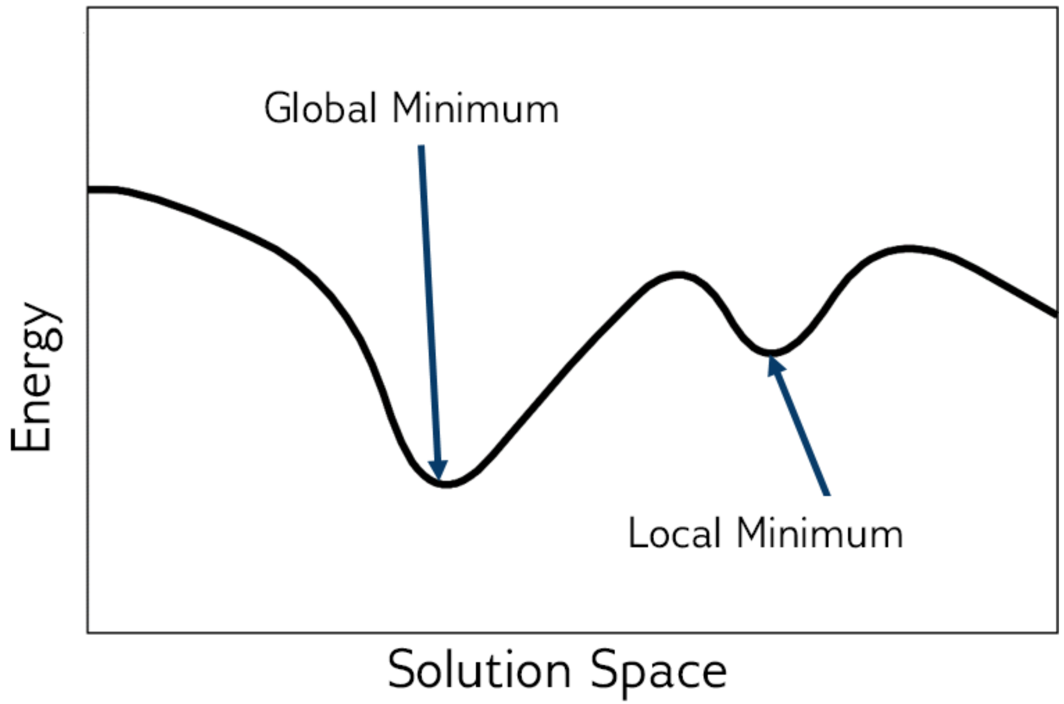
# Objective Function

- To express a problem in a form that allows its resolution through quantum annealing, we need first of all 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.



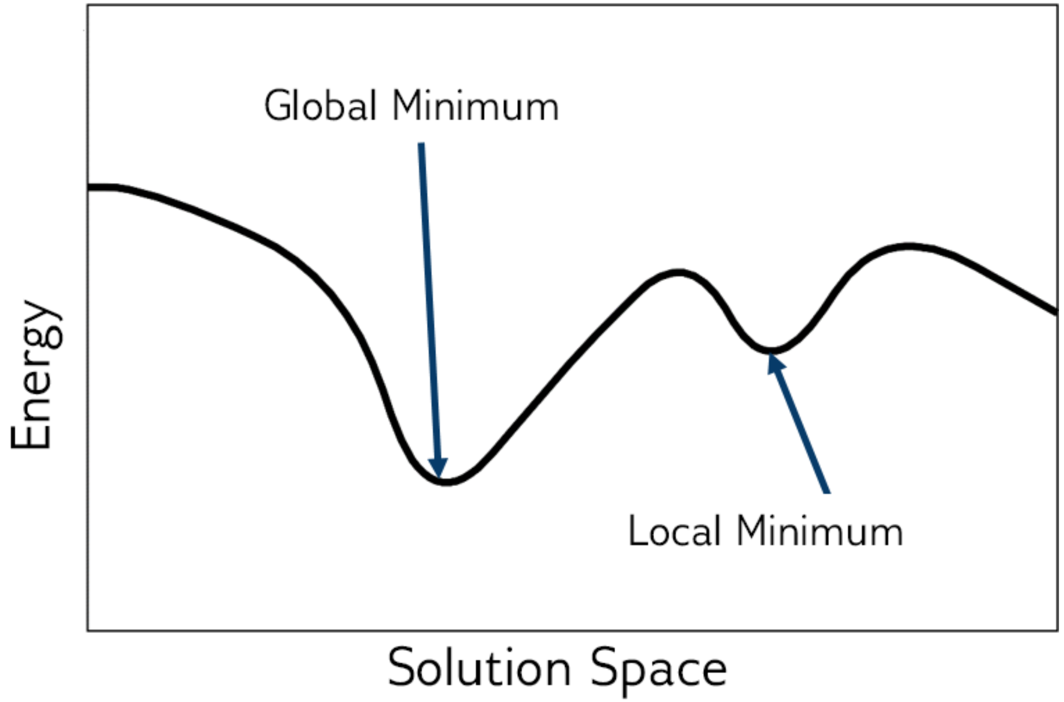
# Objective Function

- To express a problem in a form that allows its resolution through quantum annealing, we need first of all 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**

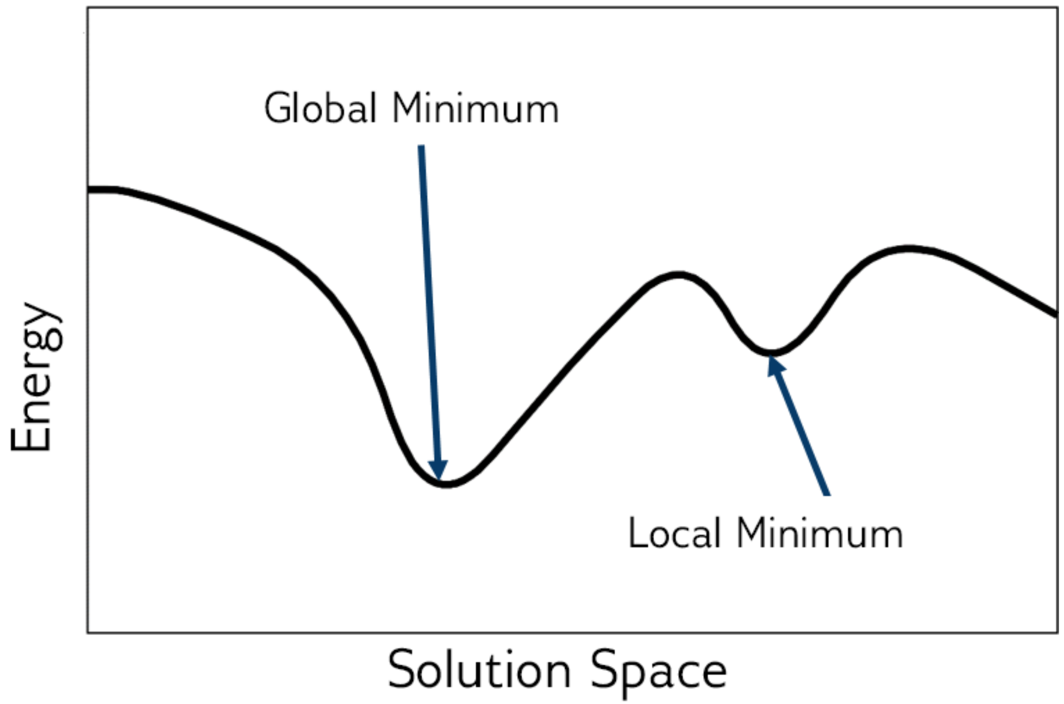



$$x + 1 = 2$$

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

# 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

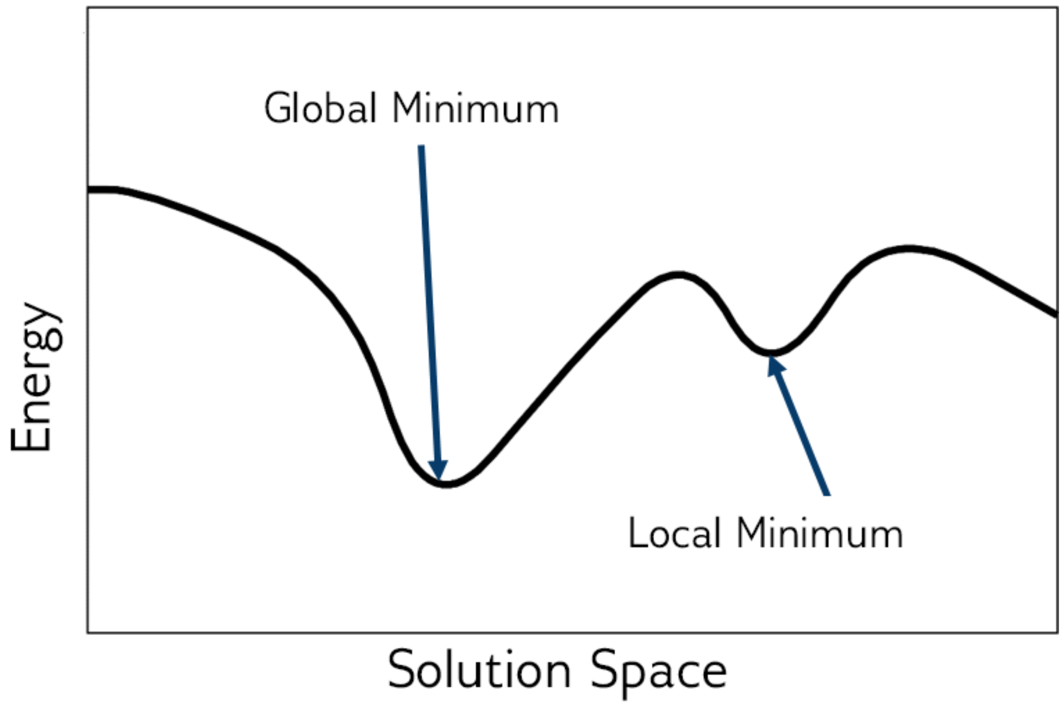



$$x + 1 = 2$$

$$\min_{x} [2 - (x + 1)]^2$$

# 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.

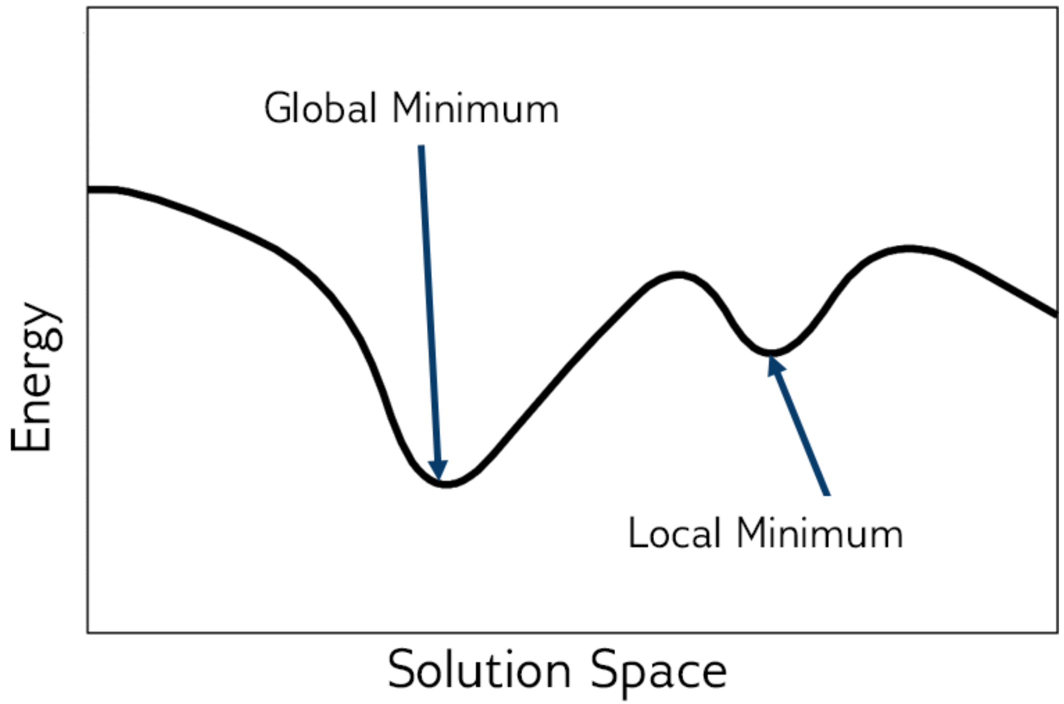



$$x + 1 = 2$$

$$\min_{x}[2 - (x + 1)]^2$$

# 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**




$$x + 1 = 2$$

$$\min_{x}\left[2 - (x+1)\right]^2$$

# Ising Model

- The Ising Model is a well-known model in statistical mechanics.

# 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).

# 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**.

# 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

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

# 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

# Game of Switches


- The switch game is a very simple game that can help you understand the nature of an optimization problem that can be solved by a quantum annealer.

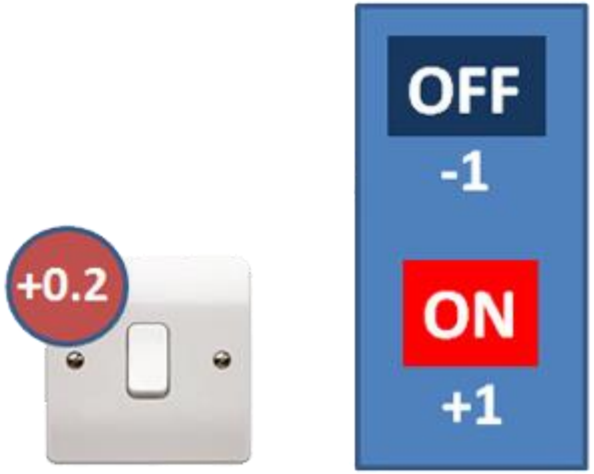
# Game of Switches


- The switch game is a very simple game that can help you understand the nature of an optimization problem that can be solved by a quantum annealer.
- Suppose we have a certain number of switches, each settable on two possible states represented by the values 1 and -1

# Game of Switches


- The switch game is a very simple game that can help you understand the nature of an optimization problem that can be solved by a quantum annealer.
- Suppose we have a certain number of switches, each settable on two possible states represented by the values 1 and -1
- Furthermore, each switch has a univocally associated weight

# Game of Switches

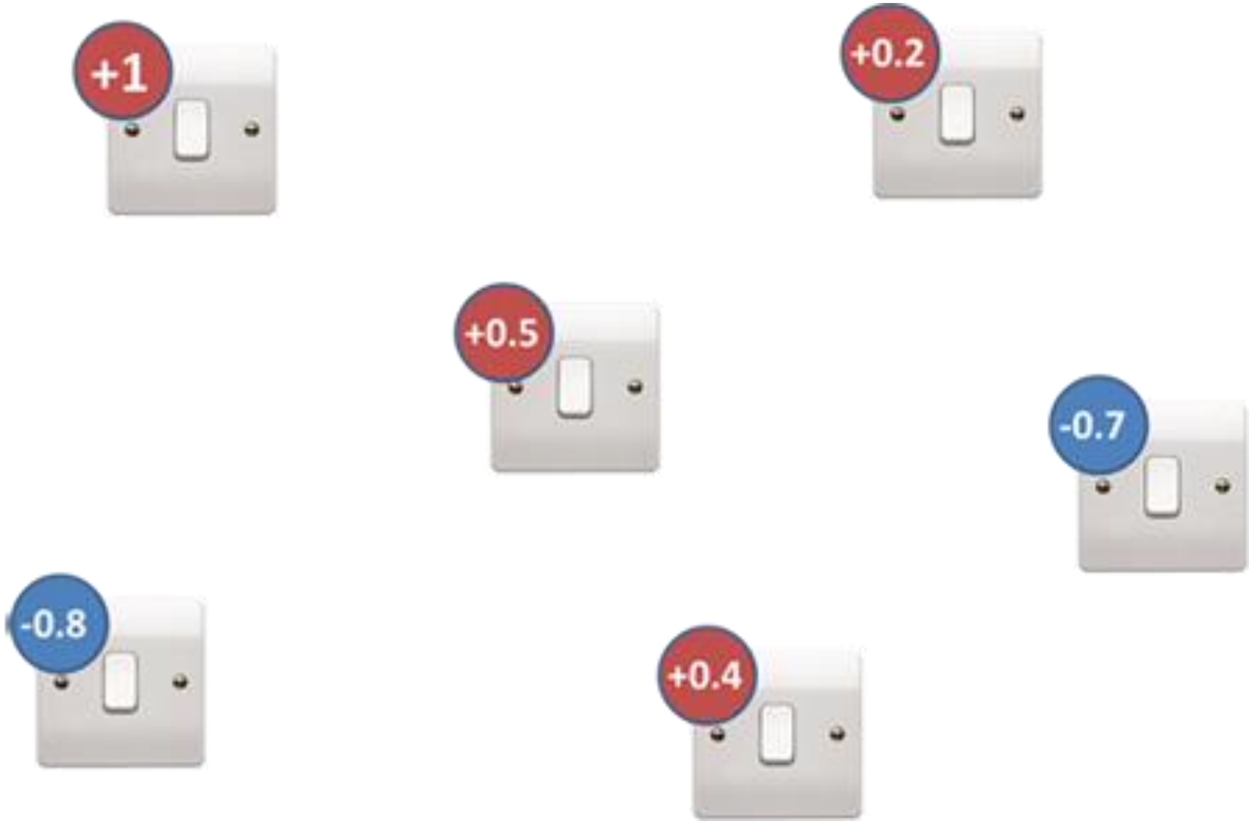

h = 'bias' value associated with each switch

s = the ON/OFF setting of each switch,+1 or -1

# Game of Switches


- The switch game is a very simple game that can help you understand the nature of an optimization problem that can be solved by a quantum annealer.
- Suppose we have a certain number of switches, each settable on two possible states represented by the values 1 and -1
- Furthermore, each switch has a univocally associated weight
- The value of a switch is calculated by multiplying its weight by its state

# Game of Switches




- The switch game is a very simple game that can help you understand the nature of an optimization problem that can be solved by a quantum annealer.
- Suppose we have a certain number of switches, each settable on two possible states represented by the values 1 and -1
- Furthermore, each switch has a univocally associated weight
- The value of a switch is calculated by multiplying its weight by its state
- The game consists in finding the combination of states for the switches such that the sum of their values is as low as possible

# Game of Switches

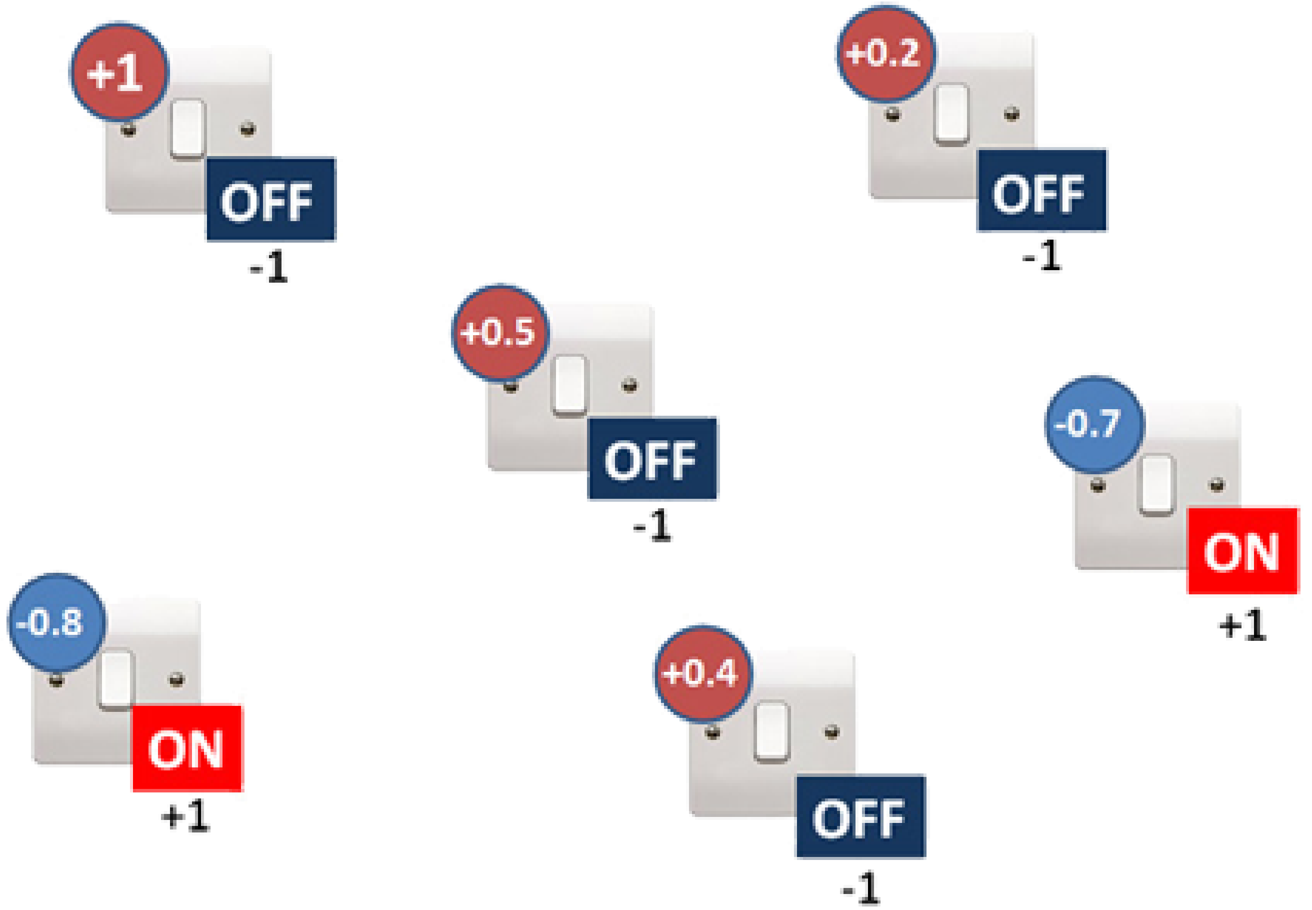

$$E(\mathbf{s}) = \sum_i h_i s_i$$



h = ‘bias’ value associated with each switch

s = the ON/OFF setting of each switch,+1 or -1

# Game of Switches




$+1 \times -1 = -1$

$+0.2 \times -1 = -0.2$

$+0.5 \times -1 = -0.5$

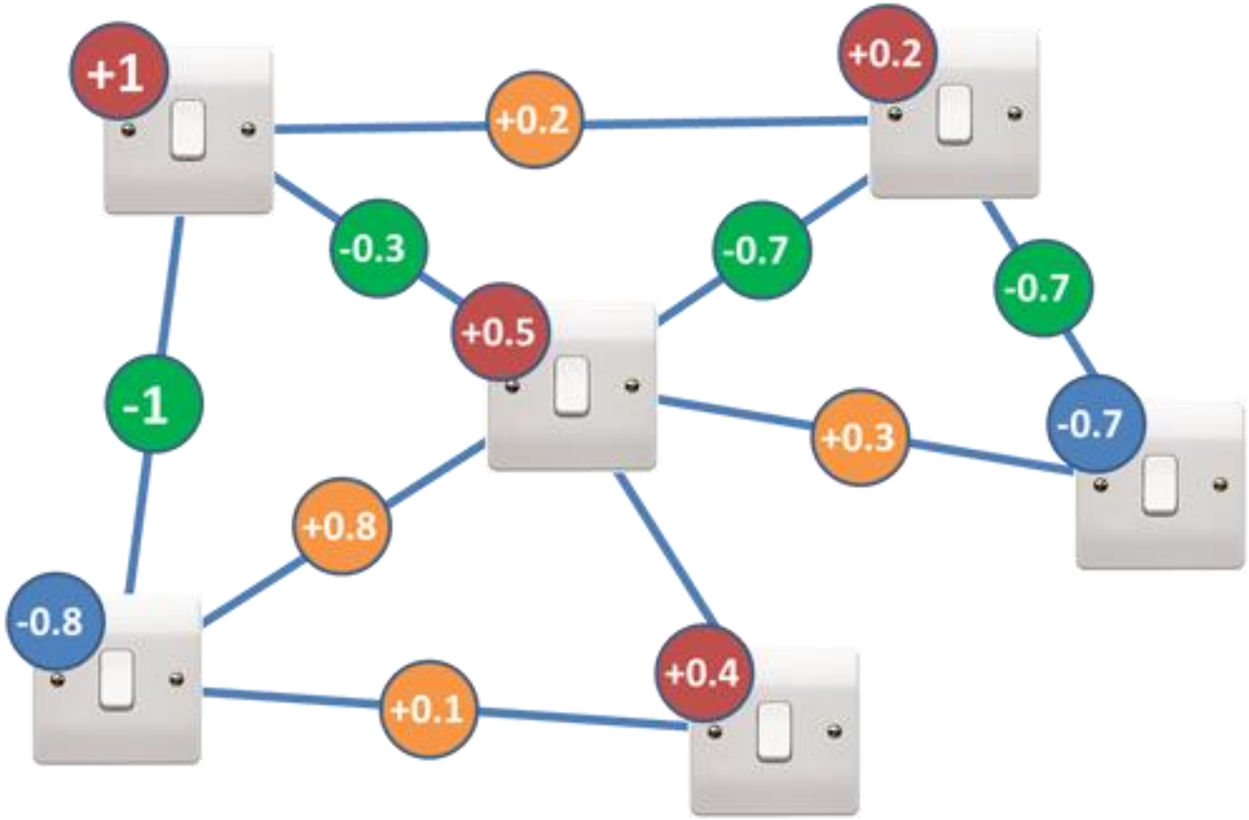
$-0.8 \times +1 = -0.8$

$+0.4 \times -1 = -0.4$

$-0.7 \times +1 = -0.7$

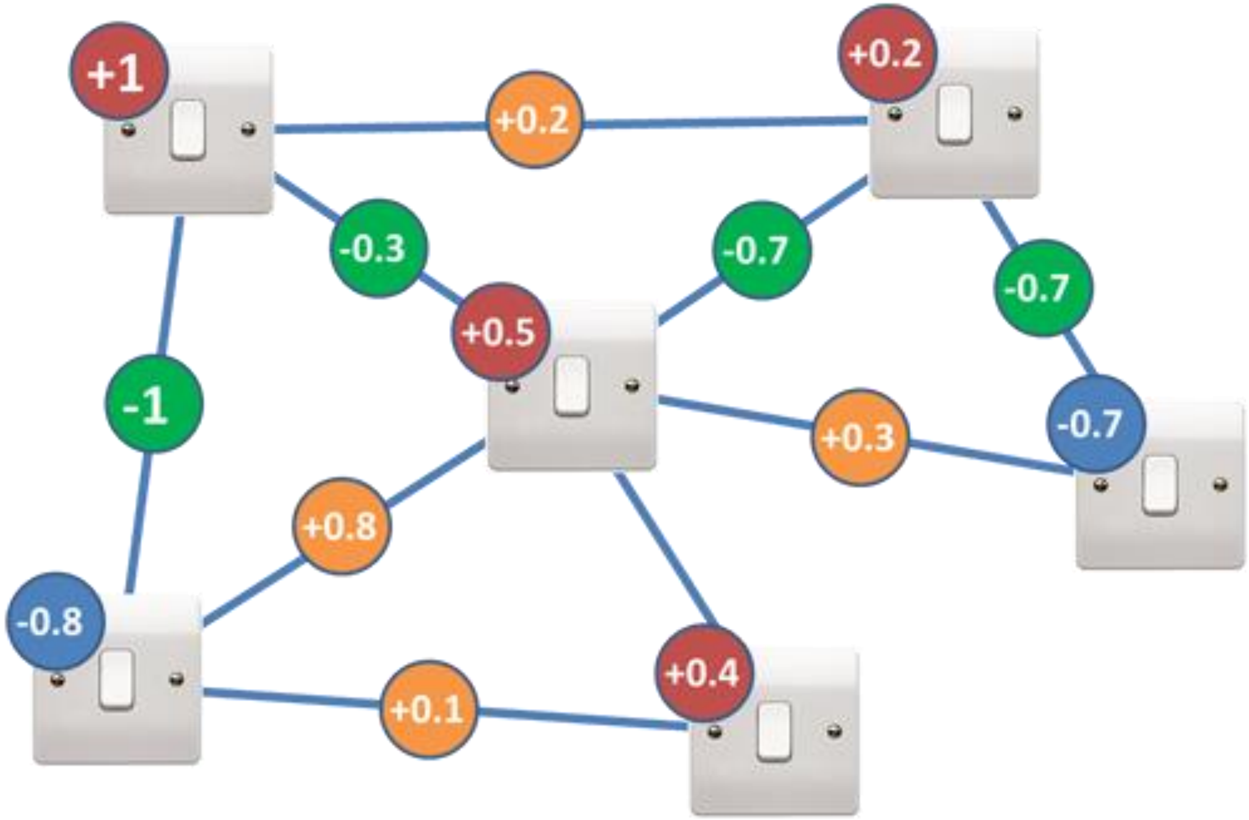
Total: -3.6

# Game of Switches



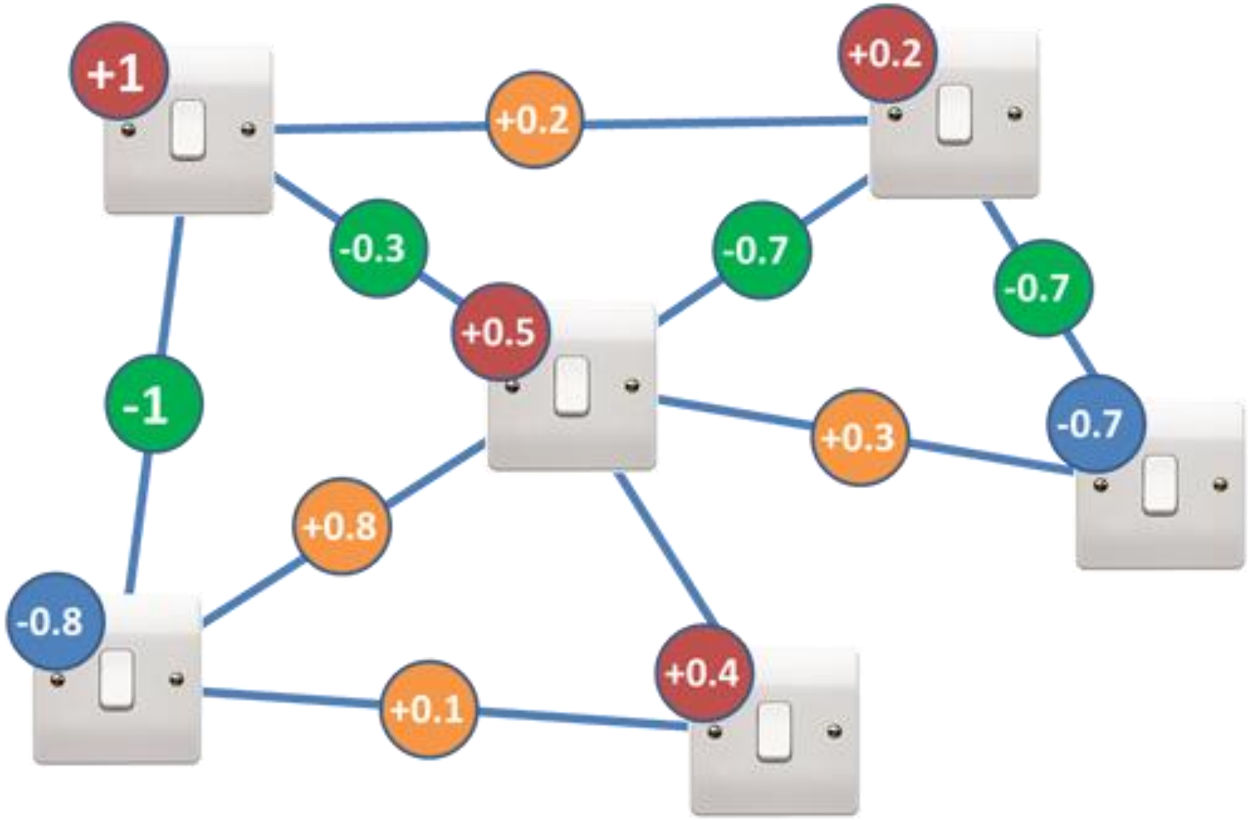

- Let us now consider another factor, namely the presence of couplers between the switches

# Game of Switches



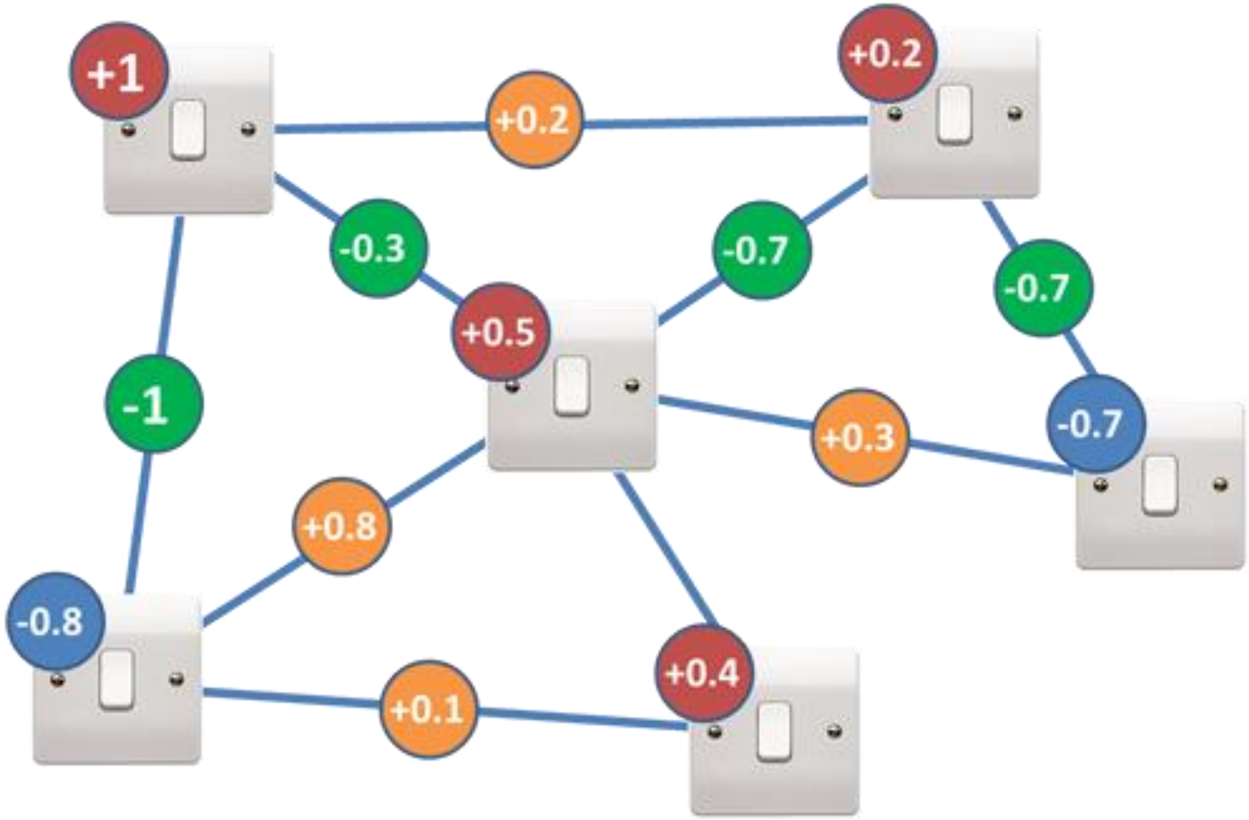

- Let us now consider another factor, namely the presence of couplers between the switches
- Couplers, just like switches, are endowed with a certain numerical weight

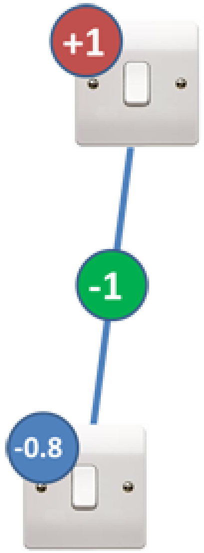
# Game of Switches




- Let us now consider another factor, namely the presence of couplers between the switches
- Couplers, just like switches, are endowed with a certain numerical weight
- The value of the couplers is given by their own weight multiplied by the state of the switches to which the coupler is associated

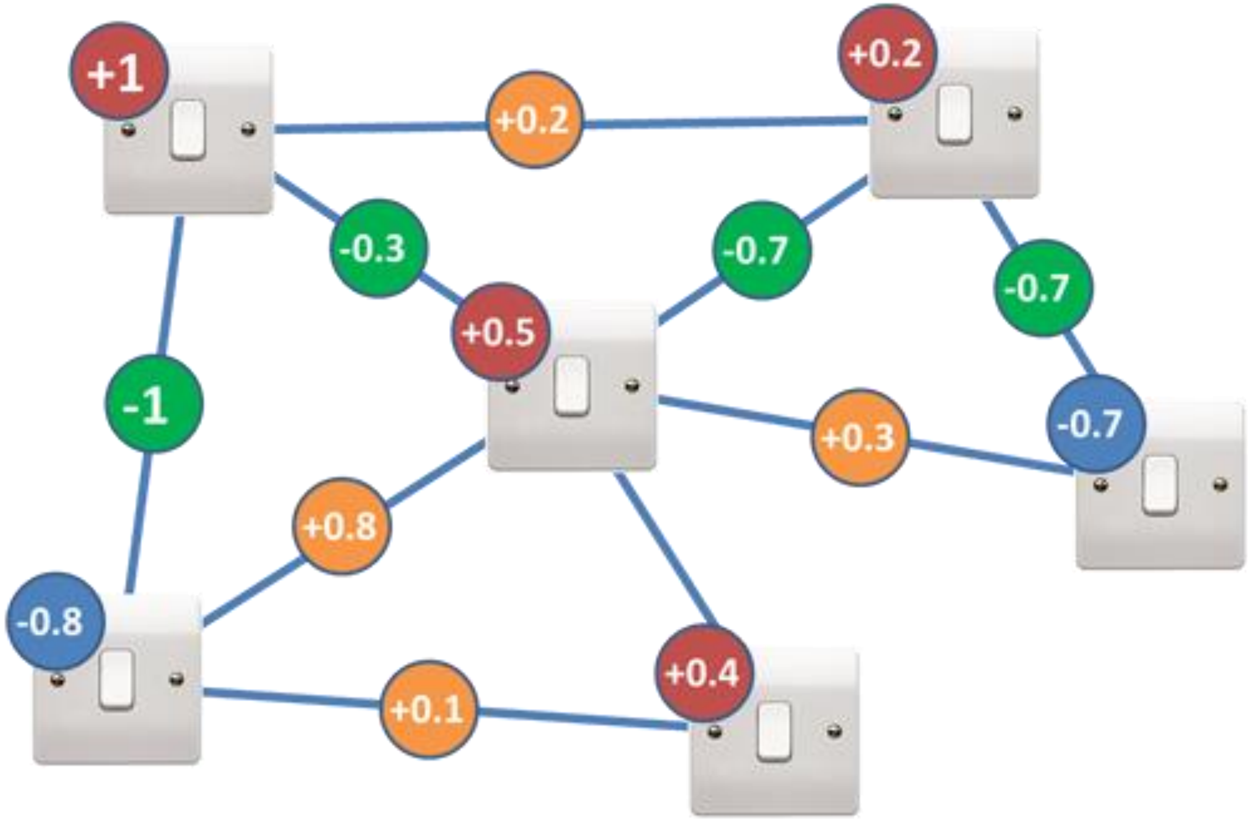
# Game of Switches



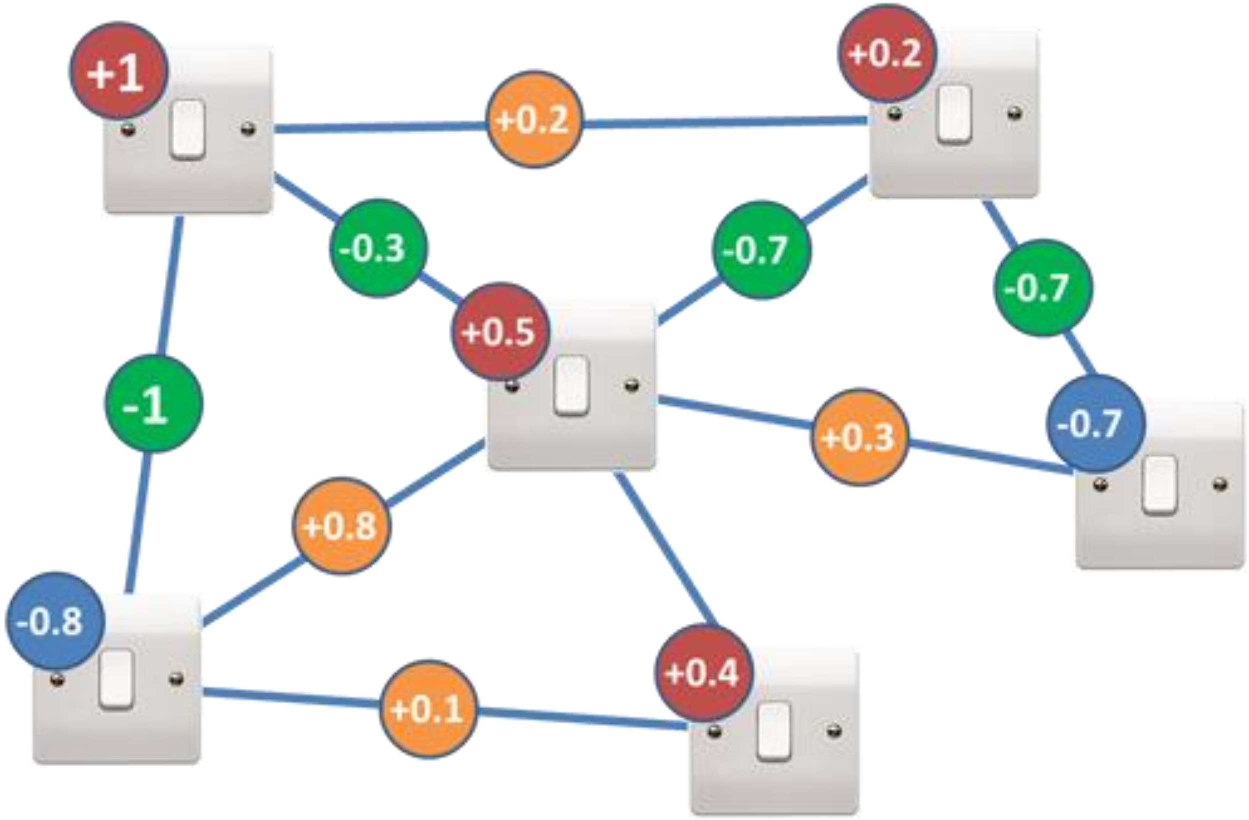

Adding another weight, J, which multiplies the product of the two switch settings.

# Game of Switches



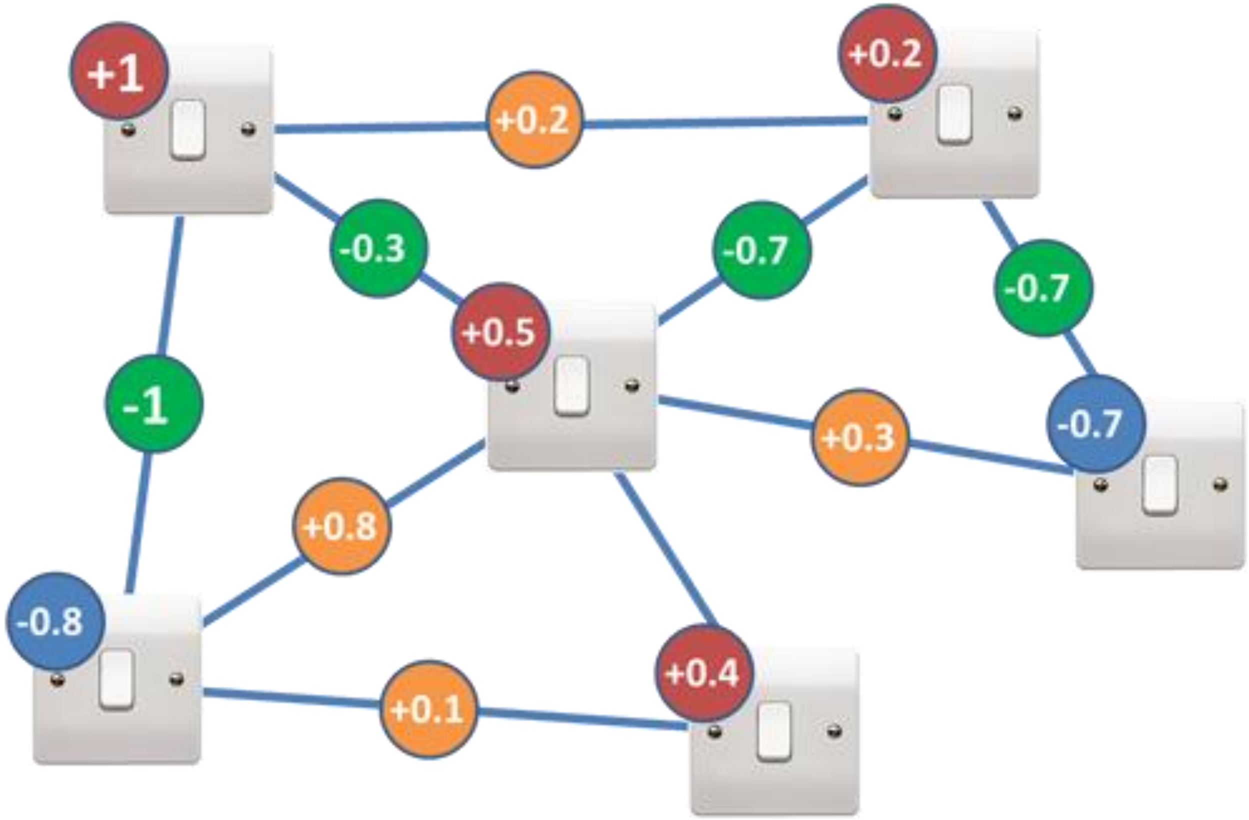

- Let us now consider another factor, namely the presence of couplers between the switches
- Couplers, just like switches, are endowed with a certain numerical weight
- The value of the couplers is given by their own weight multiplied by the state of the switches to which the coupler is associated
- In our case, the coupler will therefore have a state -1 if the two switches it connects are discordant, +1 otherwise

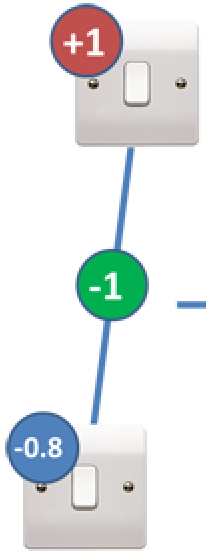
# Game of Switches




- Let us now consider another factor, namely the presence of couplers between the switches
- Couplers, just like switches, are endowed with a certain numerical weight
- The value of the couplers is given by their own weight multiplied by the state of the switches to which the coupler is associated
- In our case, the coupler will therefore have a state -1 if the two switches it connects are discordant, +1 otherwise
- We therefore add to the quantity to be minimized the contribution introduced by the couplers

# Game of Switches

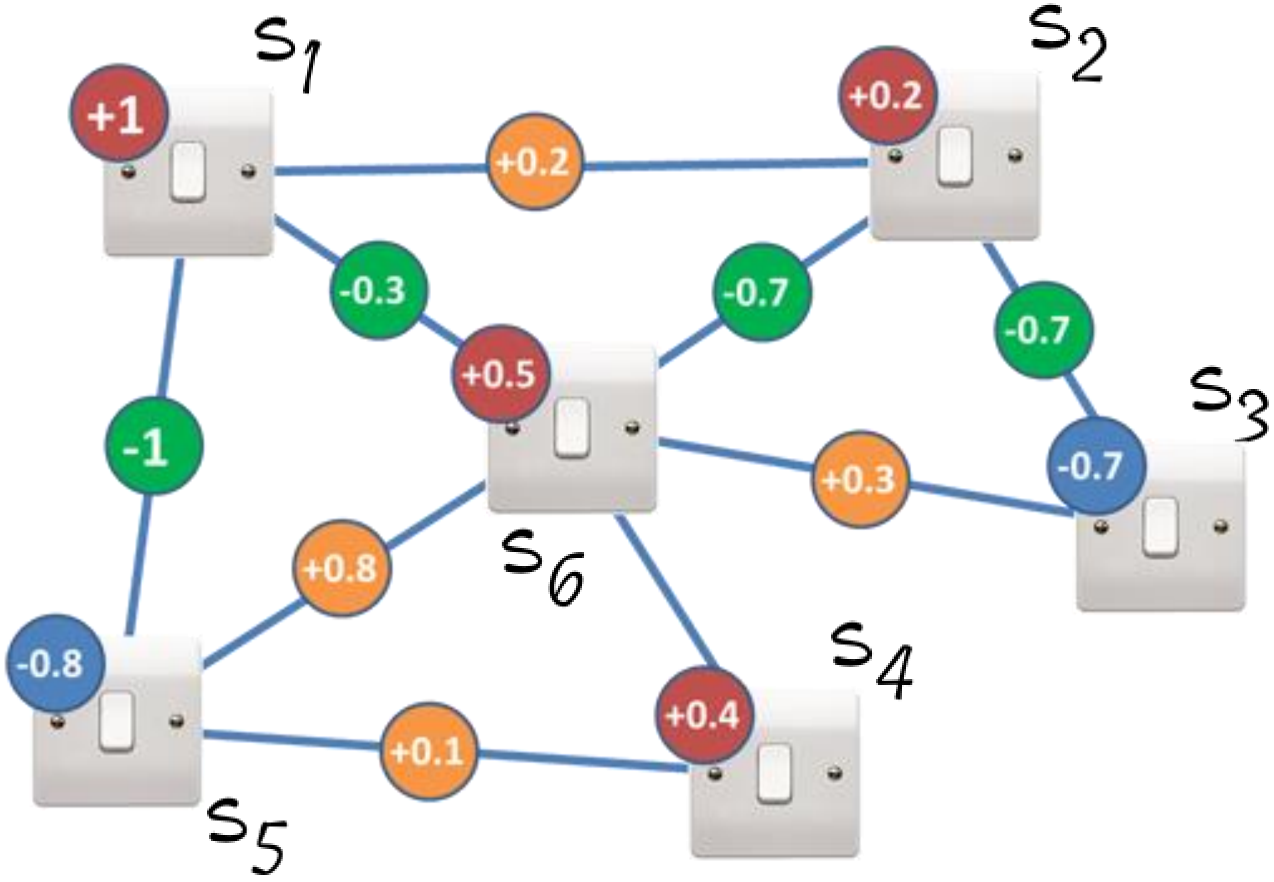



$$E(\mathbf{s}) = \sum_i h_i s_i + \boxed{\sum_{i,j} J_{i,j} s_i s_j}$$

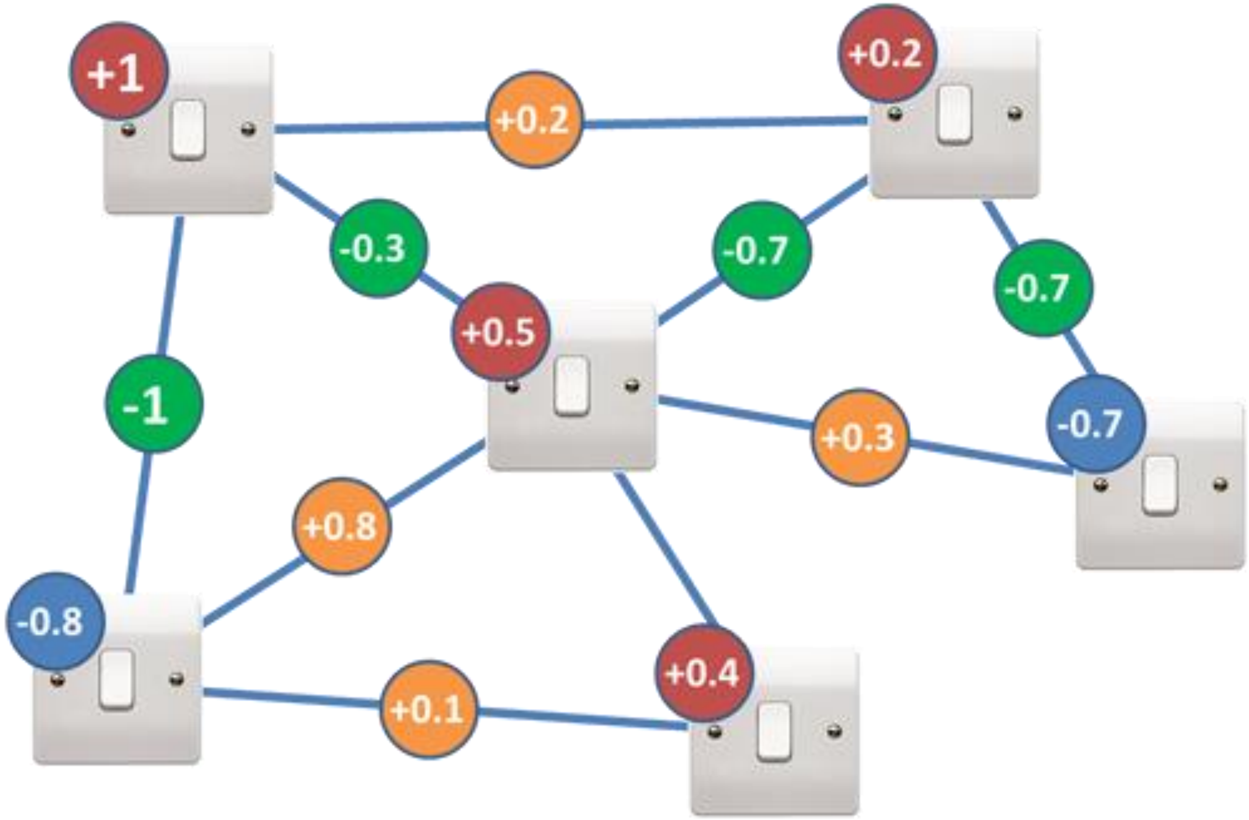
**Adding another weight, J, which multiplies the product of the two switch settings.**

# Game of Switches



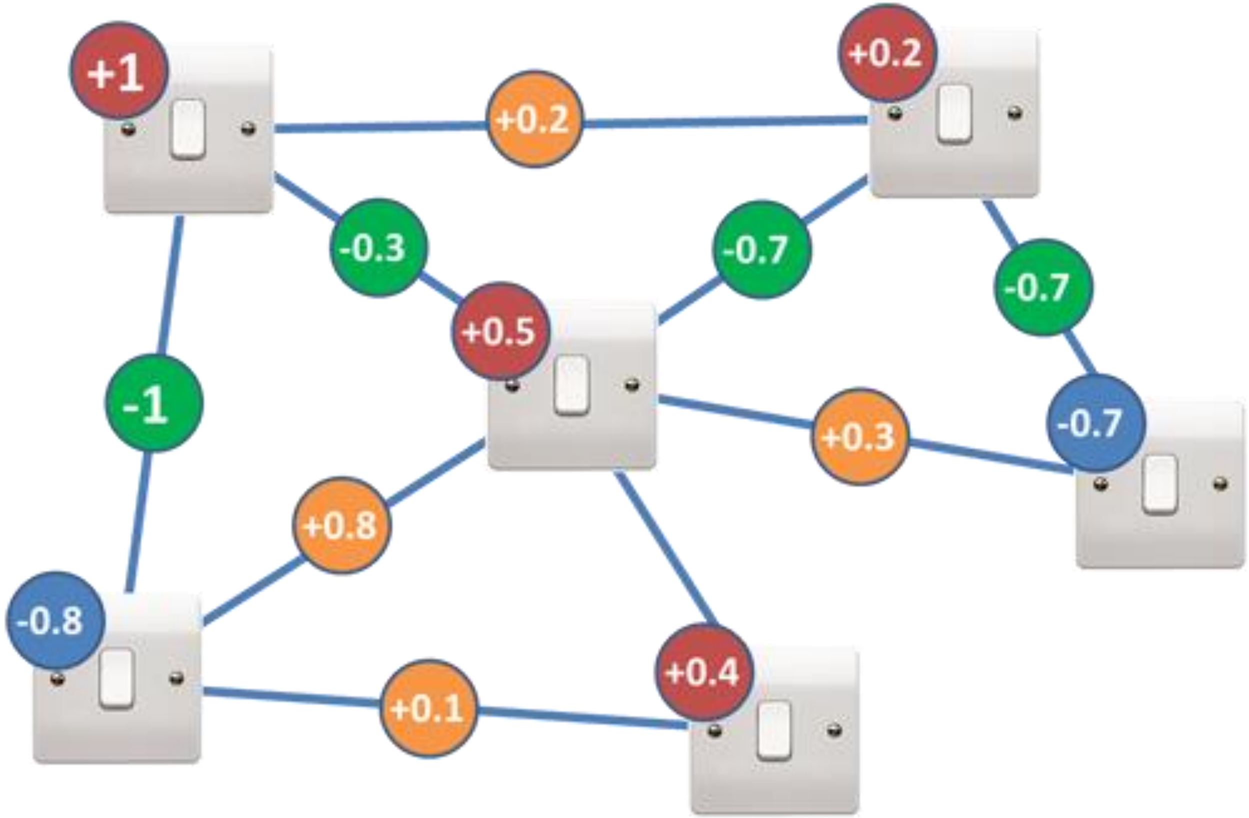

$$s_1 + 0.2s_2 - 0.7s_3 + 0.4s_4 - 0.8s_5 + 0.5s_6 + 0.2s_1s_2 - 0.7s_2s_3 + 0.3s_3s_6 - 0.7s_2s_6 - 0.3s_1s_6 - s_1s_5 + 0.1s_5s_4 + s_6s_4$$

# Game of Switches




2 switches = $2^2$ =
4 possible answers

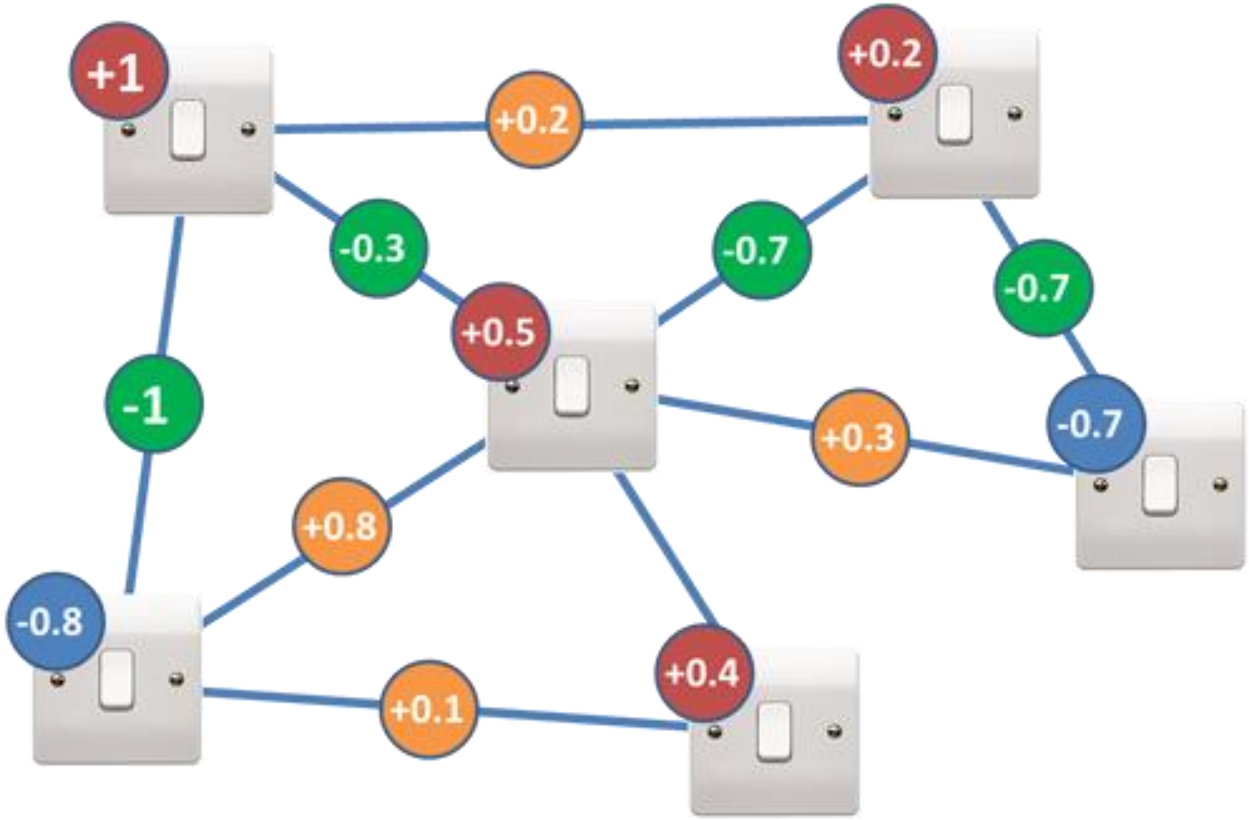
# Game of Switches




2 switches = $2^2$ =
4 possible answers

10 switches = $2^{10}$ =
1024 possible answers

# Game of Switches




2 switches = $2^2$ =
4 possible answers

10 switches = $2^{10}$ =
1024 possible answers

100 switches = $2^{100}$ =
1,267,650,600,228,229,401,496,703,205,376
possible answers

# QUBO Problems

- QUBO (Quadratic Unconstrained Binary Optimization) problems are well known problems in the field of combinatorial optimization.

# 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**.

# 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

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

# 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.

# 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

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

# QUBO Problems

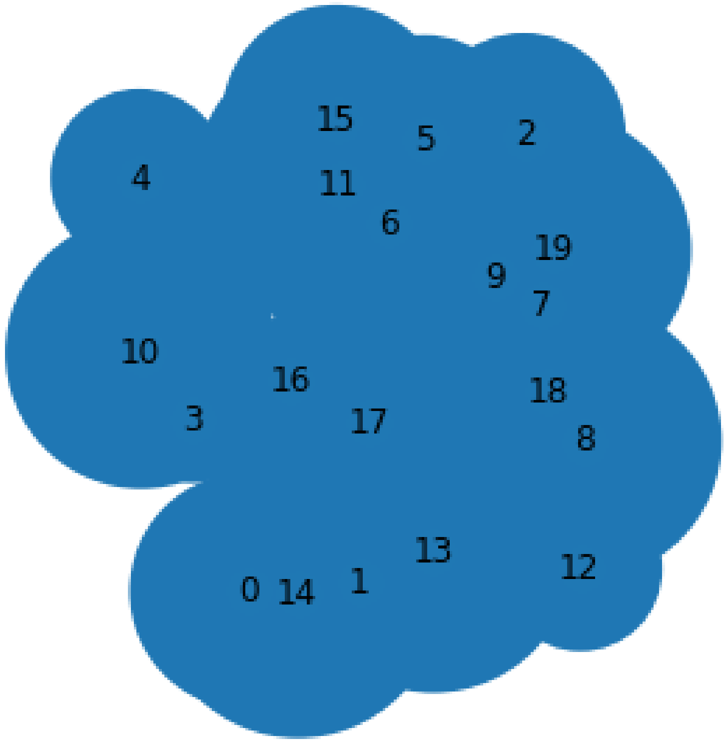
- To familiarize yourself with the QUBO formulation, let's make an example of a realistic problem whose structure can be mapped in this form

# QUBO Problems

- To familiarize yourself with the QUBO formulation, let’s make an example of a realistic problem whose structure can be mapped in this form
- Suppose we have a certain number of antennas and a certain number of possible sites to place these antennas.

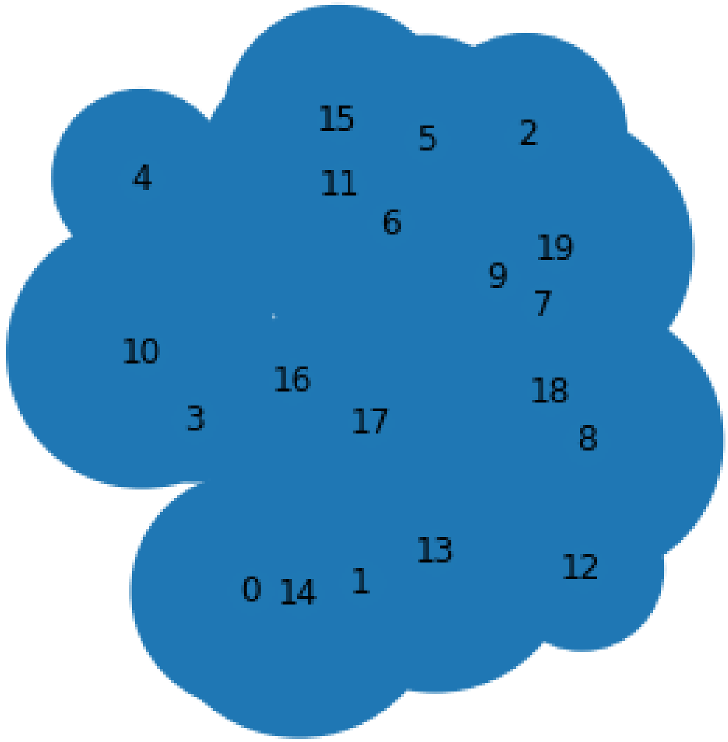
# QUBO Problems

- To familiarize yourself with the QUBO formulation, let's make an example of a realistic problem whose structure can be mapped in this form
- 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



# QUBO Problems

- To familiarize yourself with the QUBO formulation, let's make an example of a realistic problem whose structure can be mapped in this form
- 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

We define:

- 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)

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

# QUBO Problems

We define:

- 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

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

# QUBO Problems

We define:

- 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)

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

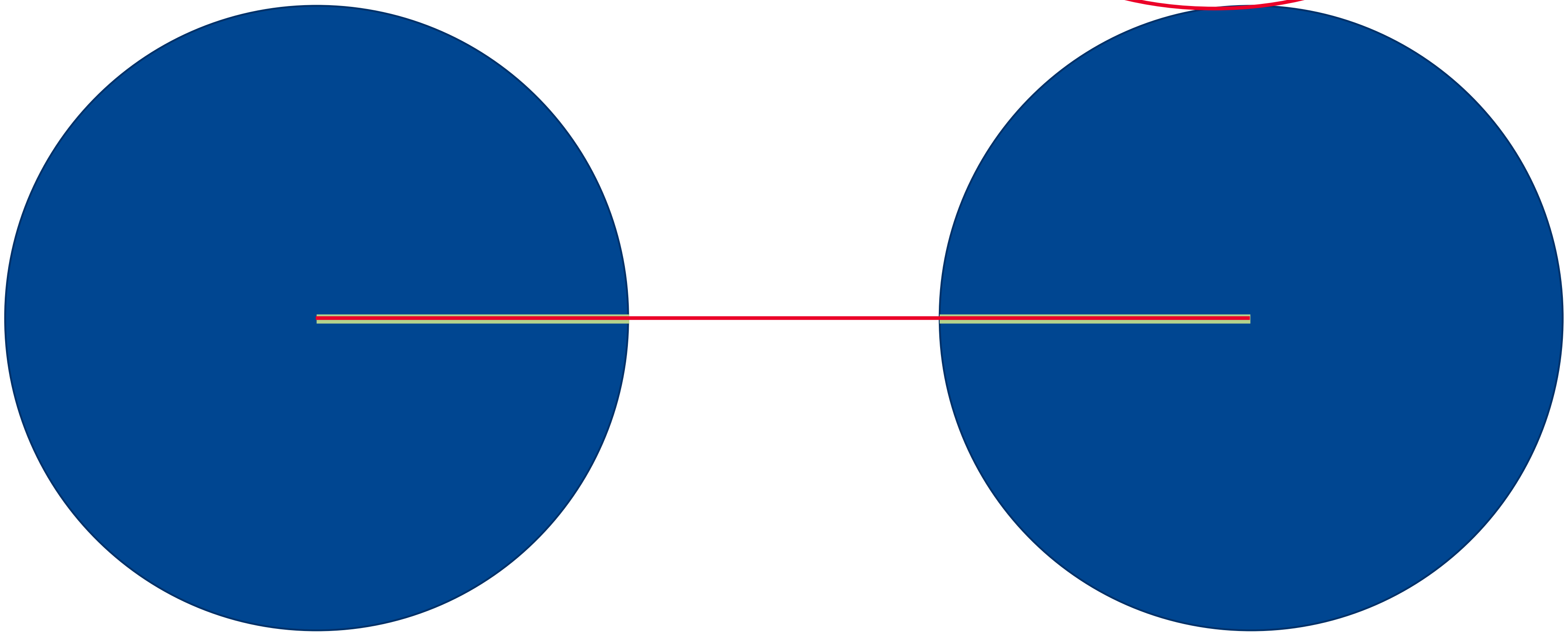
- The interference surface between two antennas as the area of the circle whose radius is given by the following formula

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

- 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

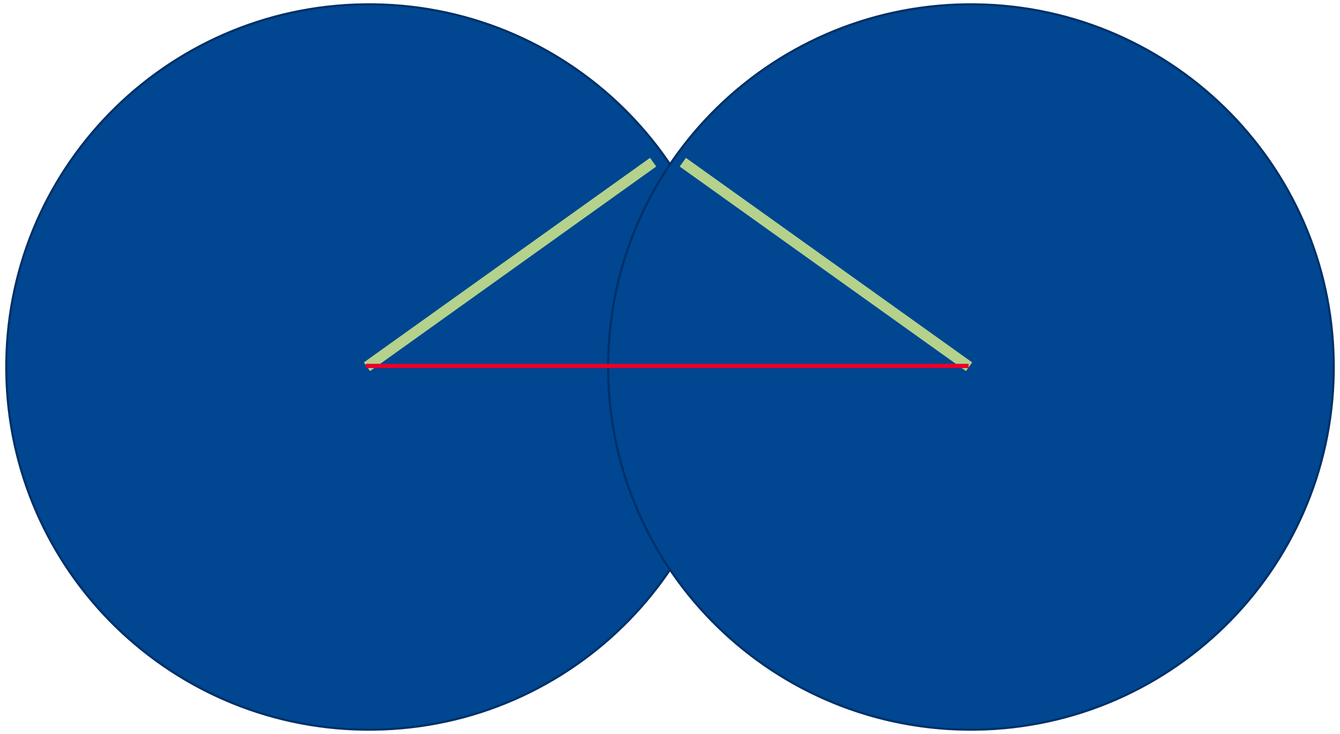
# QUBO Problems

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



# QUBO Problems

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



# 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

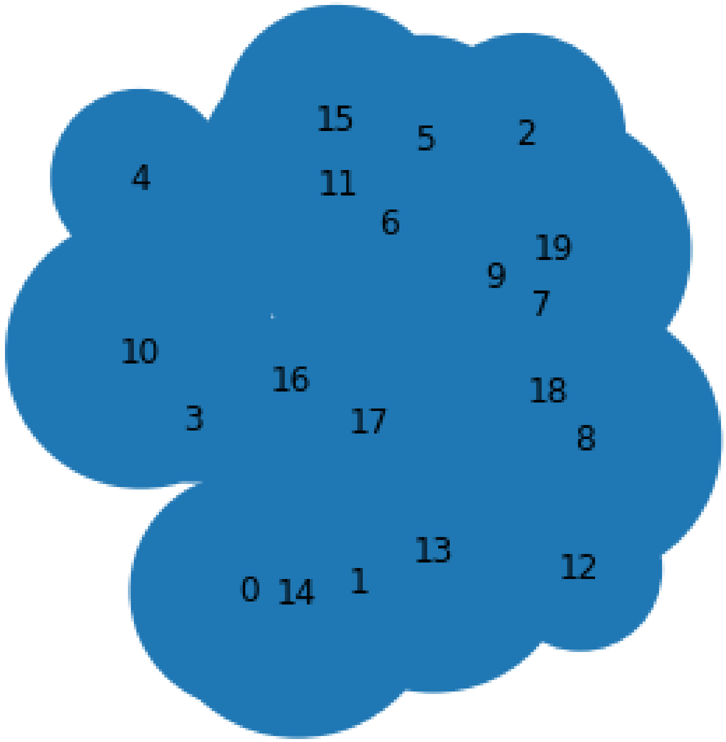
$$B_{ij} = \rho_{ij}^2 \cdot \pi$$

# 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

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

- Now we just have to 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



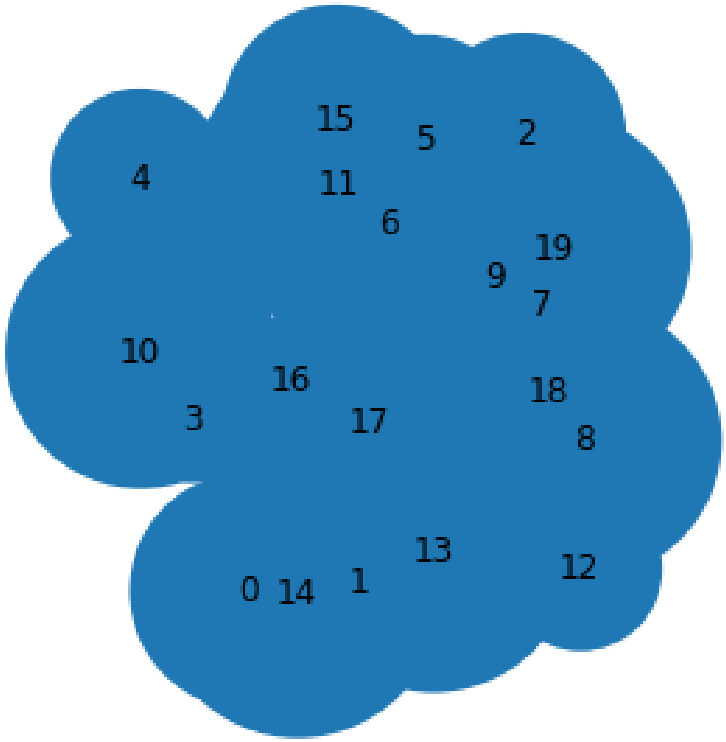
# QUBO Problems

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

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

- Now we just have to 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

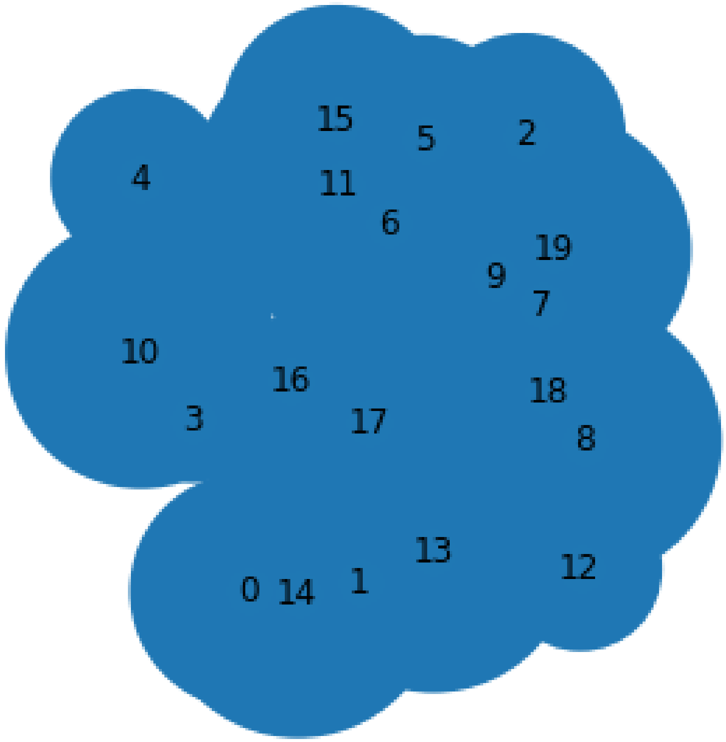
$$[q_0, \dots, q_{19}]$$



# 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

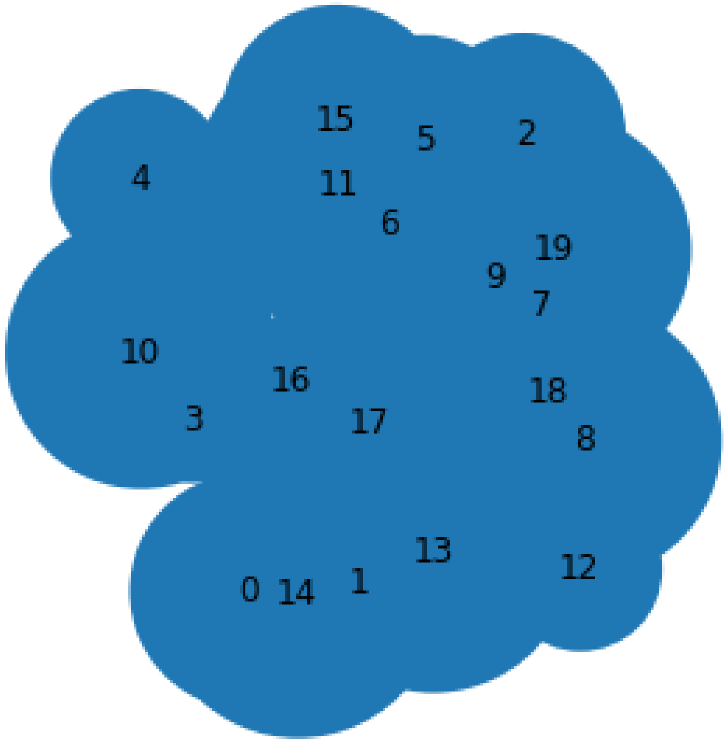
$$A_i = r_i^2 \cdot \pi$$



# 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

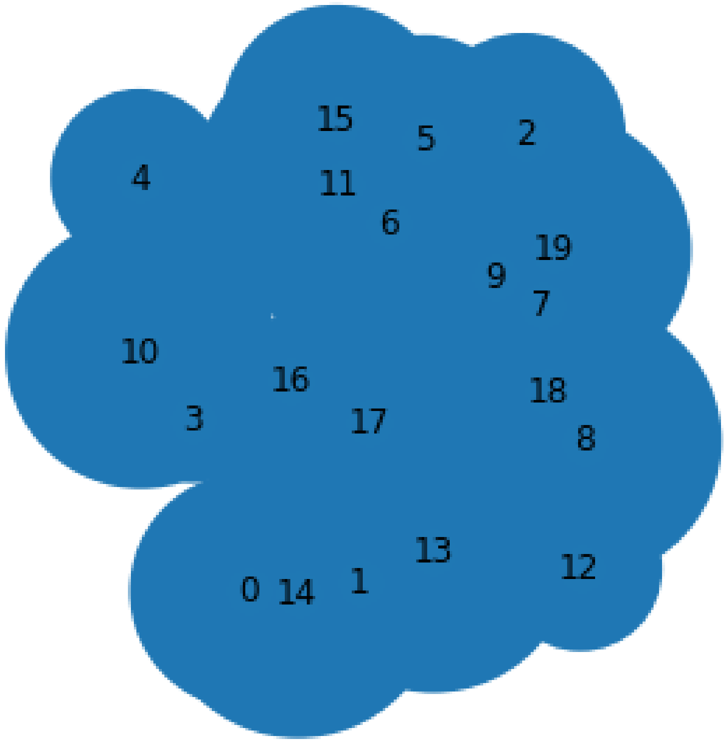
$$A_i = r_i^2 \cdot \pi \qquad B_{ij} = \rho_{ij}^2 \cdot \pi$$



# 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

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




$$QUBO =$$

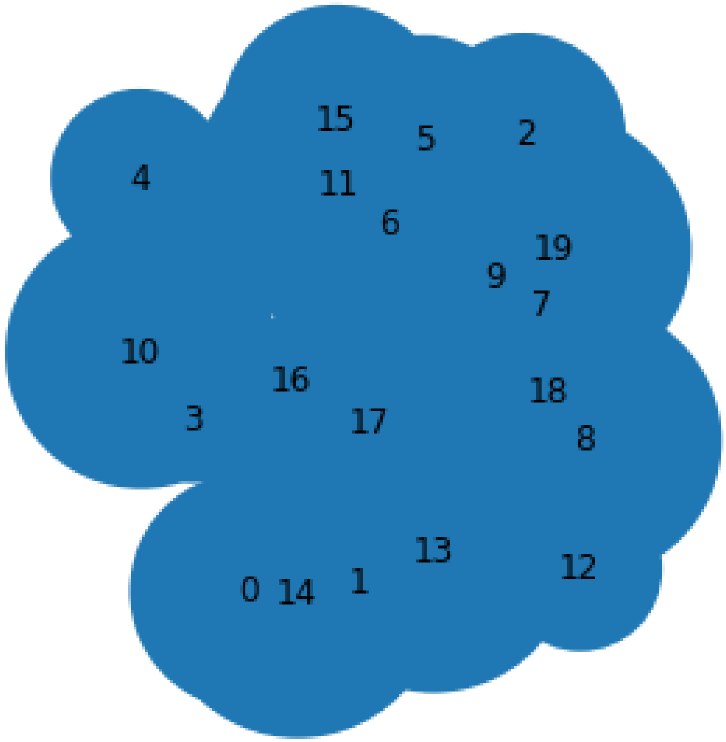
# QUBO Problems

- 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

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

- Minimize interference

$$QUBO = \sum_{i<j} B_{ij} q_i q_j$$



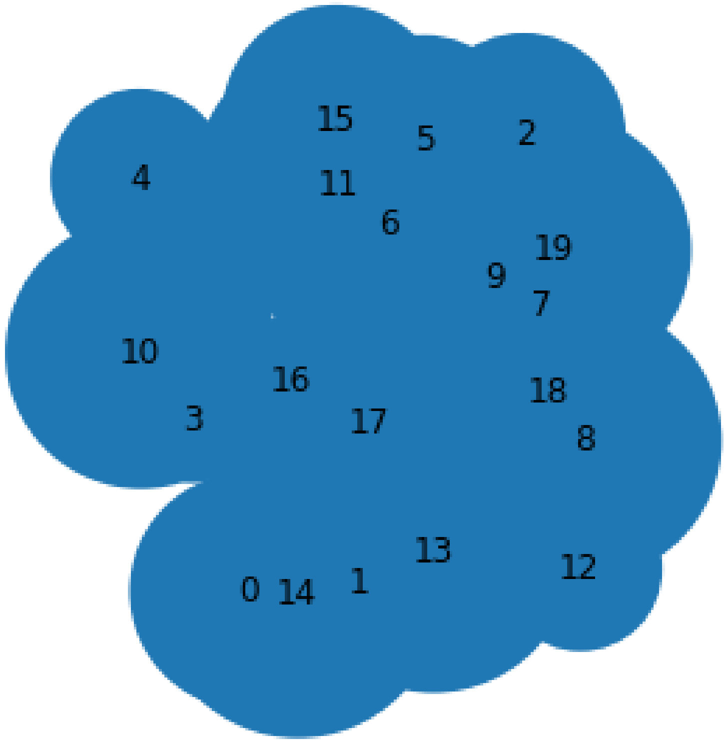
# QUBO Problems

- 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

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



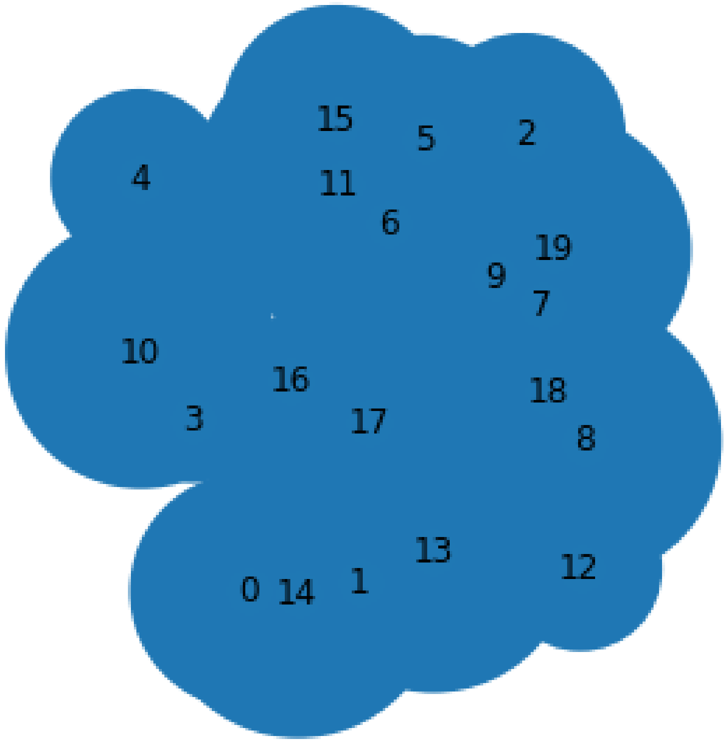
# QUBO Problems

- 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

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

- Maximize covering area

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



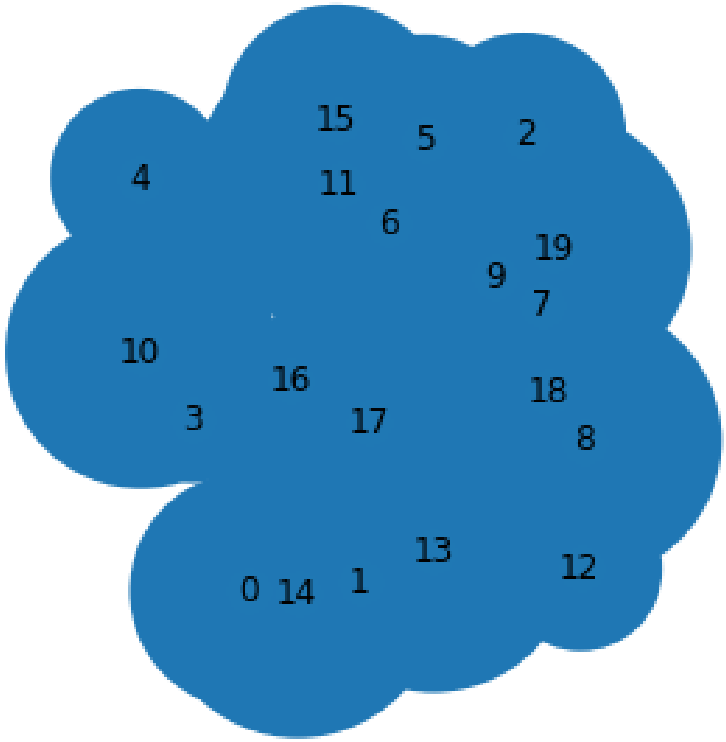
# QUBO Problems

- 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

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



# QUBO/ISING Equivalency

- Any QUBO problem can be easily mapped into an ISING problem through simple equivalence

# QUBO/ISING Equivalency

- Any QUBO problem can be easily mapped into an ISING problem through simple equivalence

$$s_i \mapsto 2x_i - 1$$

$$x_i \mapsto \frac{s_i + 1}{2}$$

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

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

# QUBO/ISING Equivalency

- Any QUBO problem can be easily mapped into an ISING problem through simple equivalence

$$s_i \mapsto 2x_i - 1$$

$$x_i \mapsto \frac{s_i + 1}{2}$$

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

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

- More generally, any mathematical problem can be mapped into a QUBO problem

# QUBO/ISING Equivalency

- Any QUBO problem can be easily mapped into an ISING problem through simple equivalence

$$s_i \mapsto 2x_i - 1$$

$$x_i \mapsto \frac{s_i + 1}{2}$$

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

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

- More generally, any mathematical problem can be mapped into a QUBO problem
- You just have to understand if it's worth it :)

# Programming a Quantum Annealer

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

# Programming a Quantum Annealer

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

- Installation:

```
pip install pyqubo
```

# Programming a Quantum Annealer

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

- Installation:

```
pip install pyqubo
```

- PyQUBO is a very handy utility for writing problems in QUBO or ISING form. Let's see how to use it

# Programming a Quantum Annealer

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

- Installation:

```
pip install pyqubo
```

- PyQUBO is a very handy utility for writing problems in QUBO or ISING form. Let's see how to use it

- Variables: Type Binary (0/1)

```
>>> from pyqubo import Binary
>>> x1, x2 = Binary('x1'), Binary('x2')
>>> H = 2*x1*x2 + 3*x1
>>> pprint(H.compile().to_qubo()) # doctest: +SKIP
({('x1', 'x1'): 3.0, ('x1', 'x2'): 2.0, ('x2', 'x2'): 0.0}, 0.0)
```

# Programming a Quantum Annealer

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

- Installation:

```
pip install pyqubo
```

- PyQUBO is a very handy utility for writing problems in QUBO or ISING form. Let's see how to use it

- Variables: Type Spin (+1/-1)

```
>>> from pyqubo import Spin
>>> s1, s2 = Spin('s1'), Spin('s2')
>>> H = 2*s1*s2 + 3*s1
>>> pprint(H.compile().to_qubo()) # doctest: +SKIP
({('s1', 's1'): 2.0, ('s1', 's2'): 8.0, ('s2', 's2'): -4.0}, -1.0)
```

# Programming a Quantum Annealer

- Arrays of Binary type variables (same for Spin type variables)

```
>>> from pyqubo import Array
>>> x = Array.create('x', shape=(2, 3), vartype='BINARY')
>>> x[0, 1] + x[1, 2]
(Binary(x[0][1])+Binary(x[1][2]))
```

# Programming a Quantum Annealer

- Arrays of Binary type variables (same for Spin type variables)

```
>>> from pyqubo import Array
>>> numbers = [4, 2, 7, 1]
>>> s = Array.create('s', shape=4, vartype='SPIN')
>>> H = sum(n * s for s, n in zip(s, numbers))**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo()
>>> pprint(qubo) # doctest: +SKIP
{('s[0]', 's[0]'): -160.0,
 ('s[0]', 's[1]'): 64.0,
 ('s[0]', 's[2]'): 224.0,
 ('s[0]', 's[3]'): 32.0,
 ('s[1]', 's[1]'): -96.0,
 ('s[1]', 's[2]'): 112.0,
 ('s[1]', 's[3]'): 16.0,
 ('s[2]', 's[2]'): -196.0,
 ('s[2]', 's[3]'): 56.0,
 ('s[3]', 's[3]'): -52.0}
```

# Programming a Quantum Annealer

- Construct a QUBO problem with PyQUBO

```
>>> from pyqubo import Binary
>>> a, b = Binary('a'), Binary('b')
>>> M = 5.0
>>> H = 2*a + b + M*(a+b-1)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo() # QUBO with M=5.0
>>> M = 6.0
>>> H = 2*a + b + M*(a+b-1)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo() # QUBO with M=6.0
```

# Programming a Quantum Annealer

- Construct a QUBO problem with PyQUBO (with Placeholders)

```
>>> from pyqubo import Binary
>>> a, b = Binary('a'), Binary('b')
>>> M = 5.0
>>> H = 2*a + b + M*(a+b-1)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo() # QUBO with M=5.0
>>> M = 6.0
>>> H = 2*a + b + M*(a+b-1)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo() # QUBO with M=6.0
```

```
>>> from pyqubo import Placeholder
>>> a, b = Binary('a'), Binary('b')
>>> M = Placeholder('M')
>>> H = 2*a + b + M*(a+b-1)**2
>>> model = H.compile()
>>> qubo, offset = model.to_qubo(feed_dict={'M': 5.0})
```

# Programming a Quantum Annealer

- Solve a problem set via pyQUBO
- After setting the Hamiltonian of the problem, it must be compiled and transformed into a bqm object

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

# Programming a Quantum Annealer

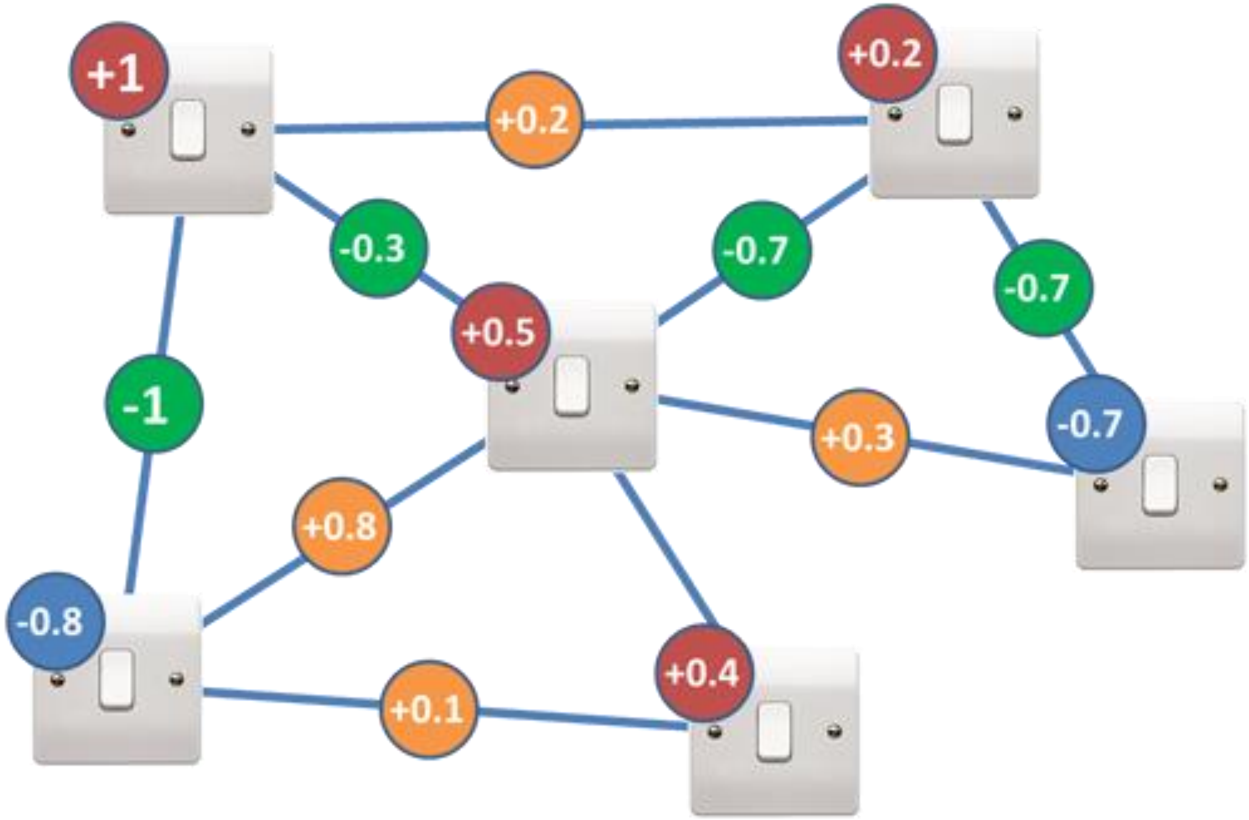
- Solve a problem set via pyQUBO
- After setting the Hamiltonian of the problem, it must be compiled and transformed into a bqm object

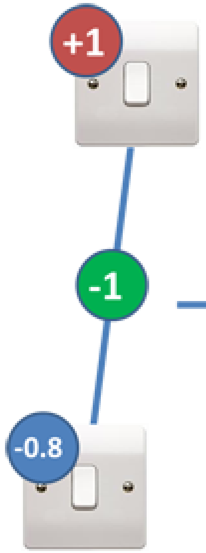
```
>>> from pyqubo import Binary
>>> x1, x2 = Binary('x1'), Binary('x2')
>>> H = (x1 + x2 - 1)**2
>>> model = H.compile()
>>> bqm = model.to_bqm()
```

```
>>> import neal
>>> sa = neal.SimulatedAnnealingSampler()
>>> sampleset = sa.sample(bqm, num_reads=10)
>>> decoded_samples = model.decode_sampleset(sampleset)
>>> best_sample = min(decoded_samples, key=lambda x: x.energy)
>>> pprint(best_sample.sample)
{'x1': 0, 'x2': 1}
```

# Exercise 1: Game of Switches

- Try to implement the Game of Switches




$$E(\mathbf{s}) = \sum_i h_i s_i + \boxed{\sum_{i,j} J_{i,j} s_i s_j}$$

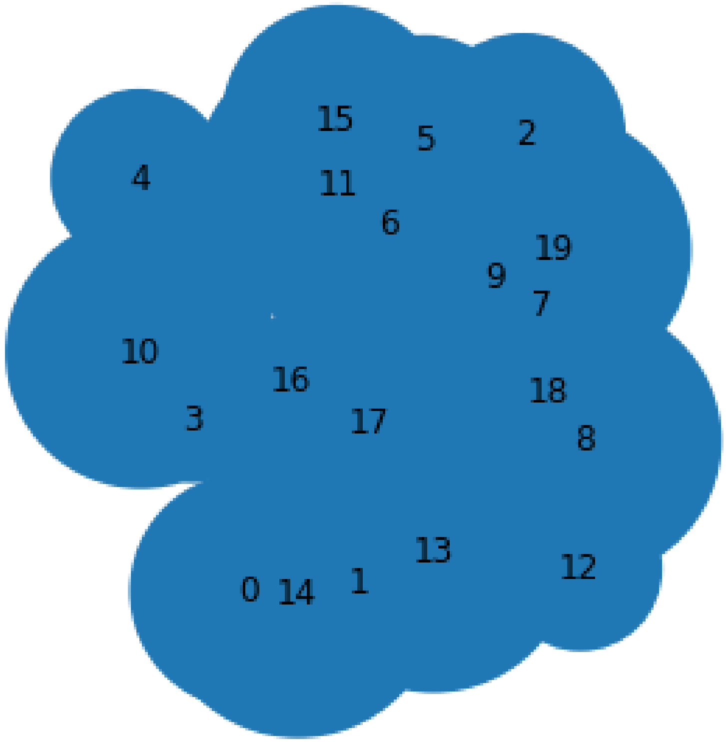
**Adding another weight, J, which multiplies the product of the two switch settings.**

# Exercise 2: Antenna Placement

- Try to implement the Antenna Placement Problem

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

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



# Add a Constraint to a QUBO Problem

- By definition, a QUBO problem admits no constraints

**Q**uadratic
**U**nconstrained
**B**inary
**O**ptimization

- Still, there is a way.

# Add a Constraint to a QUBO Problem

- Let's see how to implement a linear constraint in a QUBO problem.

# Add a Constraint to a QUBO Problem

- Let's see how to implement a linear constraint in a QUBO problem.
- Everything relies around the concept of **penalty function**

# Add a Constraint to a QUBO Problem

- Let's see how to implement a linear constraint in a QUBO problem.
- Everything relies around the concept of **penalty function**
- A penalty function is in fact an **additional quantity** to the original minimization problem, **which must be optimized** in order for the entire problem to be optimized

# Add a Constraint to a QUBO Problem

- Let's see how to implement a linear constraint in a QUBO problem.
- Everything relies around the concept of **penalty function**
- A penalty function is in fact an **additional quantity** to the original minimization problem, **which must be optimized** in order for the entire problem to be optimized
- Suppose we want to add the following constraint to our antenna optimization problem

- *Let F be the exact number of antennas to be placed*

# Add a Constraint to a QUBO Problem

- Let's see how to implement a linear constraint in a QUBO problem.
- Everything relies around the concept of **penalty function**
- A penalty function is in fact an **additional quantity** to the original minimization problem, **which must be optimized** in order for the entire problem to be optimized
- Suppose we want to add the following constraint to our antenna optimization problem

- *Let F be the exact number of antennas to be placed*

- Remembering the mathematical formulation of our problem, requested constraint can be seen as

$$\sum_{i=0}^{N} q_i = F$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min \left( \sum_{i=0}^{N} q_i - F \right)^2$$

$$\left( \sum_{i=0}^{N} q_i - F \right)^2$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\left(\sum_{i=0}^{N} q_i - F\right)^2 = \left(\sum_{i=0}^{N} q_i\right)^2 + F^2 - 2F\sum_{i=0}^{N} q_i$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\left(\sum_{i=0}^{N} q_i - F\right)^2 = \left(\sum_{i=0}^{N} q_i\right)^2 + \cancel{F^2} - 2F\sum_{i=0}^{N} q_i = \left(\sum_{i=0}^{N} q_i\right)^2 - 2F\sum_{i=0}^{N} q_i =$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\left(\sum_{i=0}^{N} q_i - F\right)^2 = \left(\sum_{i=0}^{N} q_i\right)^2 + \cancel{F^2} - 2F\sum_{i=0}^{N} q_i = \left(\sum_{i=0}^{N} q_i\right)^2 - 2F\sum_{i=0}^{N} q_i =$$

$$= \sum_{i=0}^{N} q_i^2 + 2\sum_{i<j} q_i q_j - 2F\sum_{i=0}^{N} q_i$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\left(\sum_{i=0}^{N} q_i - F\right)^2 = \left(\sum_{i=0}^{N} q_i\right)^2 + \cancel{F^2} - 2F\sum_{i=0}^{N} q_i = \left(\sum_{i=0}^{N} q_i\right)^2 - 2F\sum_{i=0}^{N} q_i =$$

$$= \sum_{i=0}^{N} q_i^2 + 2\sum_{i<j} q_i q_j - 2F\sum_{i=0}^{N} q_i = \sum_{i=0}^{N} q_i + 2\sum_{i<j} q_i q_j - 2F\sum_{i=0}^{N} q_i =$$

# Add a Constraint to a QUBO Problem

- Let's do some math

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\left(\sum_{i=0}^{N} q_i - F\right)^2 = \left(\sum_{i=0}^{N} q_i\right)^2 + \cancel{F^2} - 2F\sum_{i=0}^{N} q_i = \left(\sum_{i=0}^{N} q_i\right)^2 - 2F\sum_{i=0}^{N} q_i =$$

$$= \sum_{i=0}^{N} q_i^2 + 2\sum_{i<j} q_i q_j - 2F\sum_{i=0}^{N} q_i = \sum_{i=0}^{N} q_i + 2\sum_{i<j} q_i q_j - 2F\sum_{i=0}^{N} q_i =$$

$$= \sum_{i=0}^{N} (1 - 2F)\, q_i + \sum_{i<j} 2 q_i q_j$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min \left( \sum_{i=0}^{N} q_i - F \right)^2$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min\left(\sum_{i=0}^{N} q_i - F\right)^2$$

$$\min\left(B \cdot \left(\sum_{i=0}^{N} (1 - 2F)\, q_i + \sum_{i<j} 2 q_i q_j\right)\right)$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=0}^{N} q_i = F \quad \Rightarrow \quad \min \left( \sum_{i=0}^{N} q_i - F \right)^2$$

$$\min \left( \left( \sum_{i=0}^{N} B(1 - 2F) q_i + \sum_{i<j} 2B q_i q_j \right) \right)$$

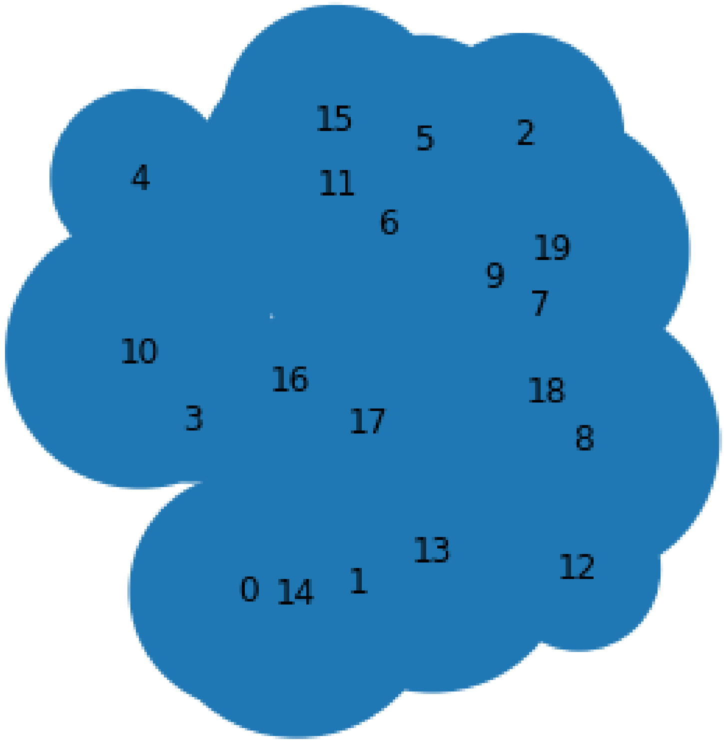
# Exercise 2: Antenna Placement

- Implement constraint into the Antenna Placement Problem

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

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

$$\min\left(\left(\sum_{i=0}^{N} \beta(1 - 2F) q_i + \sum_{i<j} 2\beta q_i q_j\right)\right)$$



# Add a Constraint to a QUBO Problem

- Now suppose we want to add another constraint.

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another constraint.
- For some reason, we have received orders from above telling us that **certain antennas must be placed, regardless of any other conditions**.

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another constraint.
- For some reason, we have received orders from above telling us that **certain antennas must be placed, regardless of any other conditions.**
- How can we implement this type of request?

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another constraint.
- For some reason, we have received orders from above telling us that **certain antennas must be placed, regardless of any other conditions.**
- How can we implement this type of request?

- First of all we consider a vector L, of length equal to the number of antennas available. We mark with 0 the free antennas and with 1 the antennas that must necessarily be activated.

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another constraint.
- For some reason, we have received orders from above telling us that certain antennas must be placed, regardless of any other conditions.
- How can we implement this type of request?

- First of all we consider a vector L, of length equal to the number of antennas available. We mark with 0 the free antennas and with 1 the antennas that must necessarily be activated.

- Consequently, penalty function can be seen as

$$\sum_{i=1}^{N} L_i (q_i - 1)^2$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{N} L_i \left( q_i - 1 \right)^2$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{N} L_i (q_i - 1)^2 = \sum_{i=1}^{N} L_i q_i + \sum_{i=1}^{N} L_i - \sum_{i=1}^{N} 2 L_i q_i =$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{N} L_i (q_i - 1)^2 = \sum_{i=1}^{N} L_i q_i + \cancel{\sum_{i=1}^{N} L_i} - \sum_{i=1}^{N} 2 L_i q_i =$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{N} L_i (q_i - 1)^2 = \sum_{i=1}^{N} L_i q_i + \cancel{\sum_{i=1}^{N} L_i} - \sum_{i=1}^{N} 2 L_i q_i =$$

$$- \sum_{i=1}^{N} L_i q_i$$

# Exercise 2: Antenna Placement

- Implement constraint into the Antenna Placement Problem

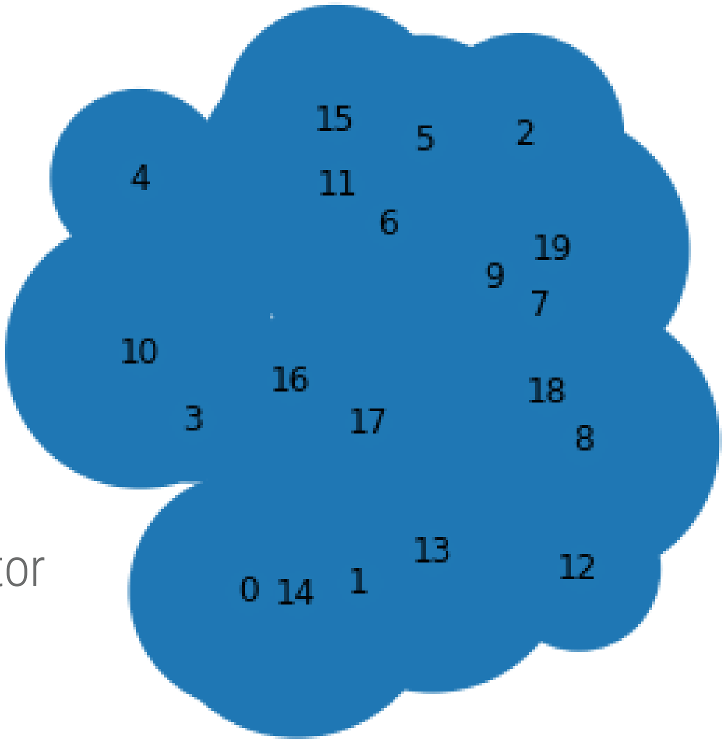
$$A_i = r_i^2 \cdot \pi$$

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

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

$$\min\left(\left(\sum_{i=0}^{N} \beta\,(1-2F)\, q_i + \sum_{i<j} 2\beta\, q_i q_j\right)\right)$$

$$-\sum_{i=1}^{N} \gamma L_i q_i$$




- Implement and configure L vector
- Add values to QUBO problem formulation

# Add a Constraint to a QUBO Problem

- Now suppose we want to add an **inequality** constraint to our problem.

# Add a Constraint to a QUBO Problem

- Now suppose we want to add an **inequality** constraint to our problem.
- An example could be

# Add a Constraint to a QUBO Problem

- Now suppose we want to add an **inequality** constraint to our problem.
- An example could be

- *Let F be the maximum number of antennas that can be placed*

# Add a Constraint to a QUBO Problem

- Now suppose we want to add an **inequality** constraint to our problem.
- An example could be
- *Let F be the maximum number of antennas that can be placed*
- Mathematically, the constraint appears in the form

$$\sum_{i=0}^{N} q_i \leq F$$

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?
- One way can be to **reduce inequality to equality**

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?
- One way can be to **reduce inequality to equality**
- To do that, we need additional binary variables. For this interpretation, we need F more variables

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?
- One way can be to **reduce inequality to equality**
- To do that, we need additional binary variables. For this interpretation, we need F more variables

$$\sum_{i=1}^{N} q_i \leq F \quad \Rightarrow \quad \sum_{i=1}^{N} q_i = F - \sum_{k=1}^{F} q_k$$

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?
- One way can be to **reduce inequality to equality**
- To do that, we need additional binary variables. For this interpretation, we need F more variables

$$\sum_{i=1}^{N} q_i \leq F \quad \Rightarrow \quad \sum_{i=1}^{N} q_i = F - \sum_{k=1}^{F} q_k$$

$$\sum_{i=1}^{N} q_i + \sum_{k=1}^{F} q_i = F$$

# Add a Constraint to a QUBO Problem

- So far we have seen how to transform constraints involving equalities into penalty functions
- How to deal with an inequality?
- One way can be to **reduce inequality to equality**
- To do that, we need additional binary variables. For this interpretation, we need F more variables

$$\sum_{i=1}^{N} q_i \leq F \quad \Rightarrow \quad \sum_{i=1}^{N} q_i = F - \sum_{k=1}^{F} q_k$$

$$\sum_{i=1}^{N} q_i + \sum_{k=1}^{F} q_i = F \quad \Rightarrow \quad \sum_{i=1}^{N+F} q_i = F$$

# Exercise 2: Antenna Placement

- Implement constraint into the Antenna Placement Problem

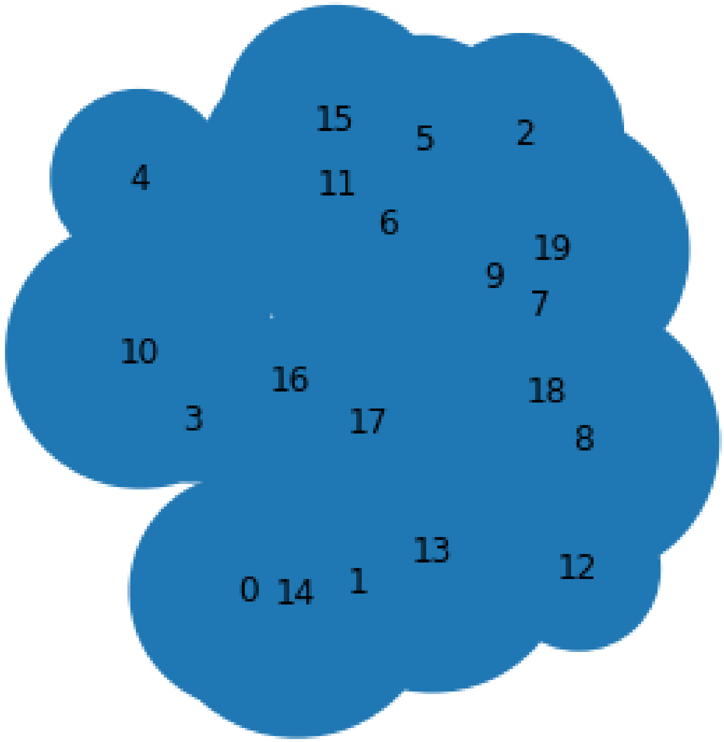
$$A_i = r_i^2 \cdot \pi \qquad B_{ij} = \rho_{ij}^2 \cdot \pi$$

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

$$\min\left(\left(\sum_{i=0}^{N} \beta(1-2F) q_i + \sum_{i<j} 2\beta q_i q_j\right)\right) - \sum_{i=1}^{N} \gamma L_i q_i$$

$$\sum_{i=1}^{N+F} q_i = F$$

- Add F more qubits to the formulation
- These qubits are a sort of ghost qubits: they MUST don't interact with the other part of the problem formulation



# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem
- For example:

*Let* $A_m$ *be a measure of area. Turn on the antennas so that the minimum covered area is greater than or equal to* $A_m$

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem
- For example:

  *Let $A_m$ be a measure of area. Turn on the antennas so that the minimum covered area is greater than or equal to $A_m$*

$$\sum_{i=1}^{N} A_i q_i \geq A_m$$

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem
- For example:

*Let $A_m$ be a measure of area. Turn on the antennas so that the minimum covered area is greater than or equal to $A_m$*

$$\sum_{i=1}^{N} A_i q_i \geq A_m \quad \Rightarrow \quad \sum_{i=1}^{N} A_i q_i = A_m + \sum_{k=1}^{N} A_k q_k$$

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem
- For example:

*Let $A_m$ be a measure of area. Turn on the antennas so that the minimum covered area is greater than or equal to $A_m$*

$$\sum_{i=1}^{N} A_i q_i \geq A_m \quad \Rightarrow \quad \sum_{i=1}^{N} A_i q_i = A_m + \sum_{k=1}^{N} A_k q_k$$

$$\sum_{i=1}^{N} A_i q_i - \sum_{k=1}^{N} q_k = A_m$$

# Add a Constraint to a QUBO Problem

- Now suppose we want to add another inequality constraint to our problem
- For example:

*Let $A_m$ be a measure of area. Turn on the antennas so that the minimum covered area is greater than or equal to $A_m$*

$$\sum_{i=1}^{N} A_i q_i \geq A_m \quad \Rightarrow \quad \sum_{i=1}^{N} A_i q_i = A_m + \sum_{k=1}^{N} A_k q_k$$

$$\sum_{i=1}^{N} A_i q_i - \sum_{k=1}^{N} q_k = A_m \quad \Rightarrow \quad \sum_{i=1}^{2N} c_i A_i q_i = A_m$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m$$

CINECA QUANTUM COMPUTING LAB

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m \Rightarrow \left( \sum_{i=1}^{2N} C_i A_i q_i - A_m \right)^2$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m \Rightarrow \left(\sum_{i=1}^{2N} C_i A_i q_i - A_m\right)^2 = \left(\sum_{i=1}^{2N} C_i A_i q_i\right)^2 - 2\sum_{i=1}^{2N} A_m C_i A_i q_i$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m \Rightarrow \left( \sum_{i=1}^{2N} C_i A_i q_i - A_m \right)^2 = \left( \sum_{i=1}^{2N} C_i A_i q_i \right)^2 - 2\sum_{i=1}^{2N} A_m C_i A_i q_i$$

$$\sum_{i=1}^{2N} A_i^2 q_i + 2\sum_{i<j} C_i C_j A_i A_j q_i - 2\sum_{i=1}^{2N} A_m C_i A_i q_i$$

# Add a Constraint to a QUBO Problem

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m \Rightarrow \left(\sum_{i=1}^{2N} C_i A_i q_i - A_m\right)^2 = \left(\sum_{i=1}^{2N} C_i A_i q_i\right)^2 - 2\sum_{i=1}^{2N} A_m C_i A_i q_i$$

$$\sum_{i=1}^{2N} A_i^2 q_i + 2\sum_{i<j} C_i C_j A_i A_j q_i - 2\sum_{i=1}^{2N} A_m C_i A_i q_i = \sum_{i=1}^{2N} \left(A_i^2 - 2A_m C_i A_i\right) q_i + \sum_{i<j} 2 C_i C_j A_i A_j q_i$$

CINECA QUANTUM COMPUTING LAB

# Exercise 2: Antenna Placement

- Implement constraint into the Antenna Placement Problem

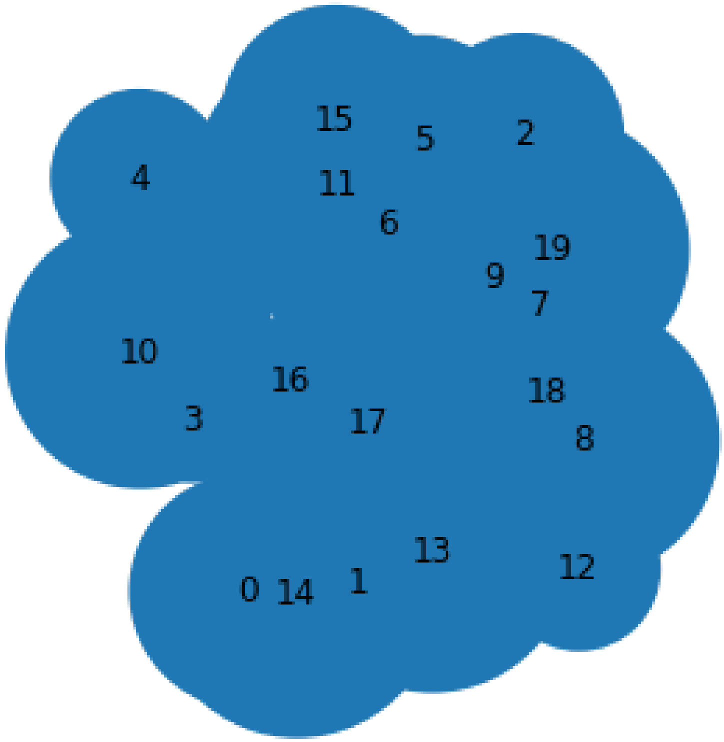
$$A_i = r_i^2 \cdot \pi \qquad B_{ij} = \rho_{ij}^2 \cdot \pi$$

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

$$\min\left(\left(\sum_{i=0}^{N} \beta (1-2F) q_i + \sum_{i<j} 2\beta q_i q_j\right)\right) - \sum_{i=1}^{N} \gamma L_i q_i \qquad \sum_{i=1}^{N+F} q_i = F$$

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m$$

- Add N more qubits to the formulation
- These qubits are a sort of ghost qubits: they don't interact with the other part of the problem formulation
- Do the math!



# Add High Order terms to our problem

- Sometimes it is necessary to add some terms of order 3 or higher to our problem.

# Add High Order terms to our problem

- Sometimes it is necessary to add some terms of order 3 or higher to our problem.
- How can we relate to a QUBO problem?

# Add High Order terms to our problem

- Sometimes it is necessary to add some terms of order 3 or higher to our problem.
- How can we relate to a QUBO problem?

$$xyz = \max_{w} \{w(x + y + z - 2)\}$$

# Add High Order terms to our problem

- Sometimes it is necessary to add some terms of order 3 or higher to our problem.
- How can we relate to a QUBO problem?

$$xyz = \max_{w} \{w(x+y+z-2)\}$$

| $\mathbf{x,y,z}$ | $\mathbf{xyz}$ | $\mathbf{x+y+z-2}$ | $\max_{\mathbf{w}} \{\mathbf{w(x+y+z-2)}\}$ |
|---|---|---|---|
| $0,0,0$ | $0$ | $-2$ | $0\|_{w=0}$ |
| $0,0,1$ | $0$ | $-1$ | $0\|_{w=0}$ |
| $0,1,0$ | $0$ | $-1$ | $0\|_{w=0}$ |
| $0,1,1$ | $0$ | $0$ | $0\|_{w=0,1}$ |
| $1,0,0$ | $0$ | $-1$ | $0\|_{w=0}$ |
| $1,0,1$ | $0$ | $0$ | $0\|_{w=0,1}$ |
| $1,1,0$ | $0$ | $0$ | $0\|_{w=0,1}$ |
| $1,1,1$ | $1$ | $1$ | $1\|_{w=1}$ |

# Exercise 2: Antenna Placement

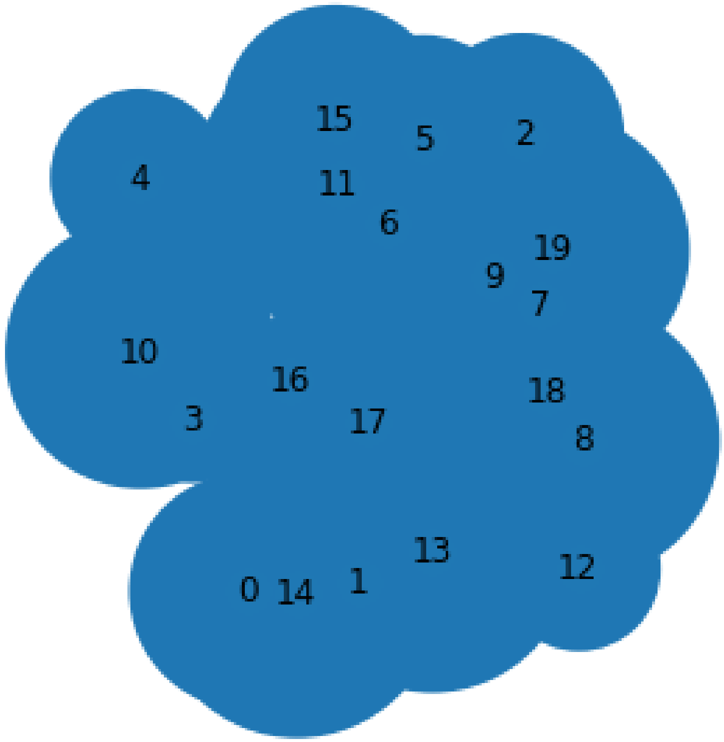
- Implement constraint into the Antenna Placement Problem

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

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

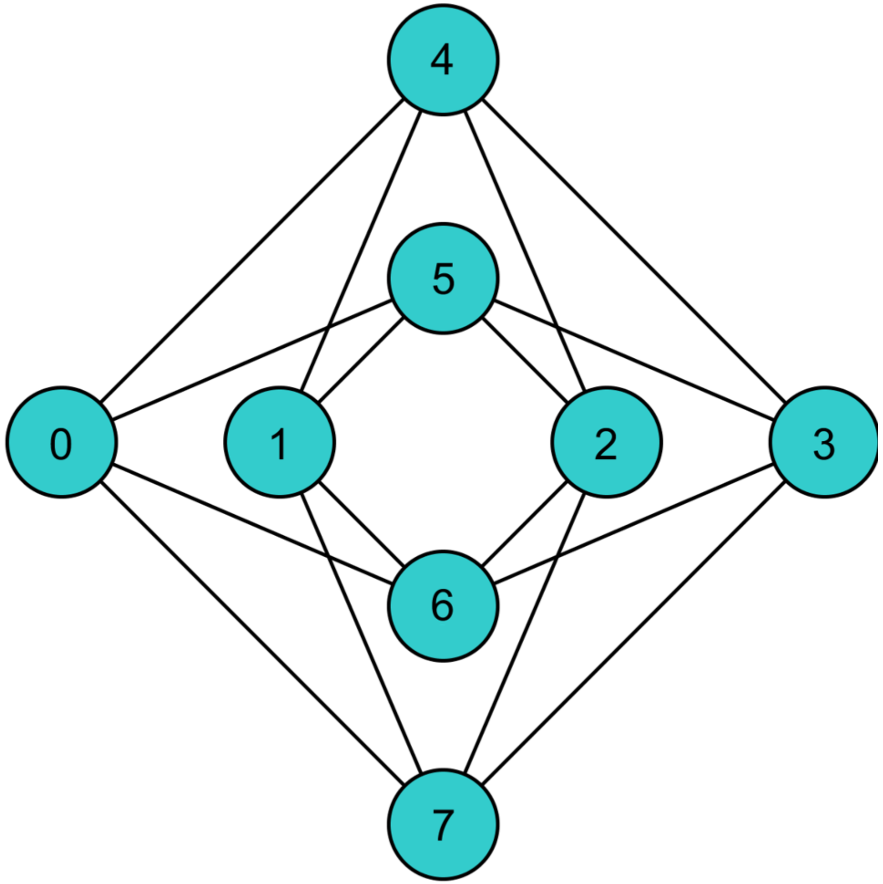
$$\min\left(\left(\sum_{i=0}^{N} \beta(1-2F) q_i + \sum_{i<j} 2\beta q_i q_j\right)\right) - \sum_{i=1}^{N} \gamma L_i q_i \qquad \sum_{i=1}^{N+F} q_i = F$$

$$\sum_{i=1}^{2N} C_i A_i q_i = A_m$$



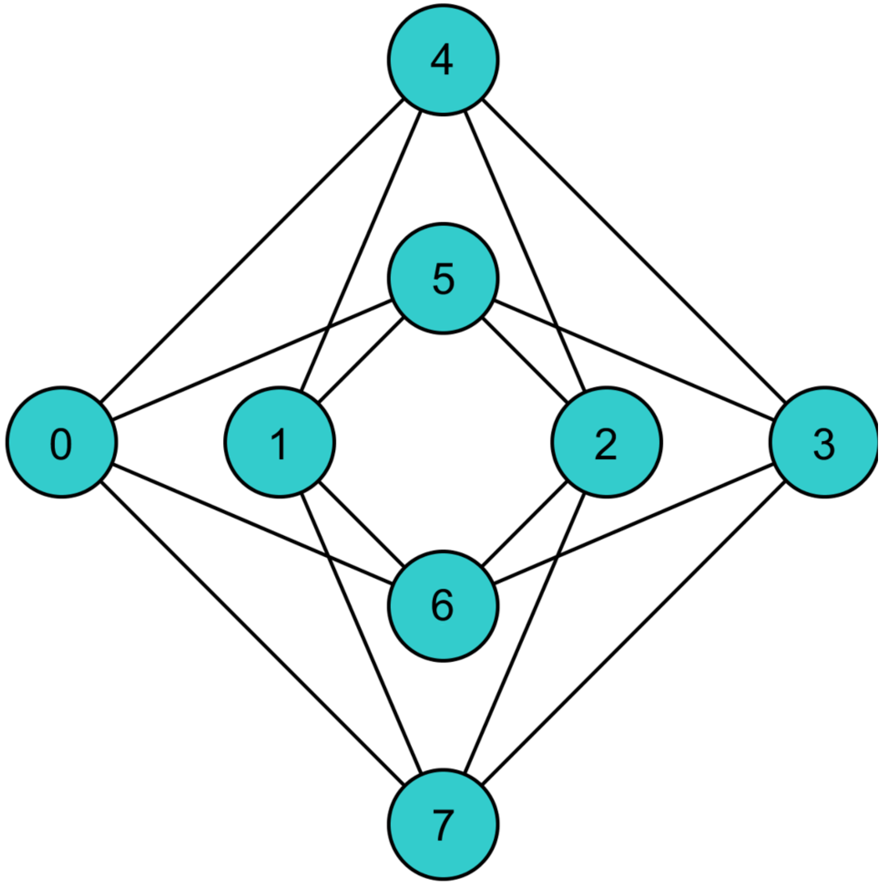

- Add High Order Terms to QUBO problem with pyqubo

# Graphs



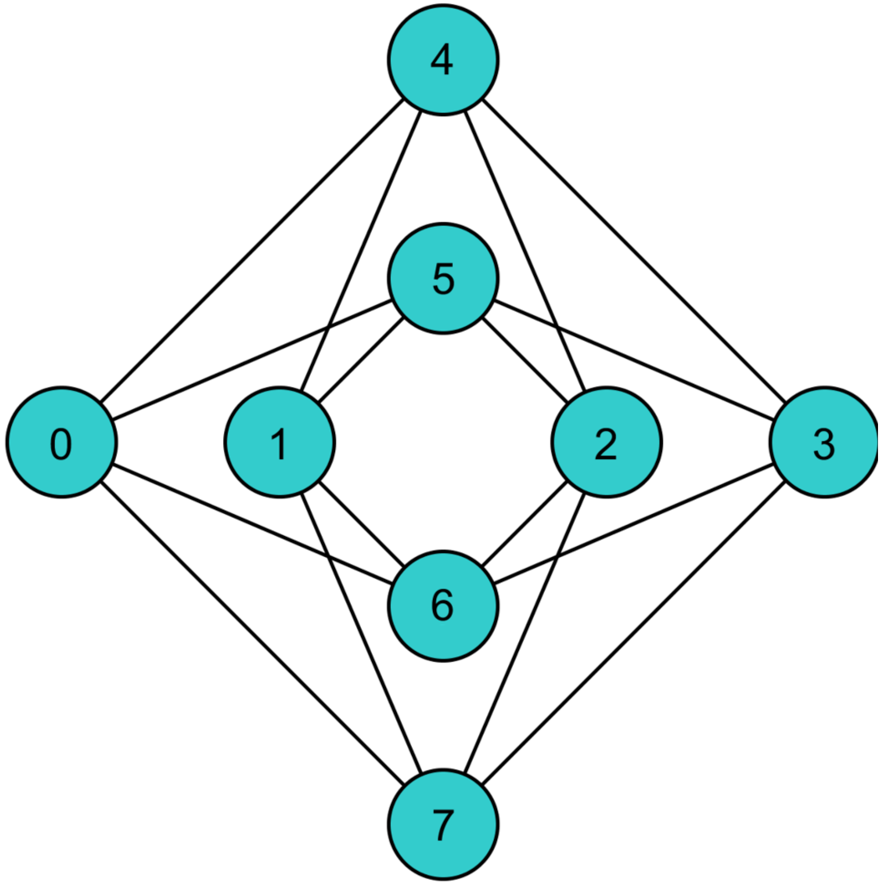

- Mathematically speaking, an undirected graph is defined as a set of vertices $V = \{v_1, \dots, v_N\}$

# Graphs



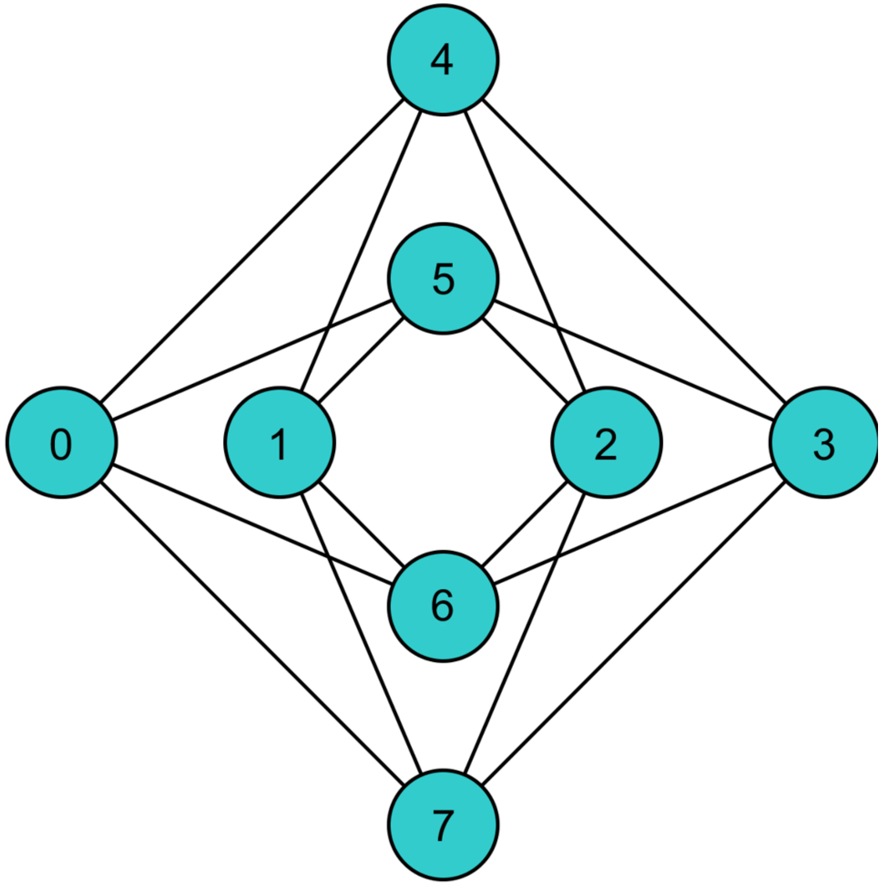

- Mathematically speaking, an undirected graph is defined as a set of vertices $V = \{v_1, \dots, v_N\}$
- and a set of edges $E \subseteq V \times V$

# Graphs



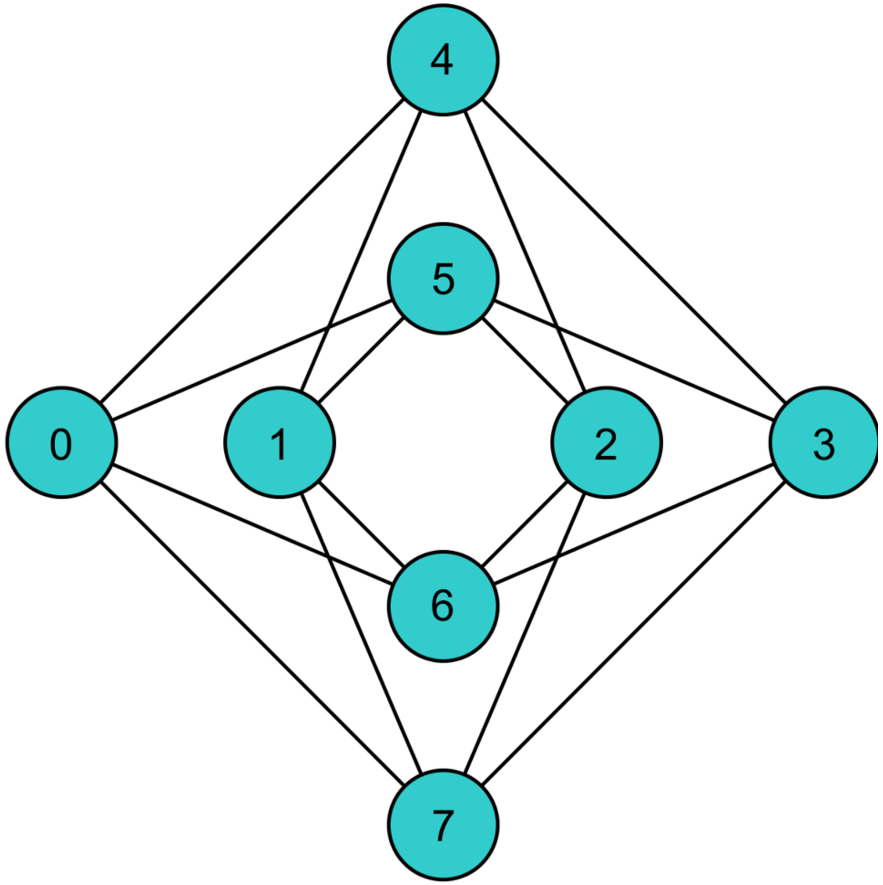

- Mathematically speaking, an undirected graph is defined as a set of vertices $V = \{v_1, \dots, v_N\}$
- and a set of edges $E \subseteq V \times V$
- Each node and each edge can be weighted with an arbitrary value (in this case we are talking about a weighted graph)

# 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

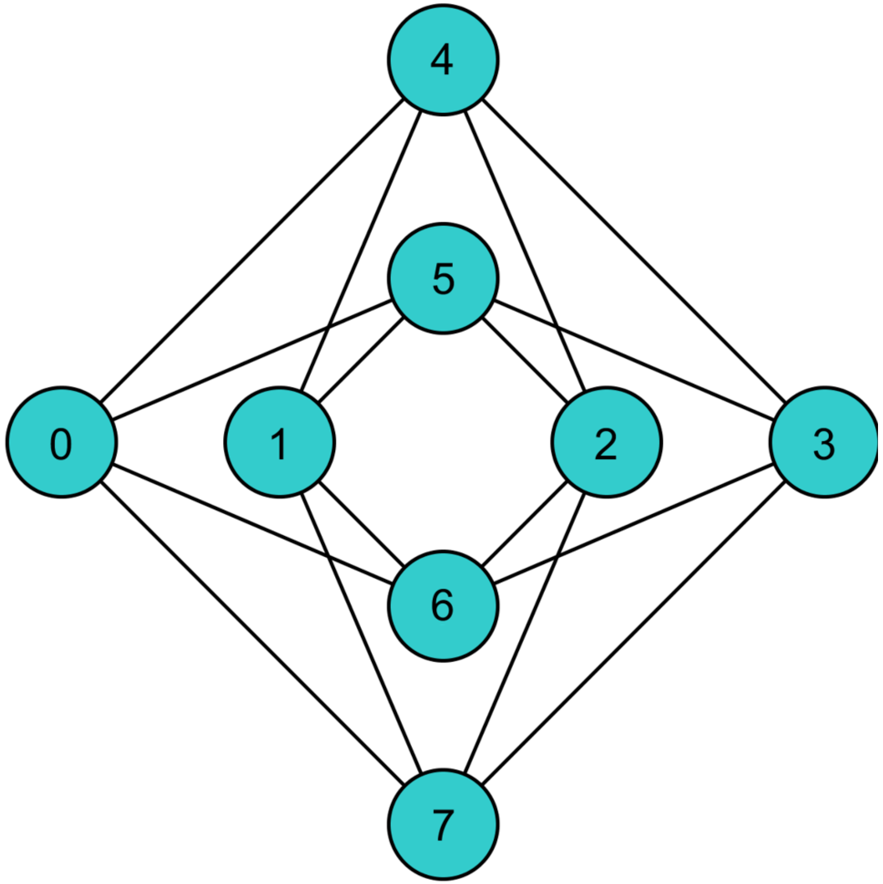
# Graphs




- 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

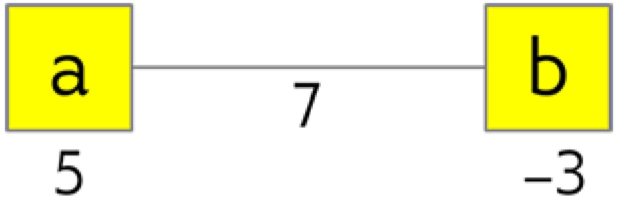
$$H(a,b) = 5a + 7ab - 3b$$

# Graphs

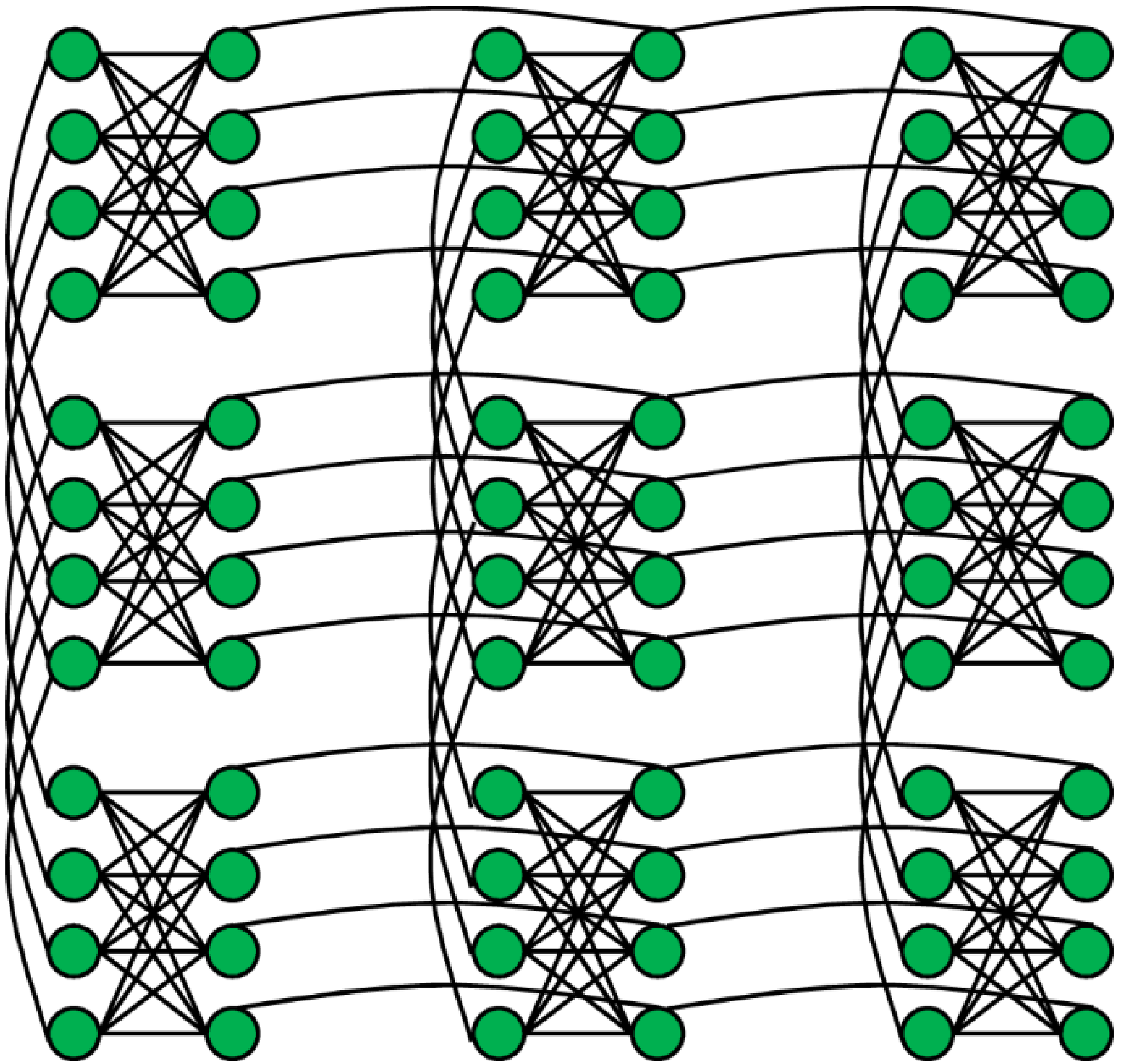



- Mathematically speaking, an undirected graph is defined as a set of vertices $V = \{v_1, \dots, v_N\}$
- and a set of edges $E \subseteq V \times V$
- Each node and each edge can be weighted with an arbitrary value (in this case we are talking about a weighted graph)
- In this way it is possible to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

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

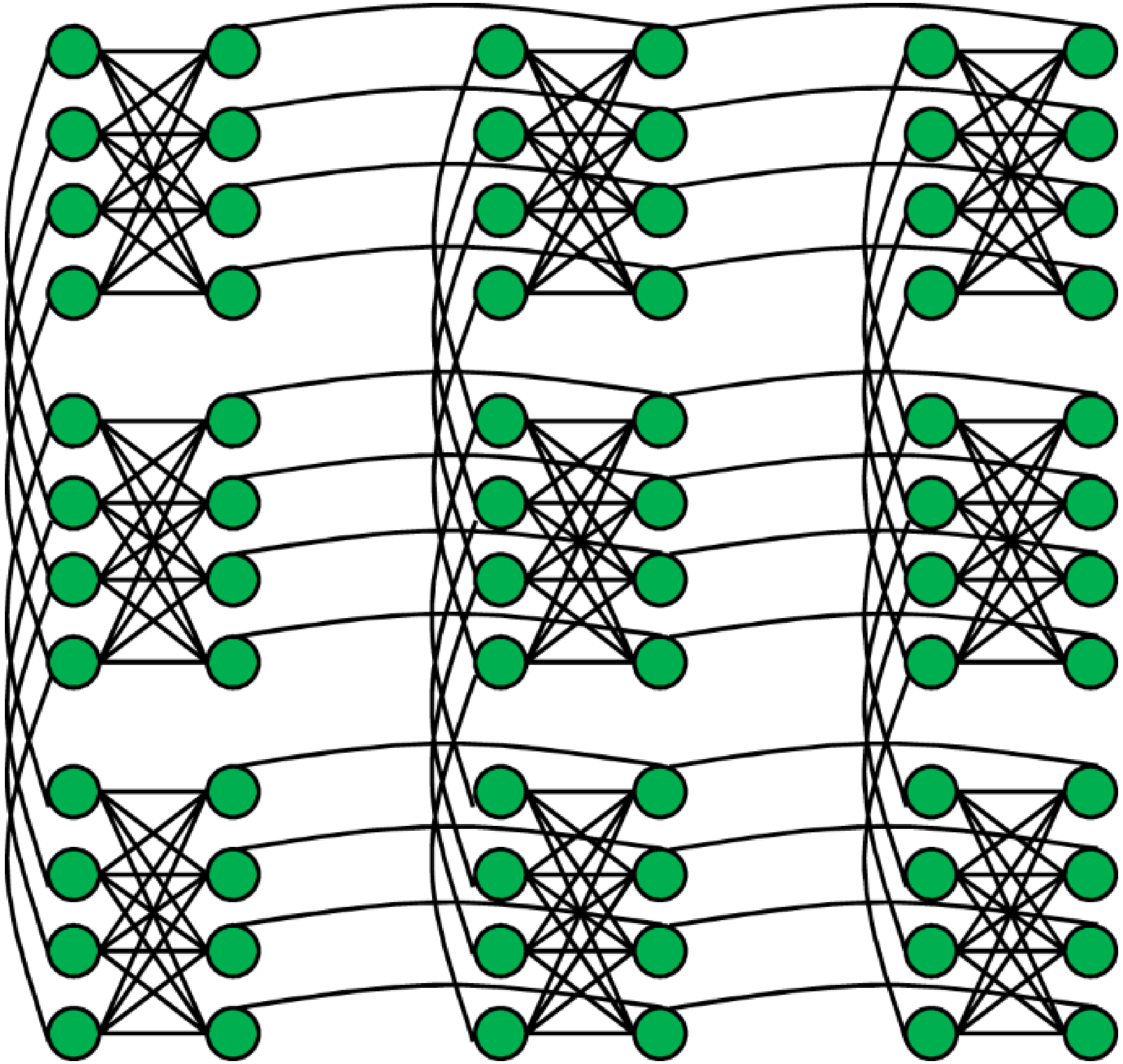


# Embedding a problem on a graph



- But what if the graph with which we want to represent the QUBO function does not have enough vertices or edges to do so?

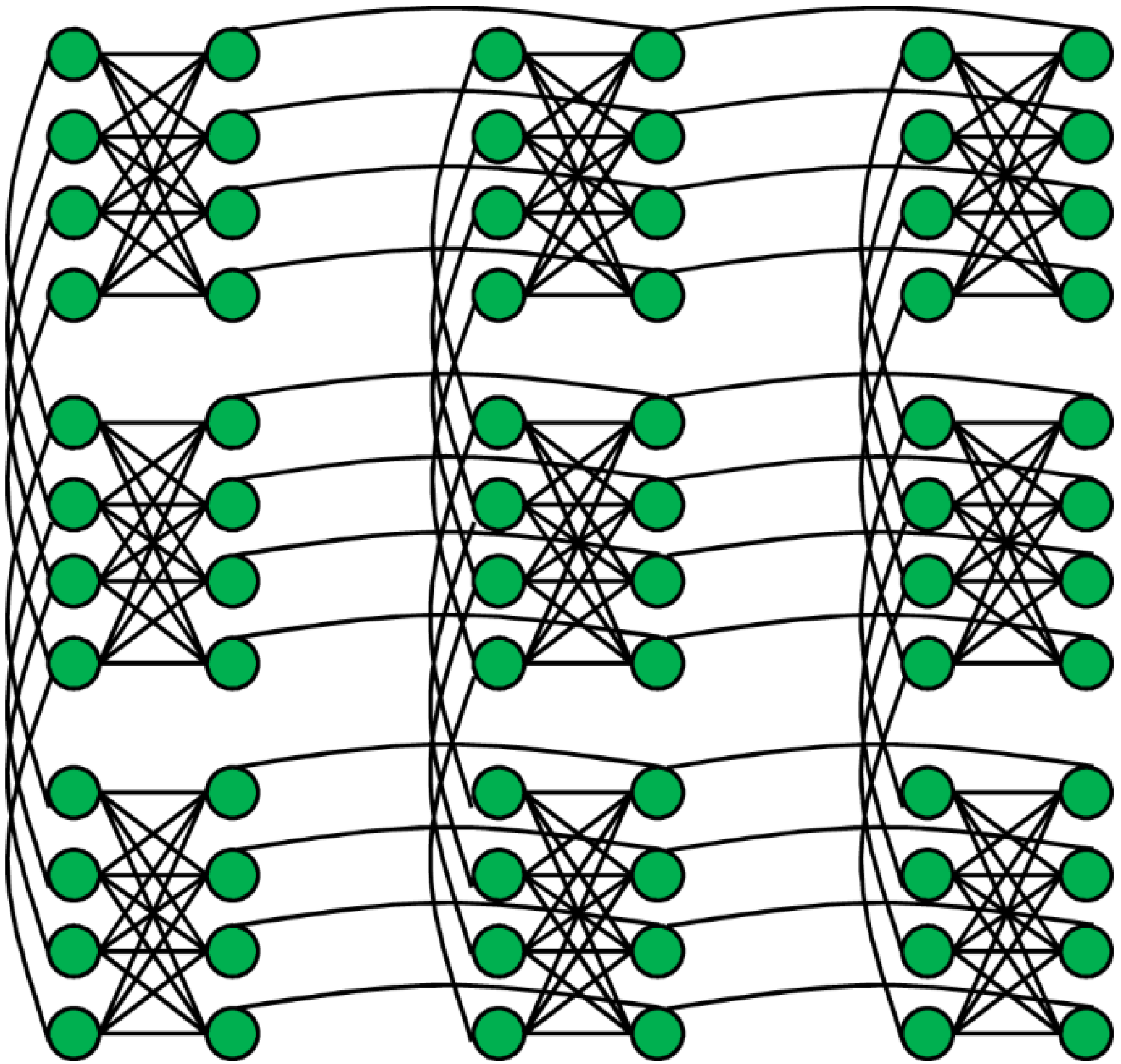
# Embedding a problem on a graph



- 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!

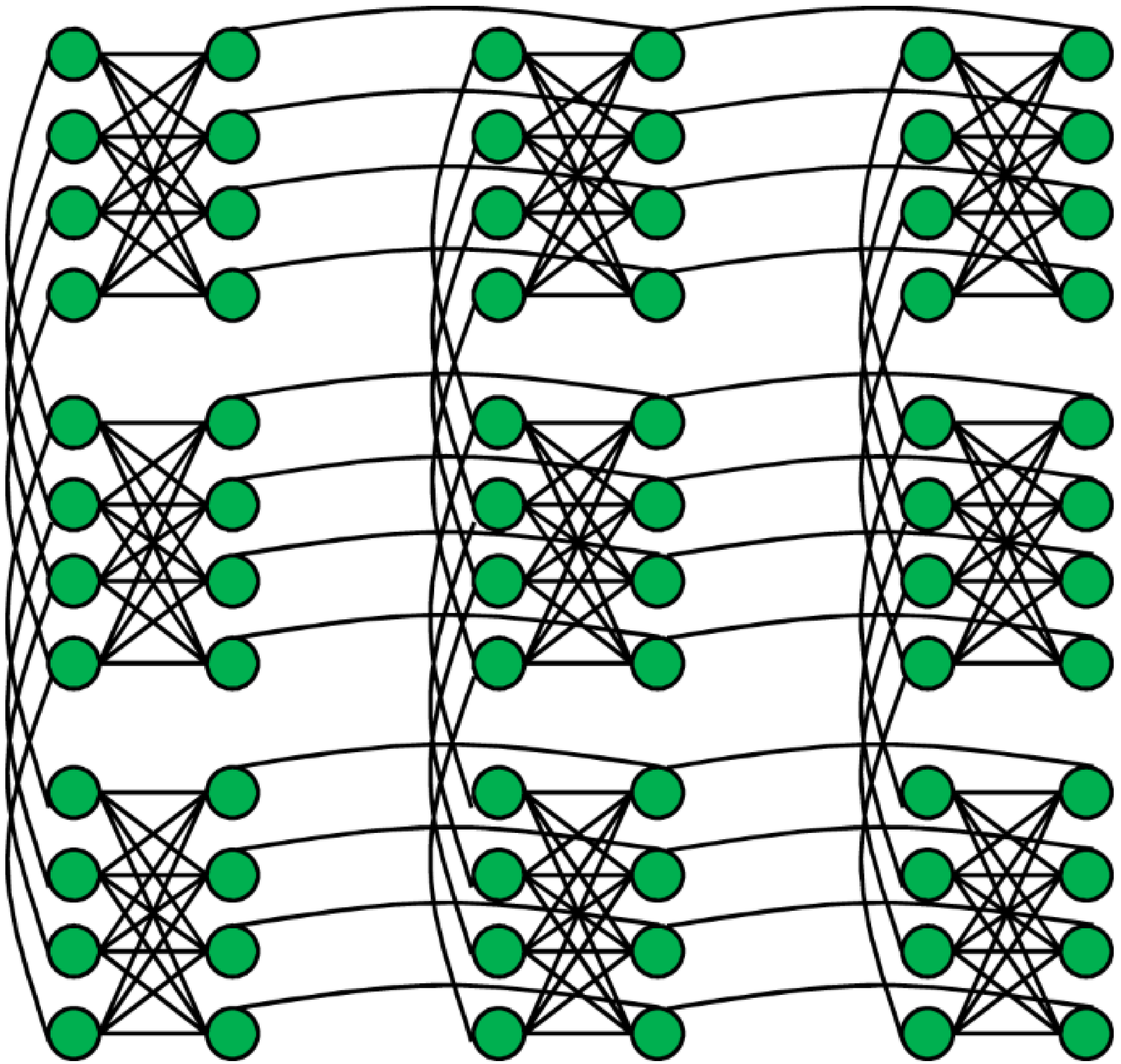
CINECA QUANTUM COMPUTING LAB

# Embedding a problem on a graph



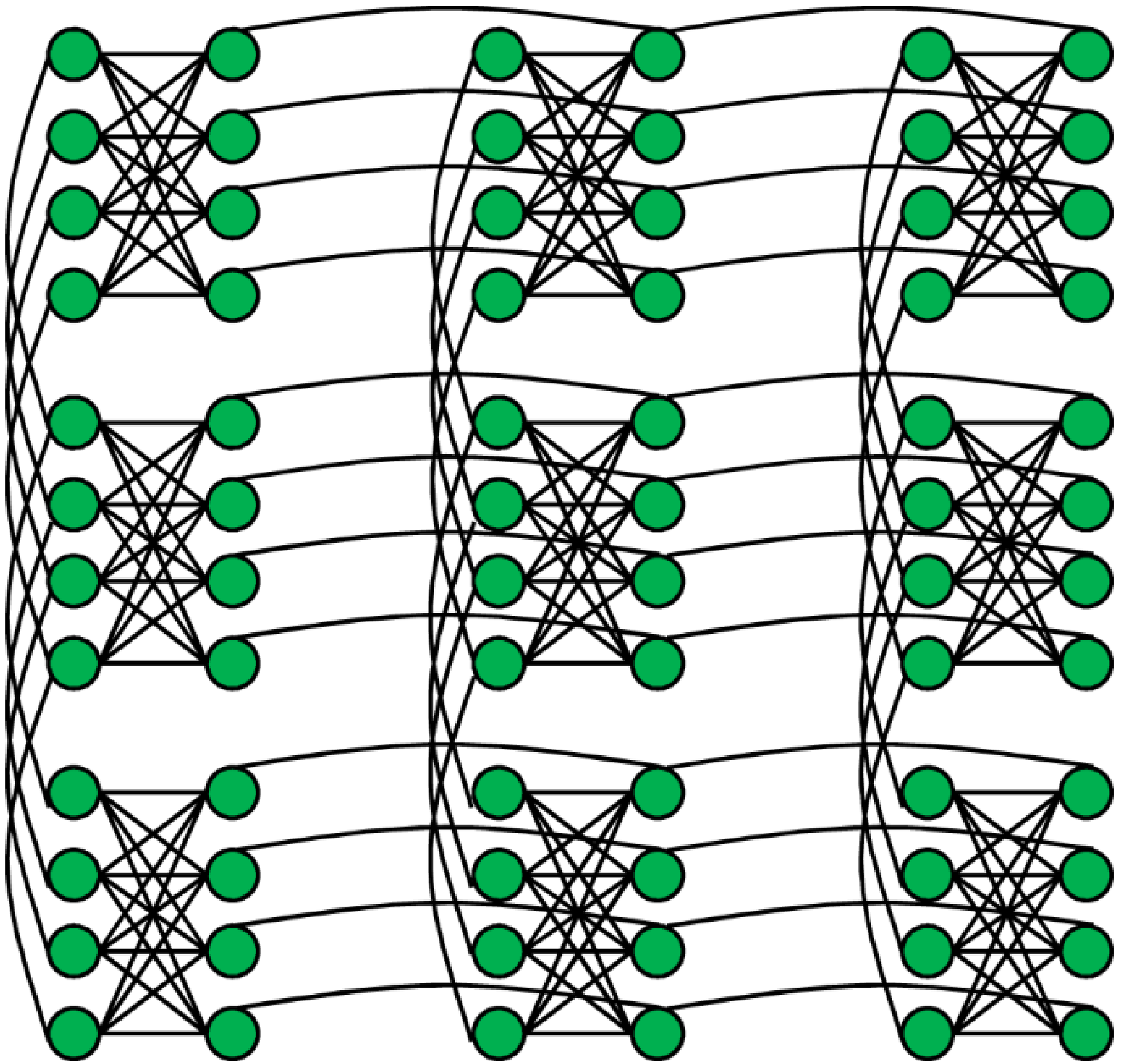
- But what if the graph with which we want to represent the QUBO function does not have enough vertices or edges to do so?
- In the case of the vertices, there is nothing to do: we have to change the problem and / or graph!
- In the case of the edges, however, something is possible to do

# Embedding a problem on a graph



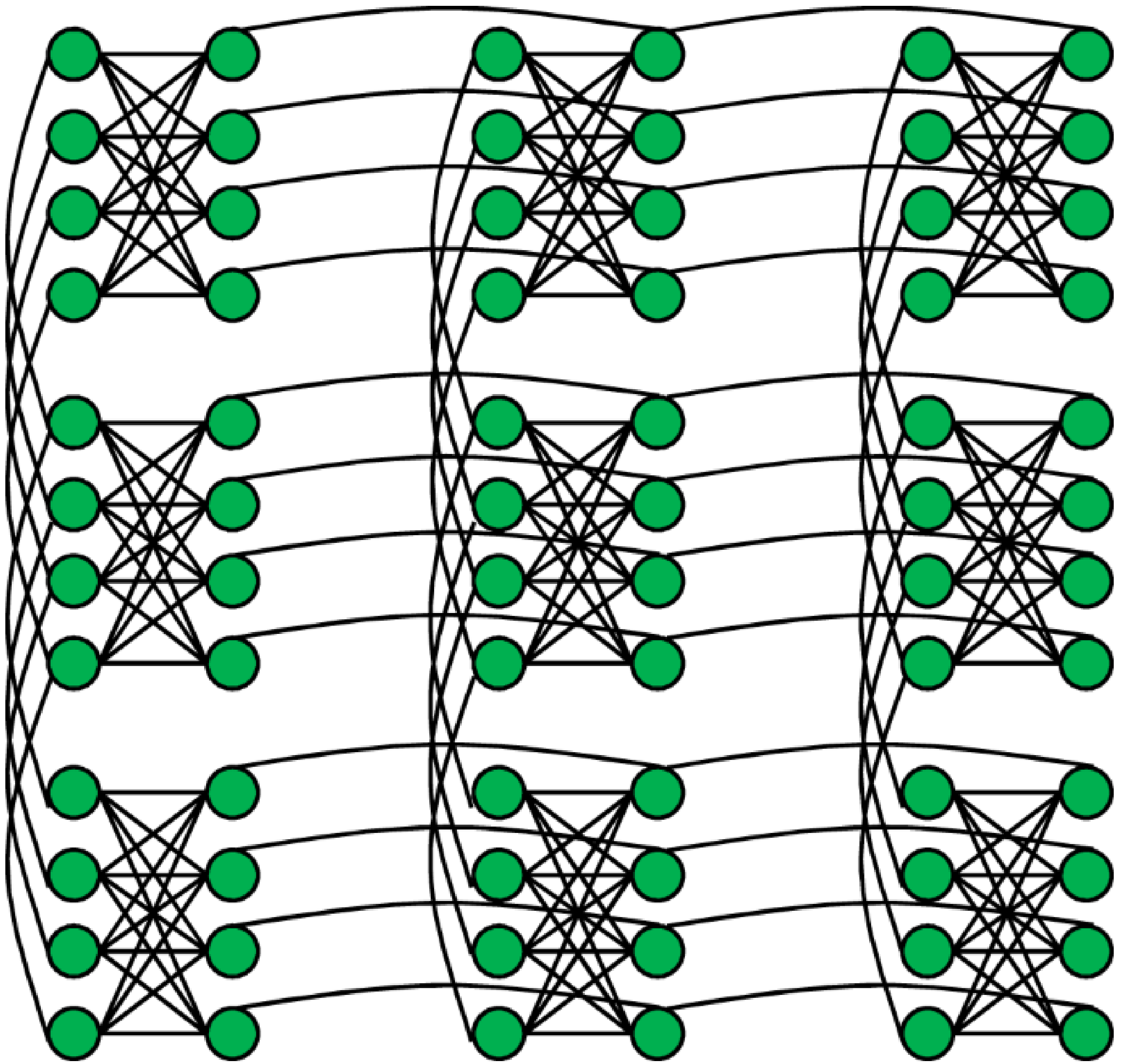
- But what if the graph with which we want to represent the QUBO function does not have enough vertices or edges to do so?
- In the case of the vertices, there is nothing to do: we have to change the problem and / or graph!
- In the case of the edges, however, something is possible to do
- The core of 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)

# 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

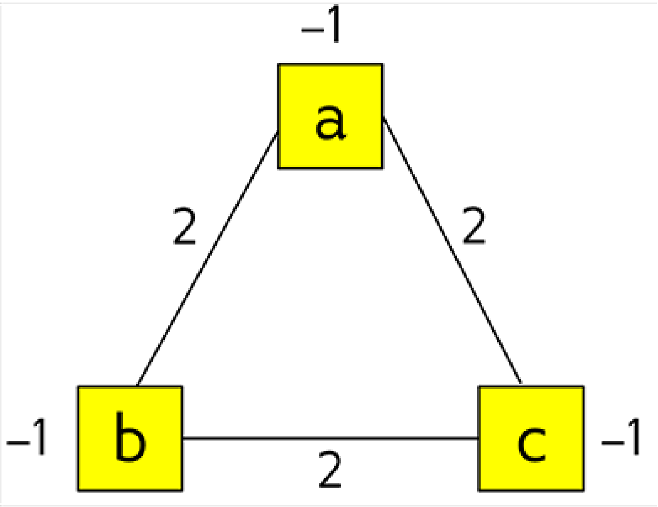
# Embedding a problem on a graph



- 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

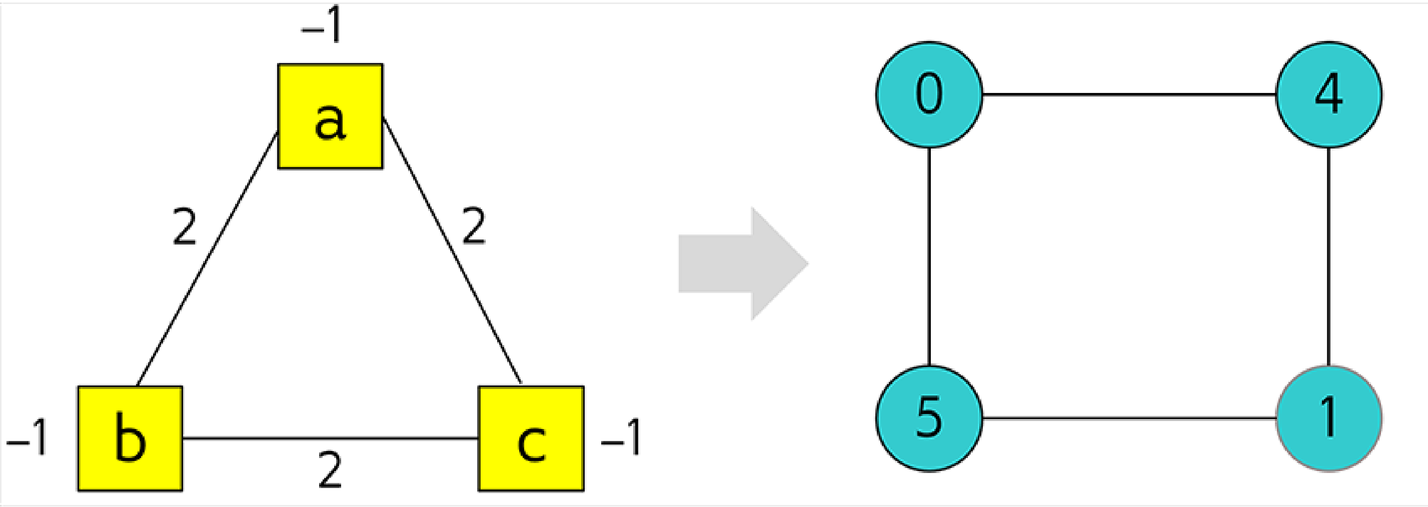
# Embedding a problem on a graph

- Suppose we have a QUBO problem that can be translated with the following graph



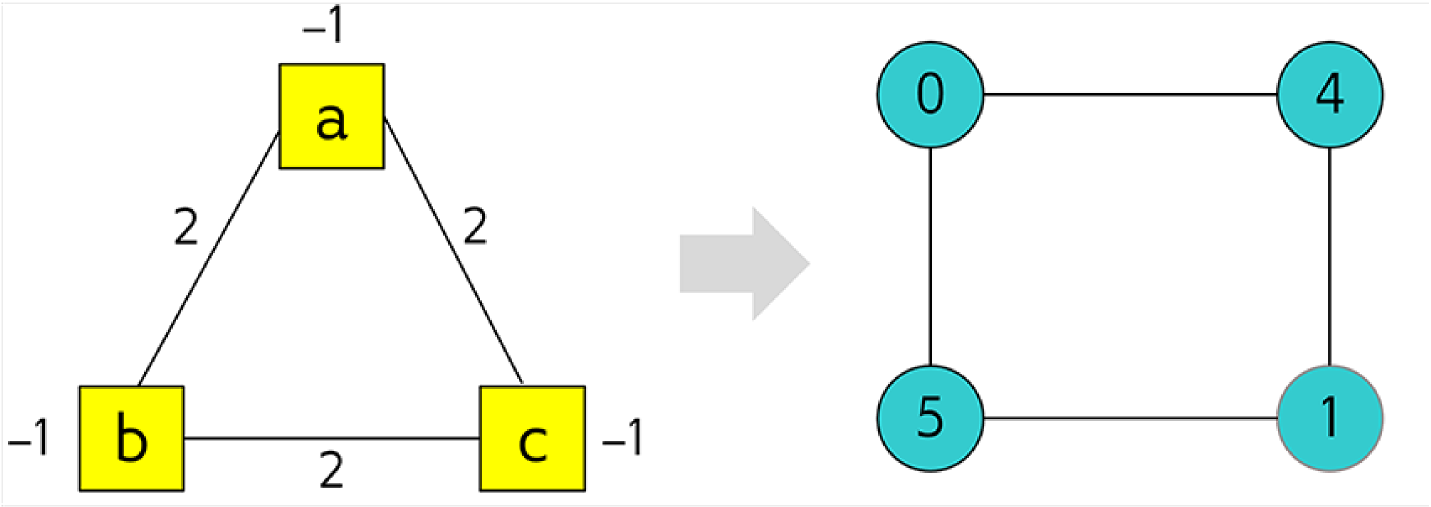
# Embedding a problem on a graph

- 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



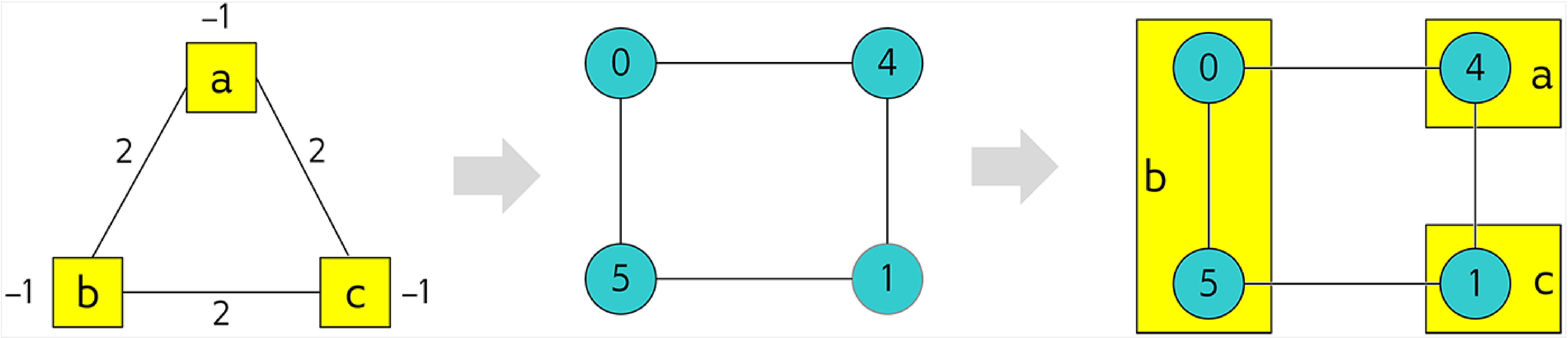
# Embedding a problem on a graph

- 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



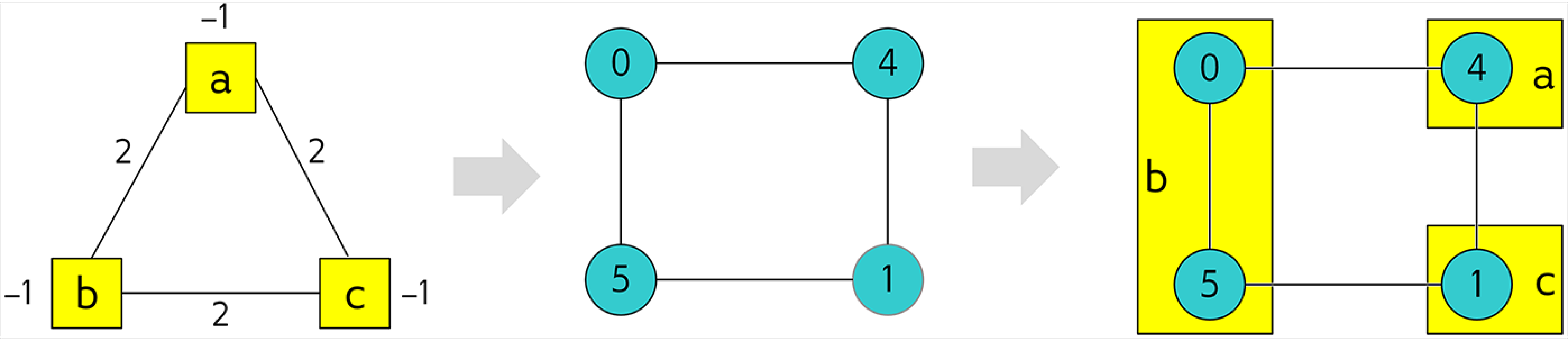
# Embedding a problem on a graph

- 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

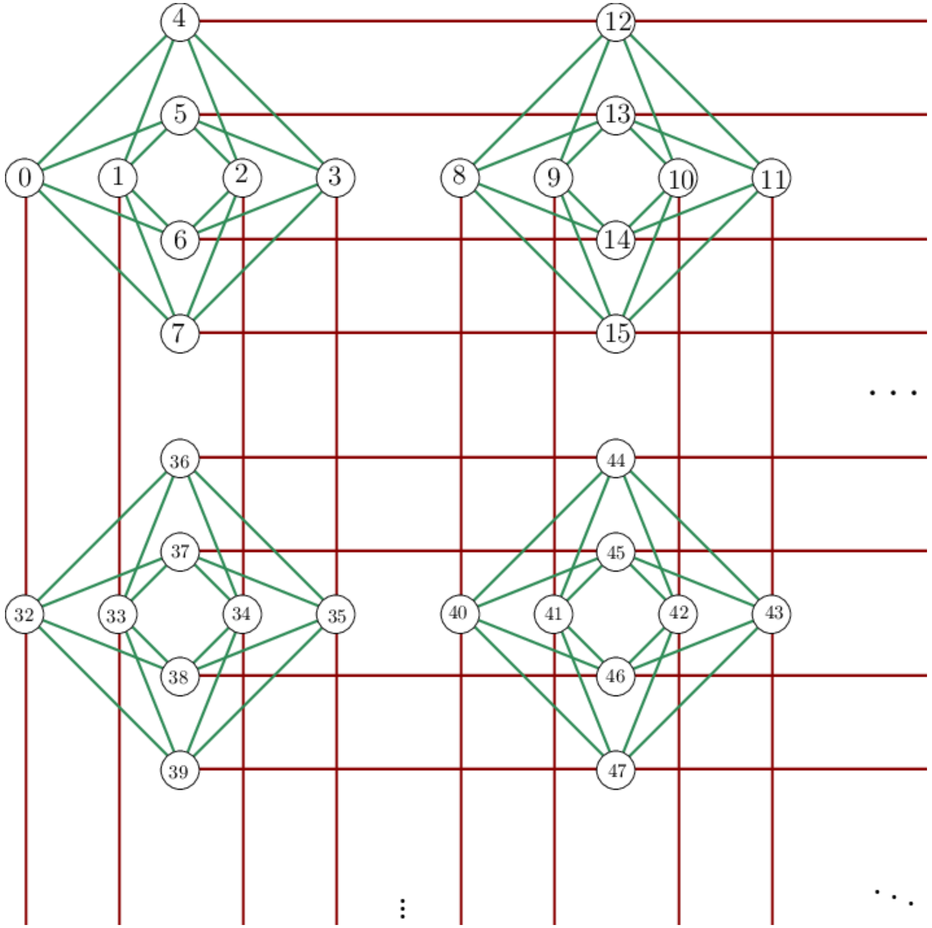


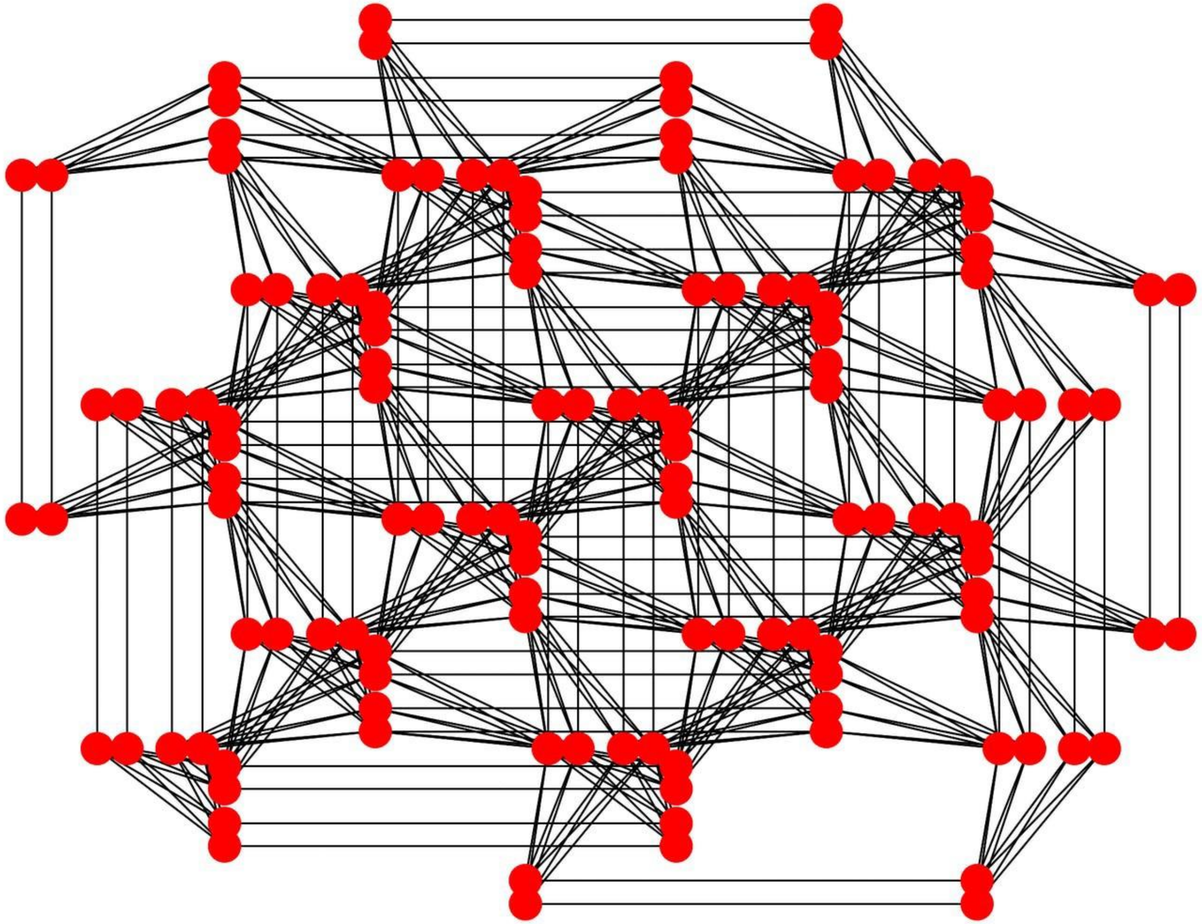
# Embedding a problem on a graph

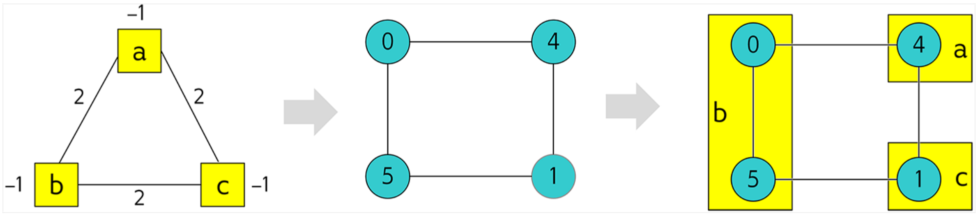
- Suppose we have a QUBO problem that can be translated with the following graph
- Suppose we also have a quantum annealer with a graph of this shape
- By looking at them, it seems impossible to map our problem to the target graph
- The embedding procedure allows for this mapping by forcing multiple 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

# Exercise 2: Antenna Placement

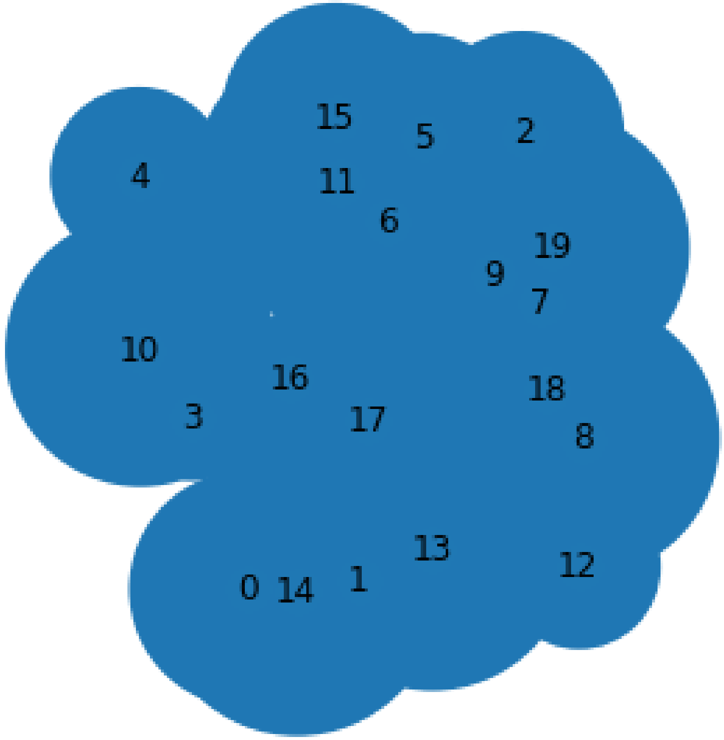
- Implement constraint into the Antenna Placement Problem

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

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

$$\min\left(\left(\sum_{i=0}^{N} \beta(1 - 2F) q_i + \sum_{i<j} 2\beta q_i q_j\right)\right) - \sum_{i=1}^{N} \gamma L_i q_i \qquad \sum_{i=1}^{N+F} q_i = F$$

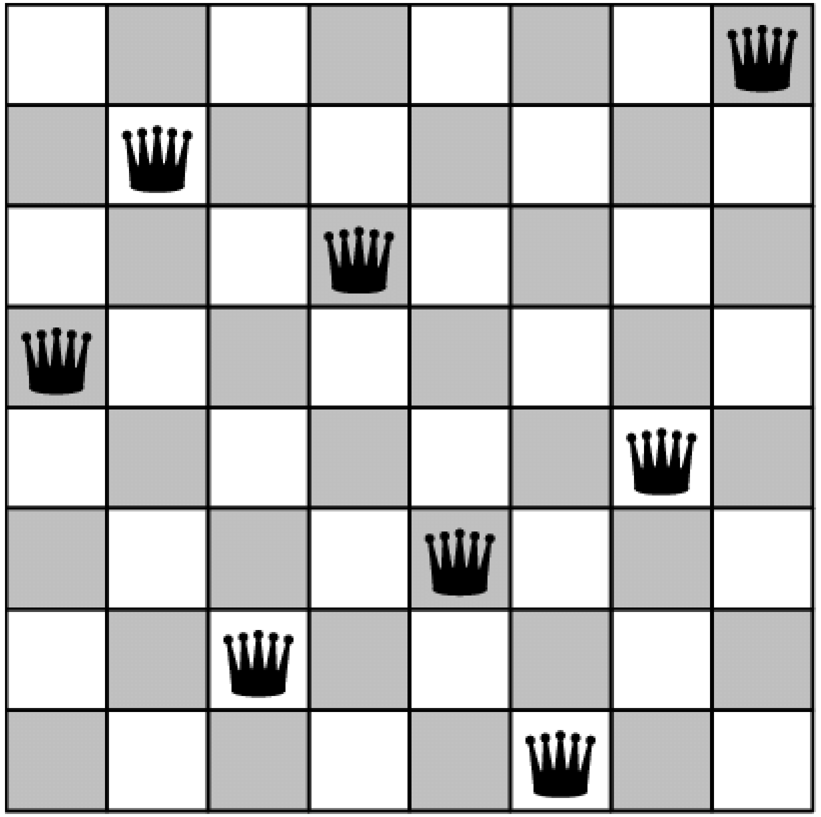
$$\sum_{i=1}^{2N} C_i A_i q_i = A_m$$




- Try the embedding on Pegasus and Chimera

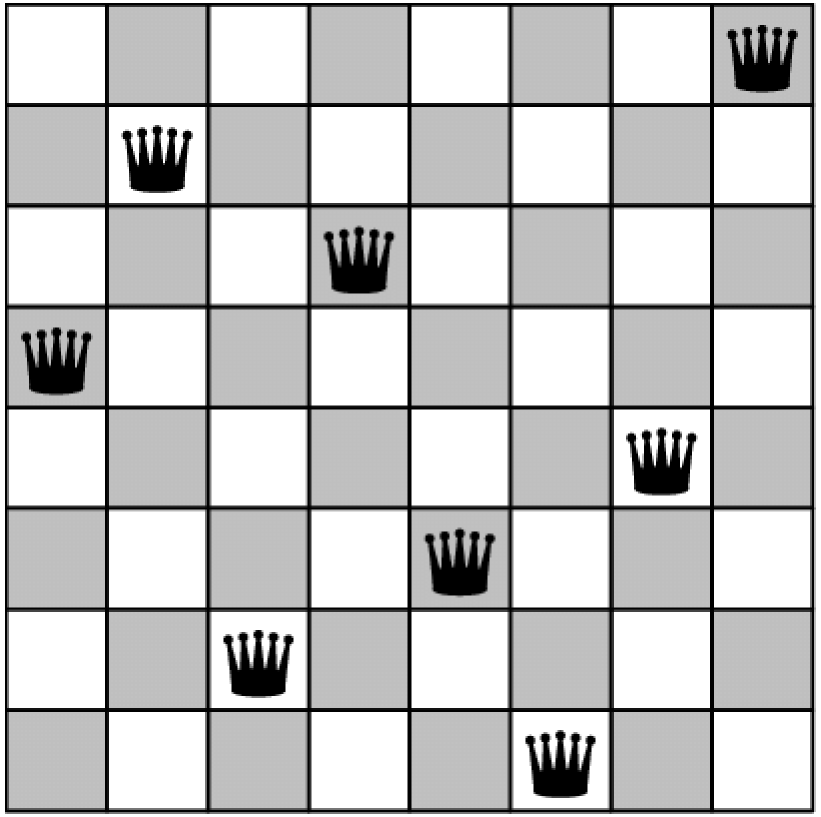
# Exercise 3: N-Queens Puzzle

- Let us now turn to another problem: **the N-queens puzzle**



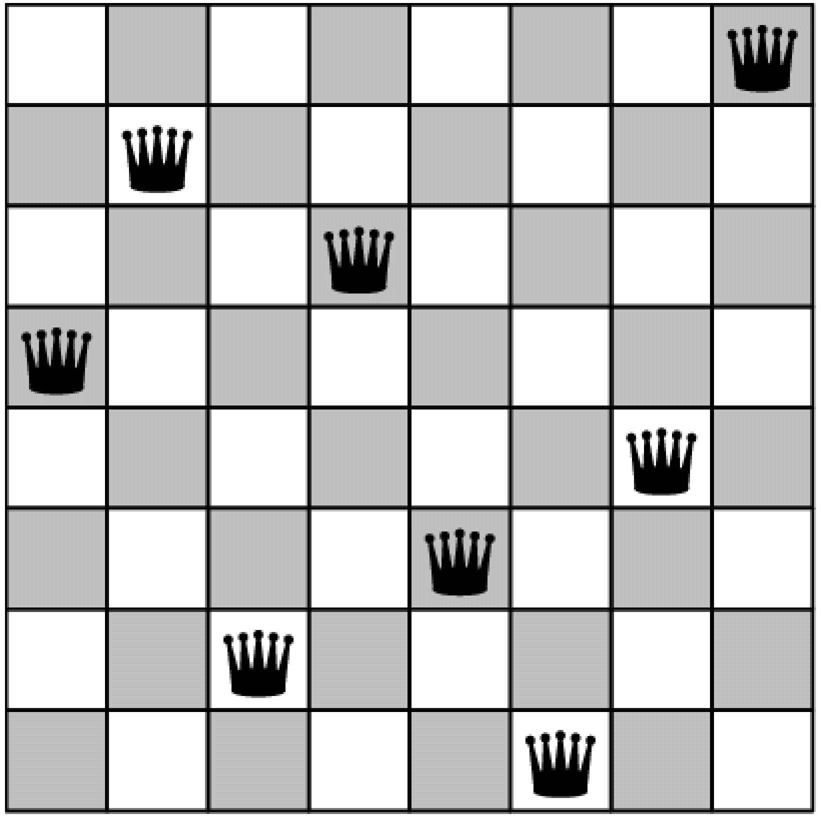
# Exercise 3: N-Queens Puzzle

- Let us now turn to another problem: **the N-queens puzzle**
- The N-queens puzzle is a generalization of the better known **8-queens puzzle**



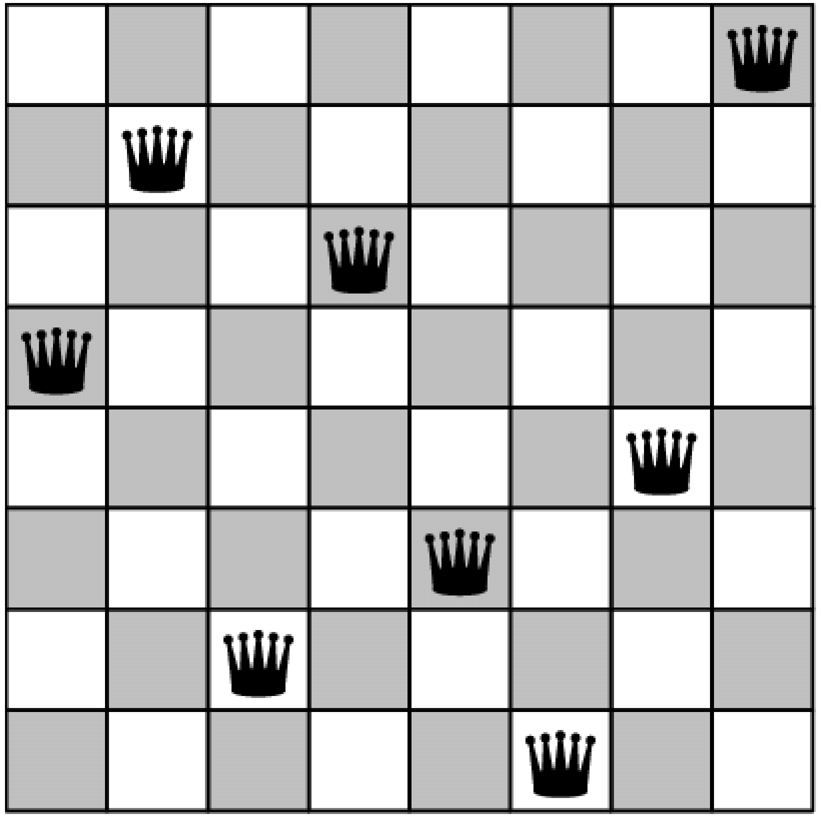
# Exercise 3: N-Queens Puzzle

- Let us now turn to another problem: **the N-queens puzzle**
- The N-queens puzzle is a generalization of the better known **8-queens puzzle**
- The 8-queens puzzle can be described in the following way: let's consider a chessboard. Find a way to arrange 8 queens on the chessboard so that **none of them are in check** by any of the other queens



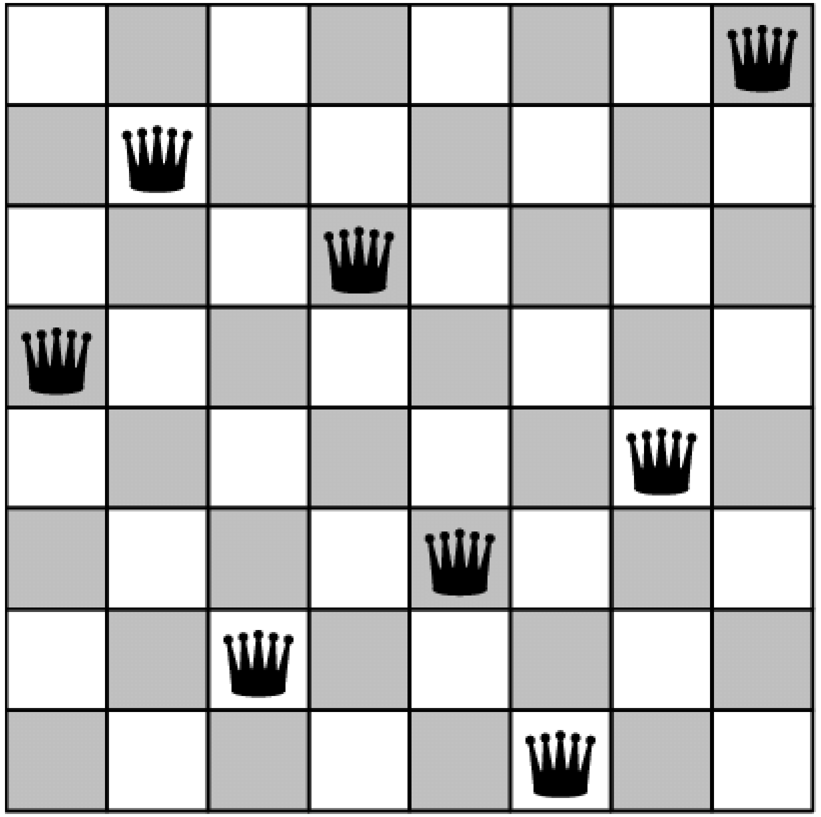
# Exercise 3: N-Queens Puzzle

- Let us now turn to another problem: **the N-queens puzzle**
- The N-queens puzzle is a generalization of the better known **8-queens puzzle**
- The 8-queens puzzle can be described in the following way: let's consider a chessboard. Find a way to arrange 8 queens on the chessboard so that **none of them are in check** by any of the other queens
- The game is generalized as follows: let's consider a chessboard **of dimension NxN**. Find a way **to arrange N queens** on the chessboard so that none of them are in check by any of the other queens



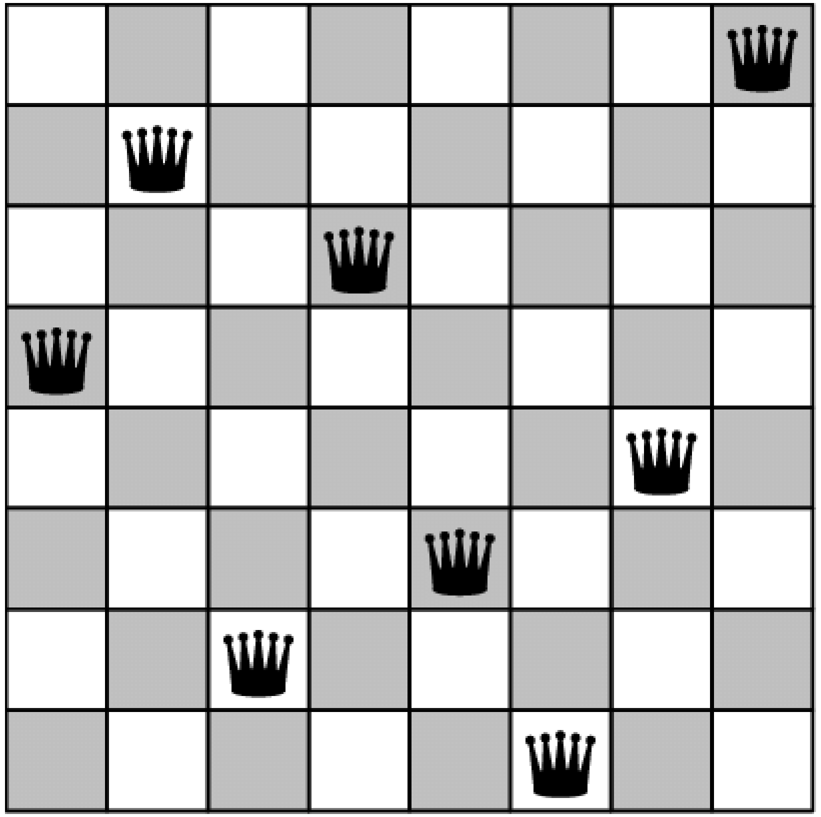
# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem



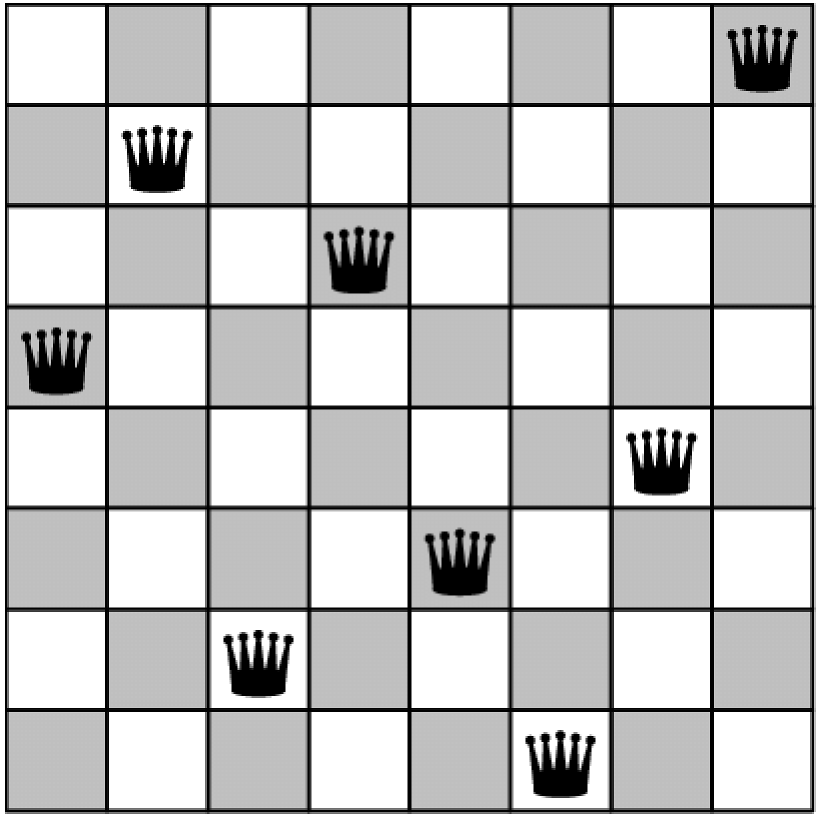
# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.



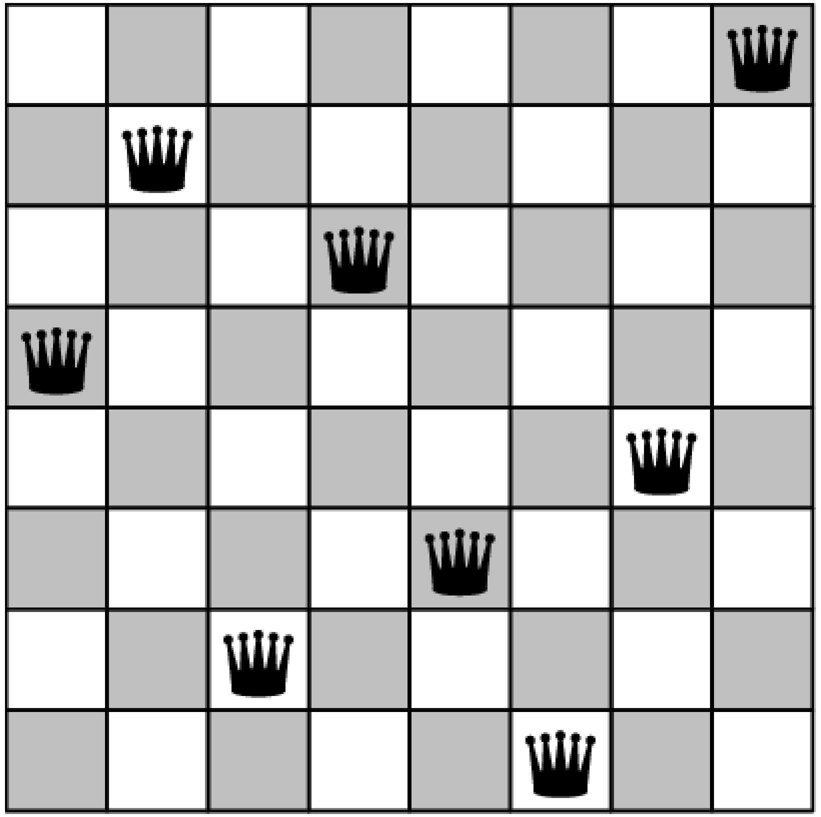
# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.
- Each binary variable is therefore associated with a square on the chessboard



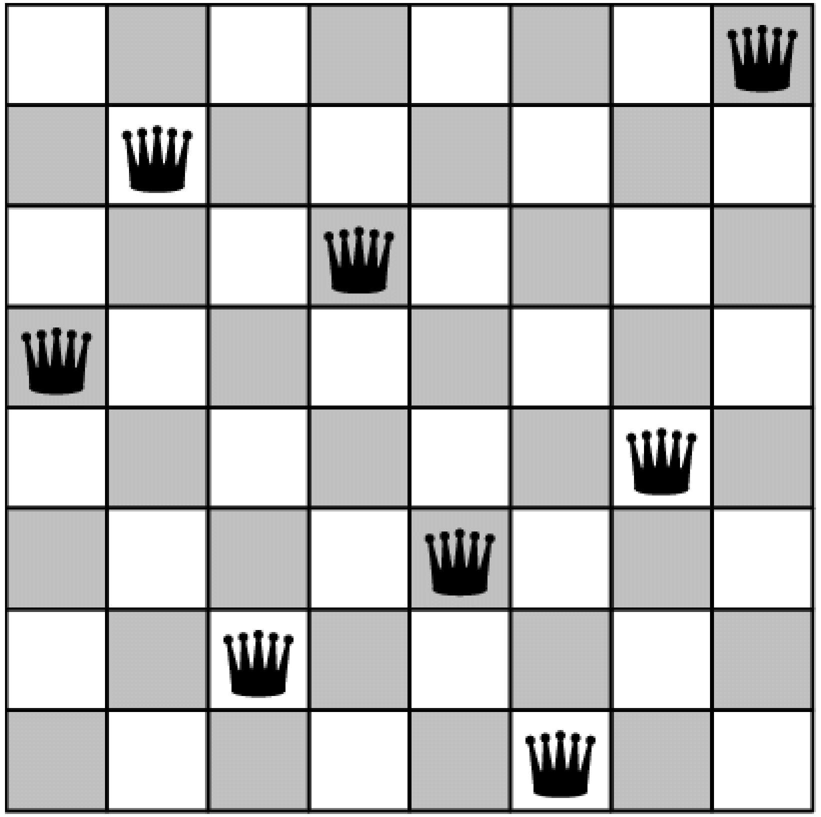
# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.
- Each binary variable is therefore associated with a square on the chessboard
- Each binary variable **will be 1 if its square contains a queen** or 0 otherwise



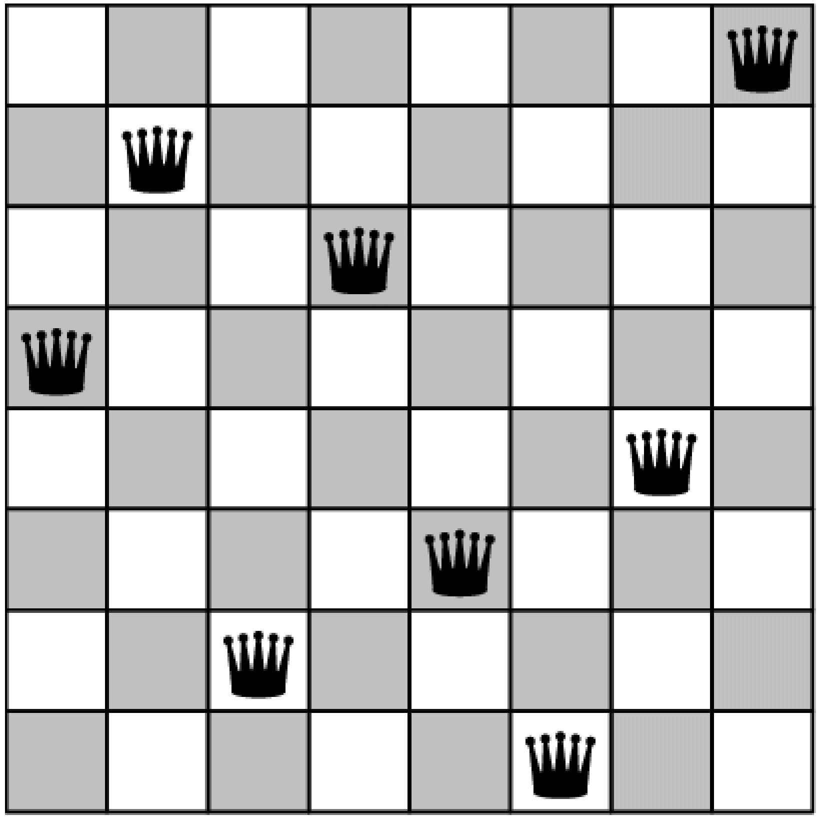
# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.
- Each binary variable is therefore associated with a square on the chessboard
- Each binary variable **will be 1 if its square contains a queen** or 0 otherwise
- Put in these terms, the first requirement that my QUBO problem has to satisfy is: *I want to have* ***exactly*** *N queens on the board*



# Exercise 3: N-Queens Puzzle

- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.
- Each binary variable is therefore associated with a square on the chessboard
- Each binary variable **will be 1 if its square contains a queen** or 0 otherwise
- Put in these terms, the first requirement that my QUBO problem has to satisfy is: *I want to have **exactly** N queens on the board*

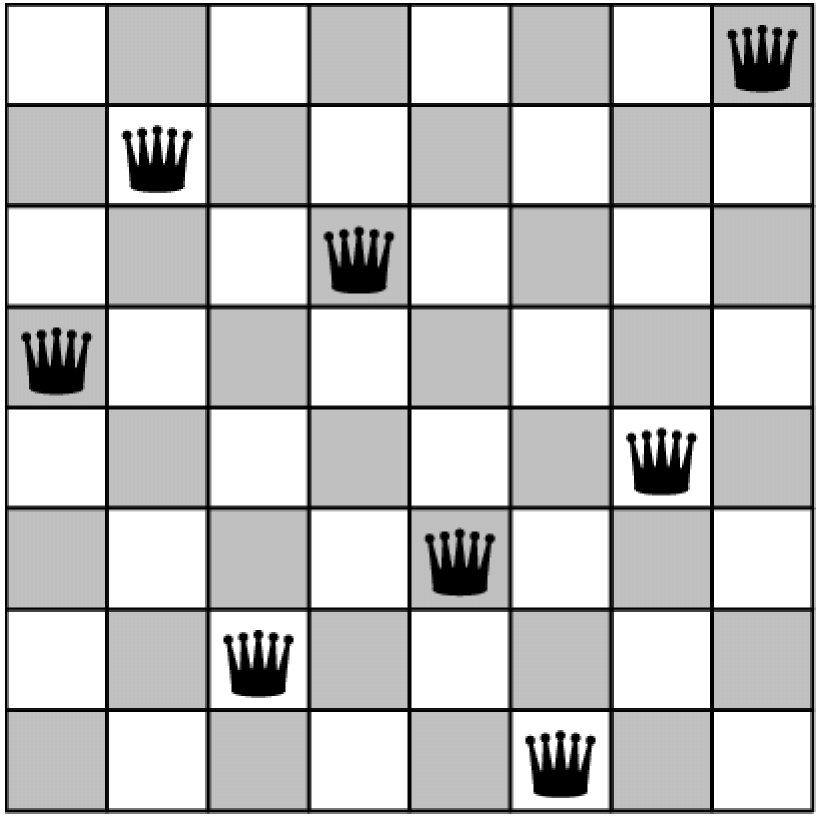


$$\sum_{i=1}^{N} q_i = N$$

# Exercise 3: N-Queens Puzzle

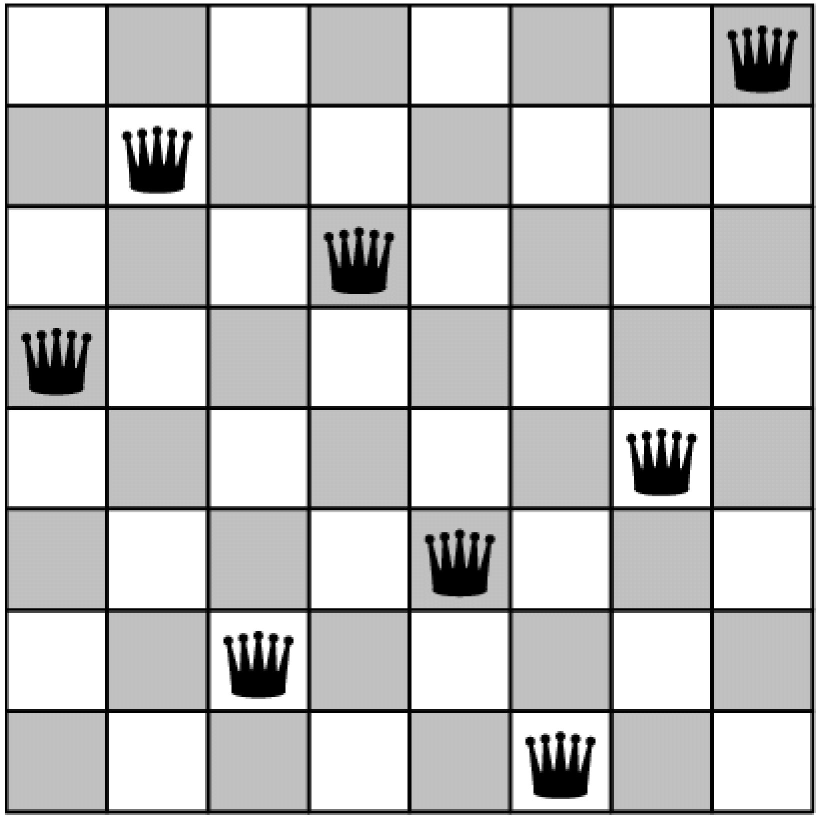
- Let's think about how to turn the problem into a QUBO problem
- We therefore consider a **vector of binary variables**. We will take **one for each square** of the board we are considering.
- Each binary variable is therefore associated with a square on the chessboard
- Each binary variable **will be 1 if its square contains a queen** or 0 otherwise
- Put in these terms, the first requirement that my QUBO problem has to satisfy is: *I want to have **exactly** N queens on the board*

$$\sum_{i=1}^{N} q_i = N \Rightarrow \min \left( \sum_{i=1}^{N} q_i - N \right)^2$$



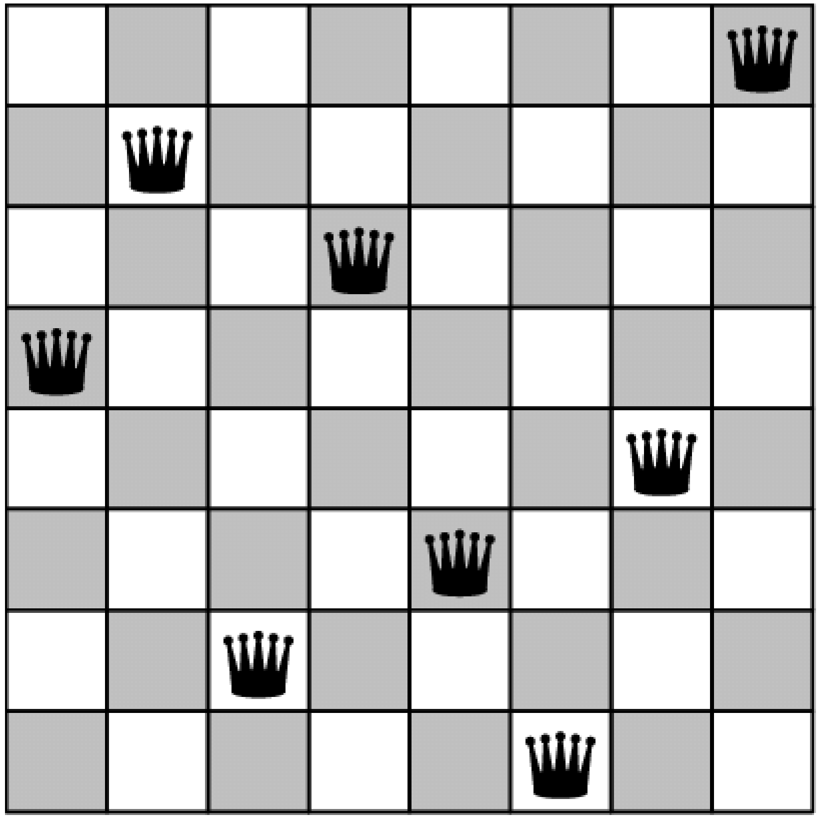
# Exercise 3: N-Queens Puzzle

$$\left(\sum_{i=1}^{N} q_i - N\right)^2$$



# Exercise 3: N-Queens Puzzle

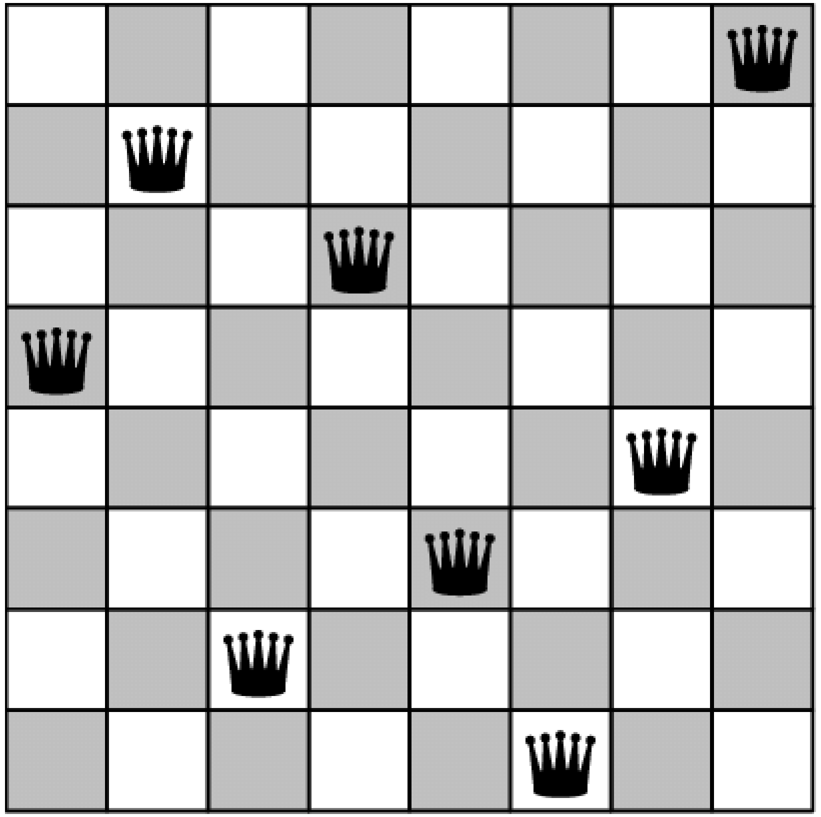
$$\left(\sum_{i=1}^{N} q_i - N\right)^2 = \left(\sum_{i=1}^{N} q_i\right)^2 - 2\sum_{i=1}^{N} q_i$$



# Exercise 3: N-Queens Puzzle

$$\left(\sum_{i=1}^{N} q_i - N\right)^2 = \left(\sum_{i=1}^{N} q_i\right)^2 - 2\sum_{i=1}^{N} q_i$$

$$= \sum_{i=1}^{N} q_i + 2\sum_{i<j} q_i q_j - 2\sum_{i=1}^{N} q_i$$

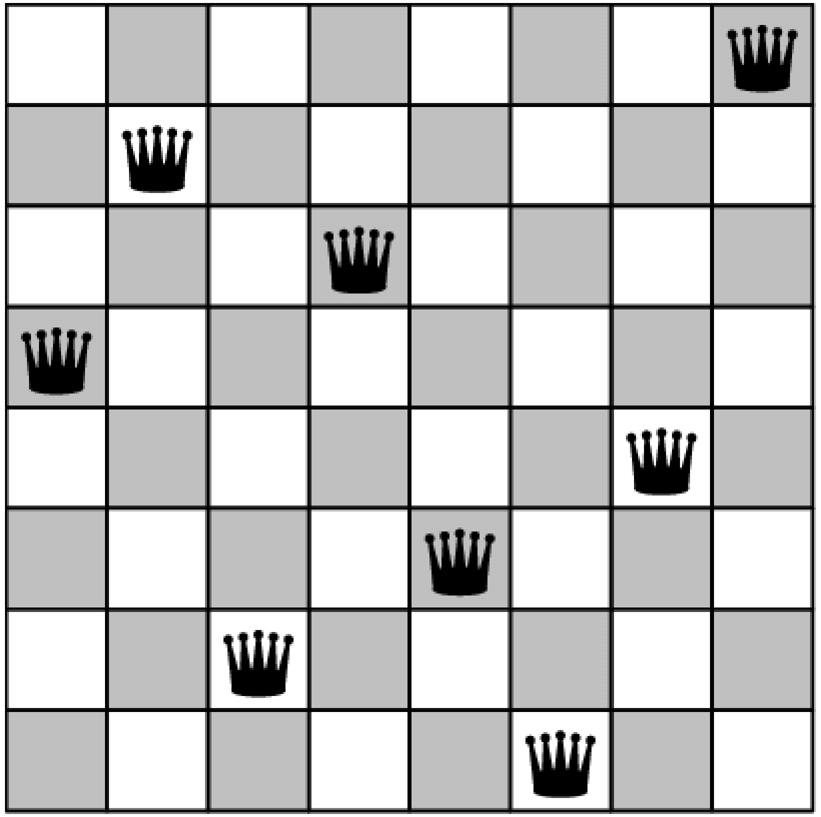


# Exercise 3: N-Queens Puzzle

$$\left(\sum_{i=1}^{N} q_i - N\right)^2 = \left(\sum_{i=1}^{N} q_i\right)^2 - 2\sum_{i=1}^{N} q_i$$

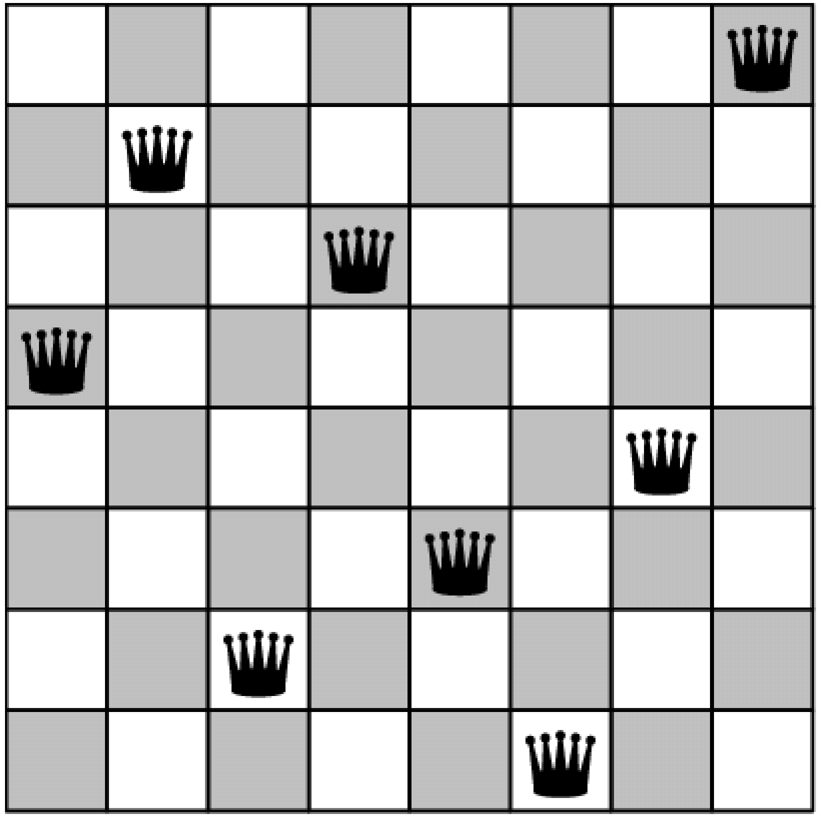
$$= \sum_{i=1}^{N} q_i + 2\sum_{i<j} q_i q_j - 2\sum_{i=1}^{N} q_i$$

$$= \sum_{i=1}^{N} (1 - 2N)\, q_i + \sum_{i<j} 2 q_i q_j$$



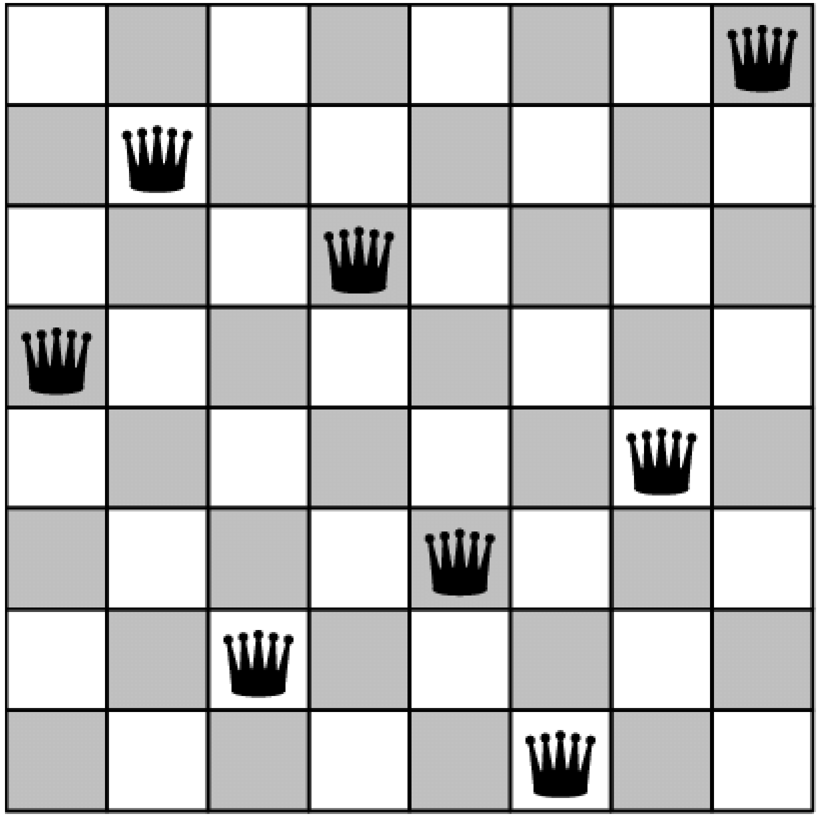
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem



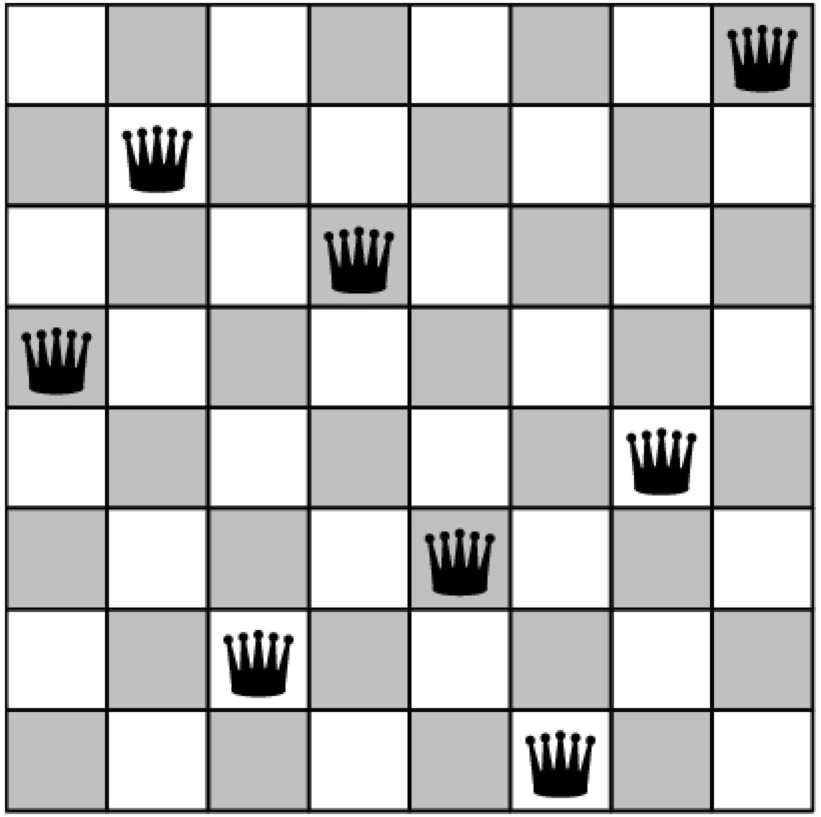
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints



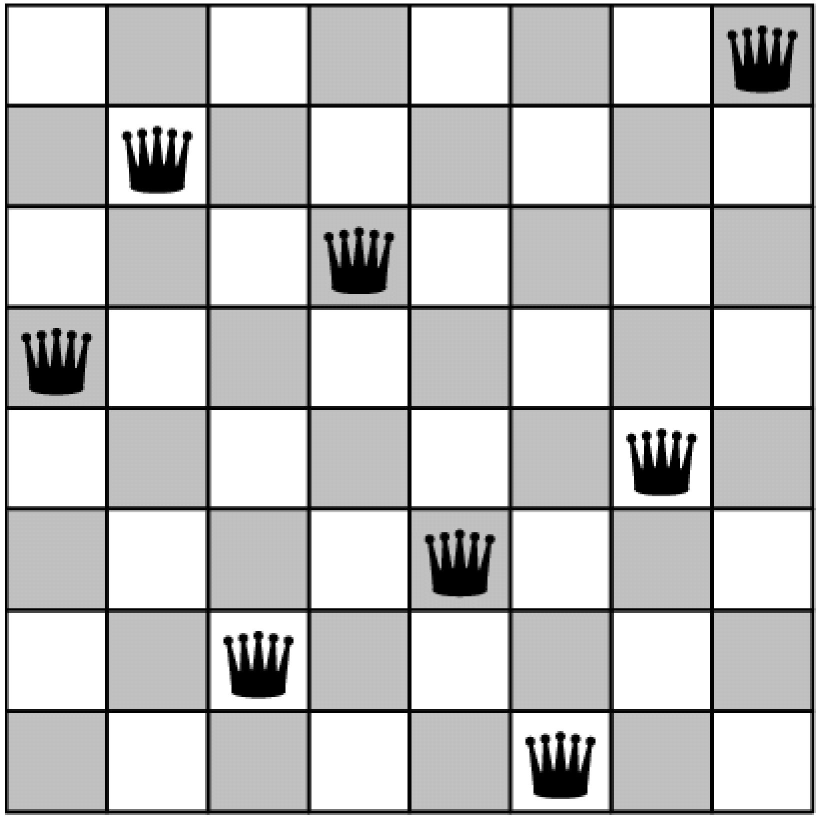
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?



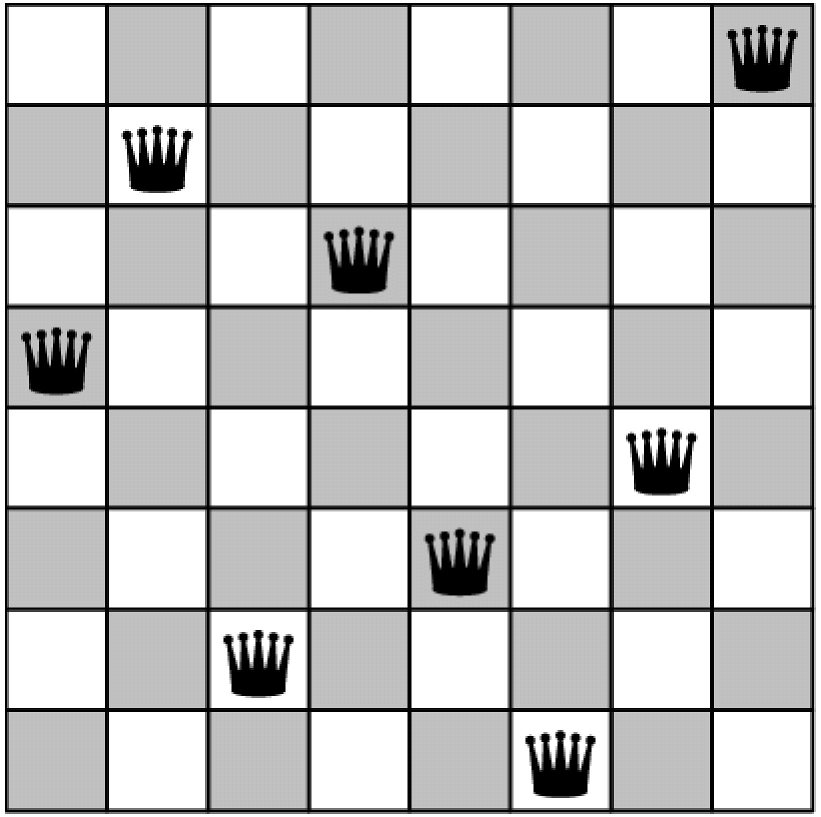
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?
- Exactly one queen for each row



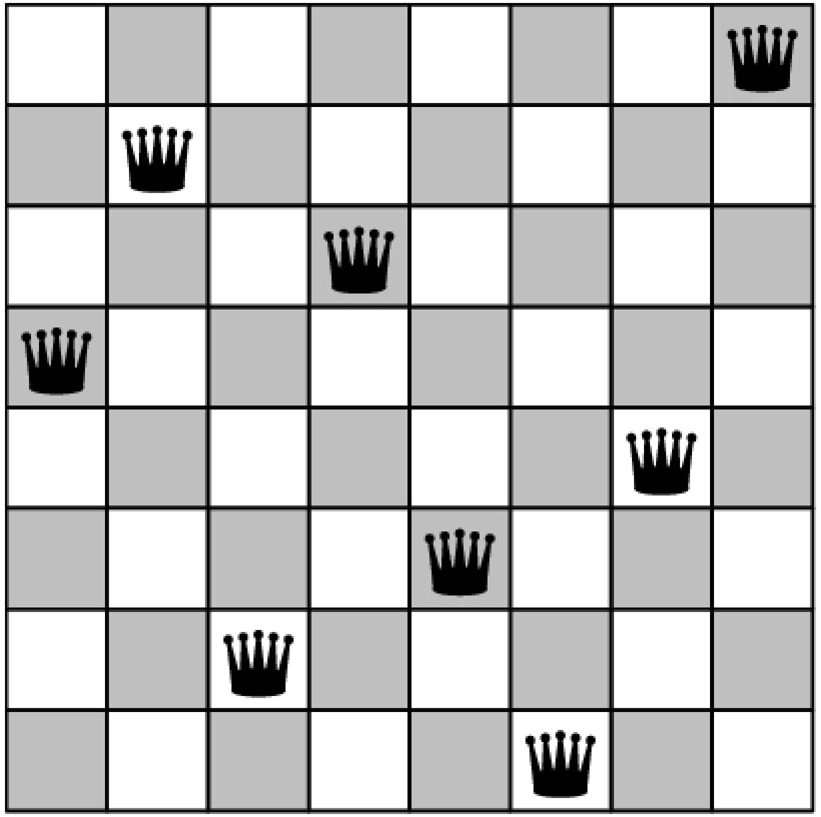
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?
- Exactly one queen for each row
- Exactly one queen for each column



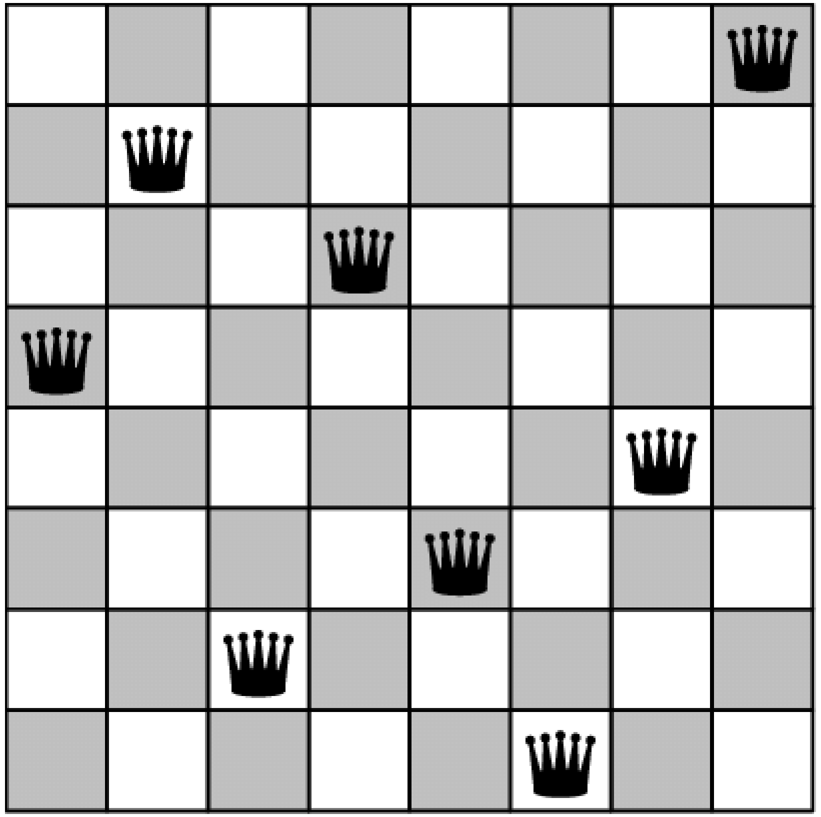
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?
- Exactly one queen for each row
- Exactly one queen for each column
- At most one queen for each diagonal (both directions)



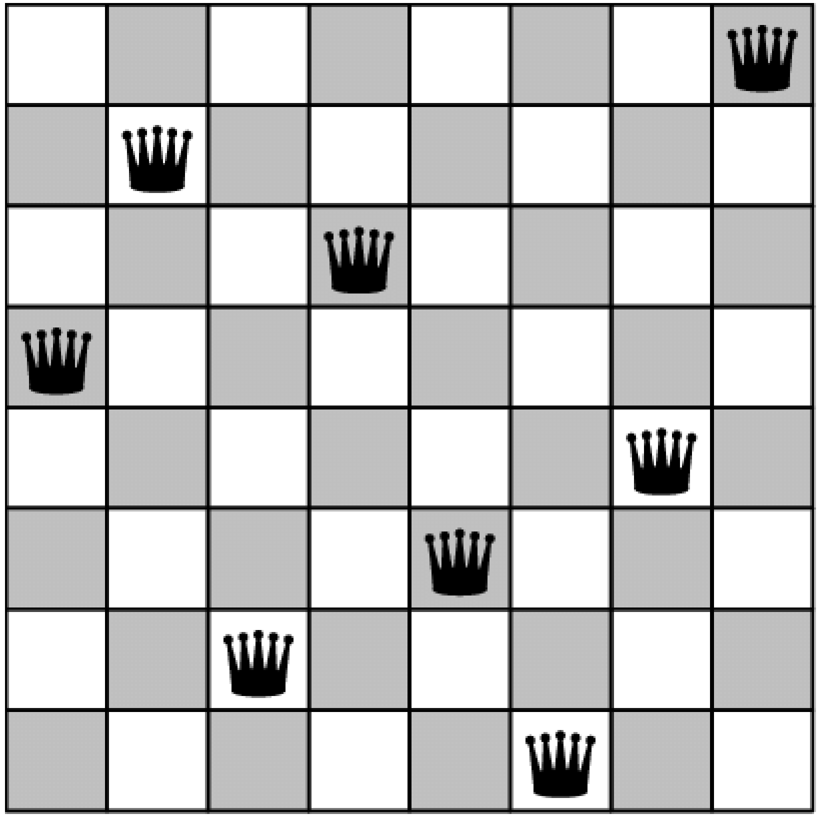
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?
- Exactly one queen for each row
- Exactly one queen for each column
- At most one queen for each diagonal (both directions)
- If we go through the previous slides, we can easily realize that every constraint can be implemented with what we already know



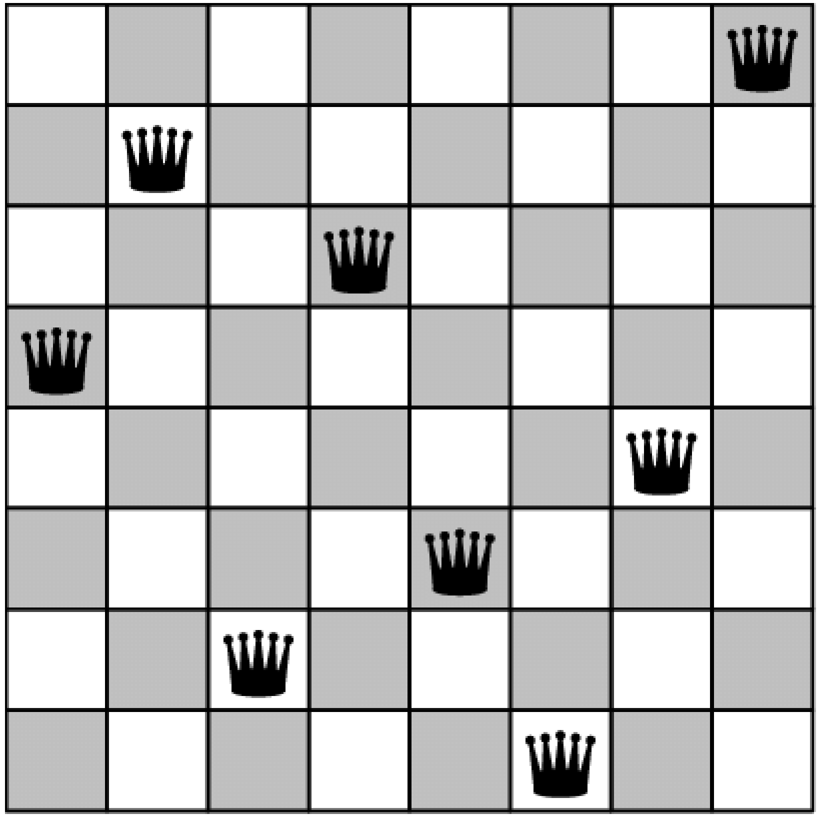
# Exercise 3: N-Queens Puzzle

- Ok, what we have just found can be a good starting point for the construction of the QUBO problem
- Now we have to put some constraints
- Try to think for a few minutes: what could be the necessary constraints?
- Exactly one queen for each row
- Exactly one queen for each column
- At most one queen for each diagonal (both directions)
- If we go through the previous slides, we can easily realize that every constraint can be implemented with what we already know
- The only problem is the large amount of math!



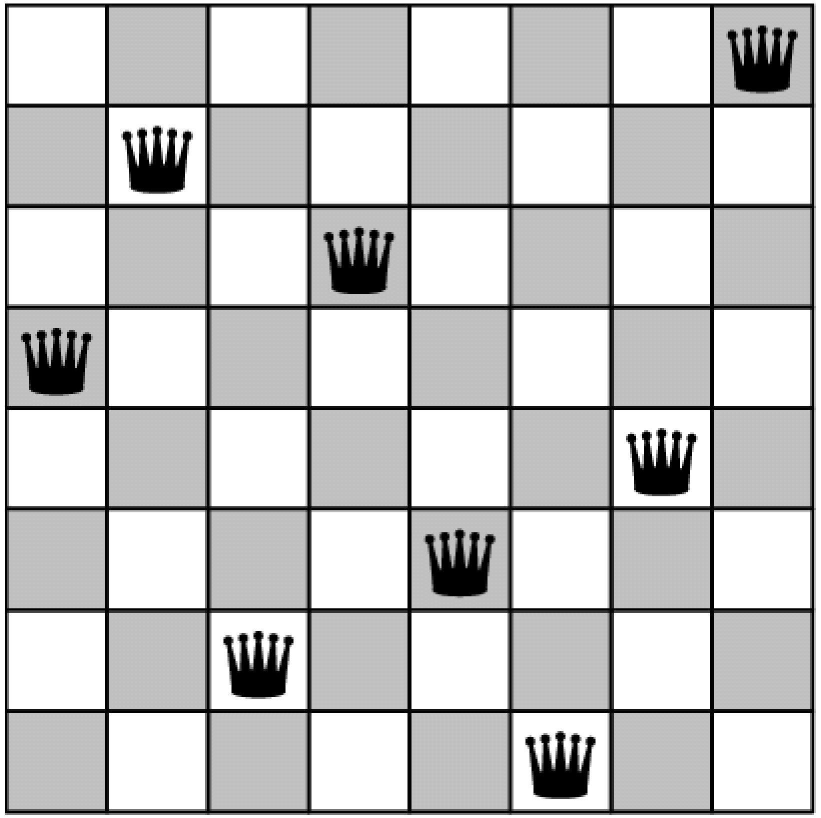
# Exercise 3: N-Queens Puzzle

- Let us then consider the problem from another point of view.



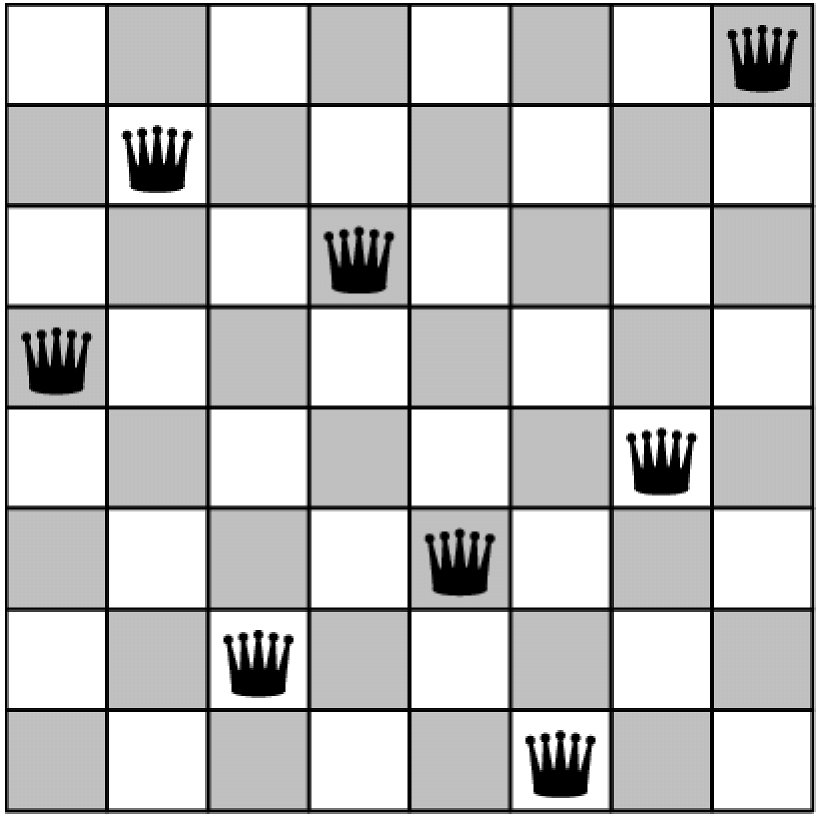
# Exercise 3: N-Queens Puzzle

- Let us then consider the problem from another point of view.
- Instead of mathematically calculating all the constraints, let's do something else



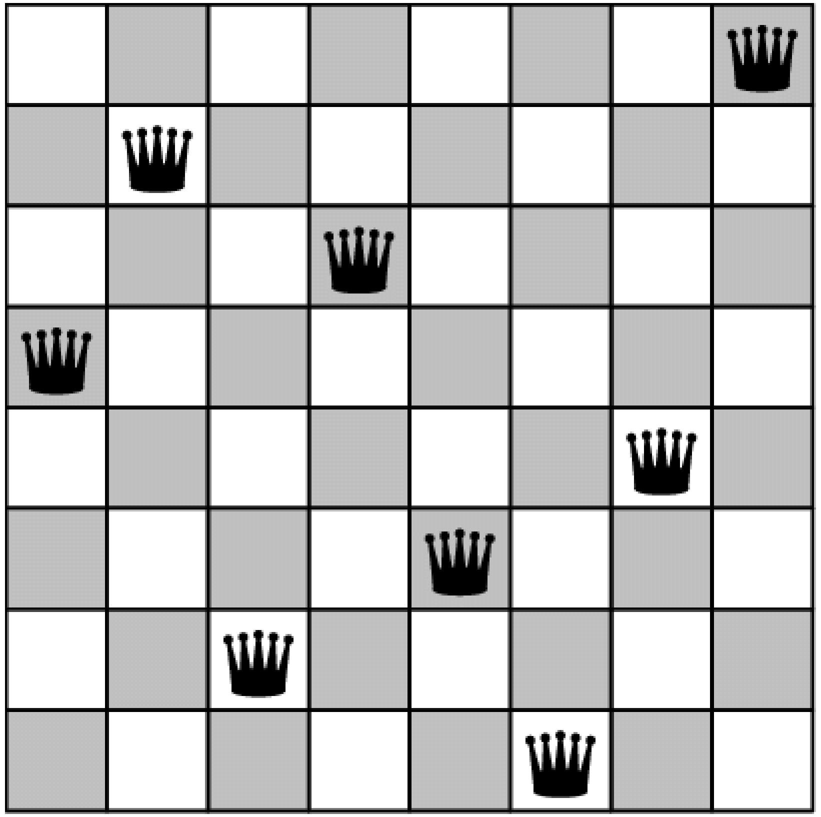
# Exercise 3: N-Queens Puzzle

- Let us then consider the problem from another point of view.
- Instead of mathematically calculating all the constraints, let's do something else
- Let us consider the matrix of the quadratic contributions of the QUBO problem



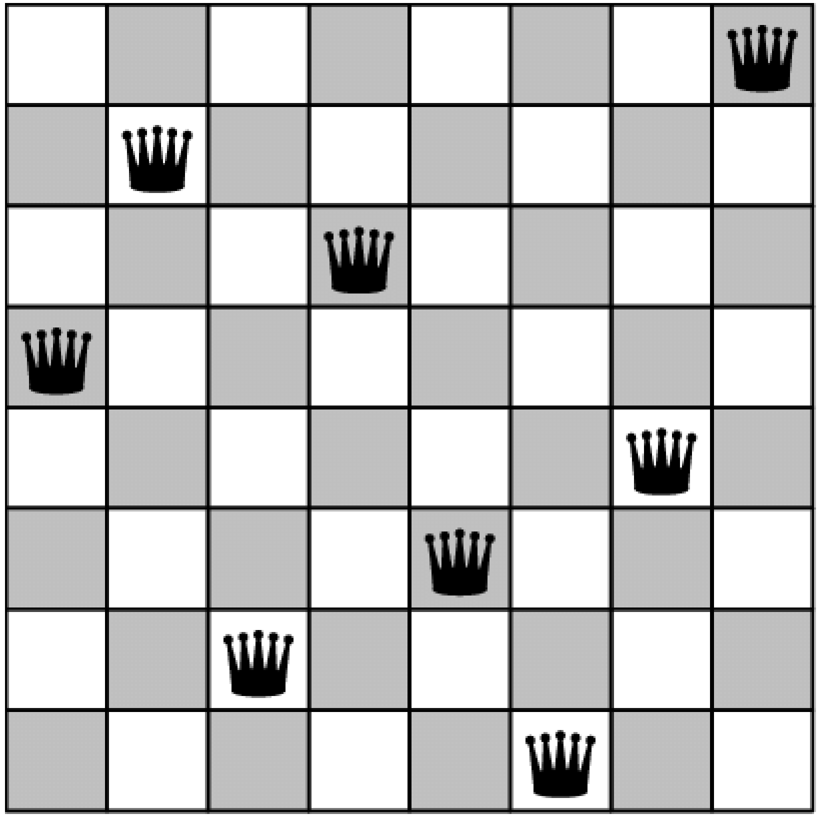
# Exercise 3: N-Queens Puzzle

- Let us then consider the problem from another point of view.
- Instead of mathematically calculating all the constraints, let's do something else
- Let us consider the matrix of the quadratic contributions of the QUBO problem
- This matrix has as elements all the possible pairs of squares on the chessboard (NxNxNxN)



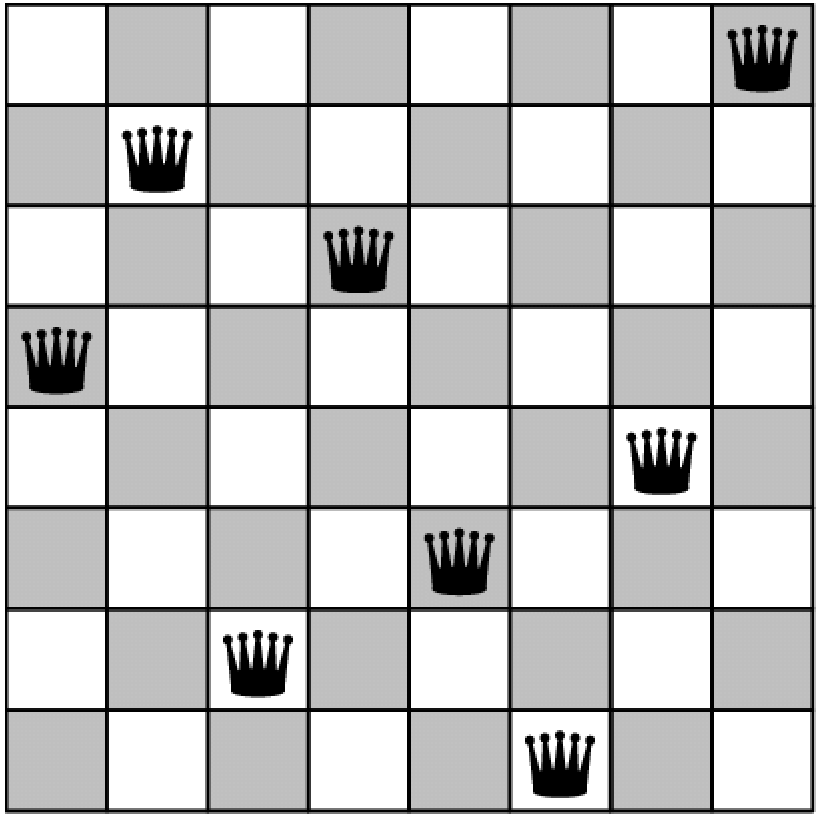
# Exercise 3: N-Queens Puzzle

- Let us then consider the problem from another point of view.
- Instead of mathematically calculating all the constraints, let's do something else
- Let us consider the matrix of the quadratic contributions of the QUBO problem
- This matrix has as elements all the possible pairs of squares on the chessboard (NxNxNxN)
- To implement our constraints, we do the following: we analyze the matrix of the quadratic contributions and, for each pair that is "forbidden", we increase the value of its weight



# Exercise 3: N-Queens Puzzle

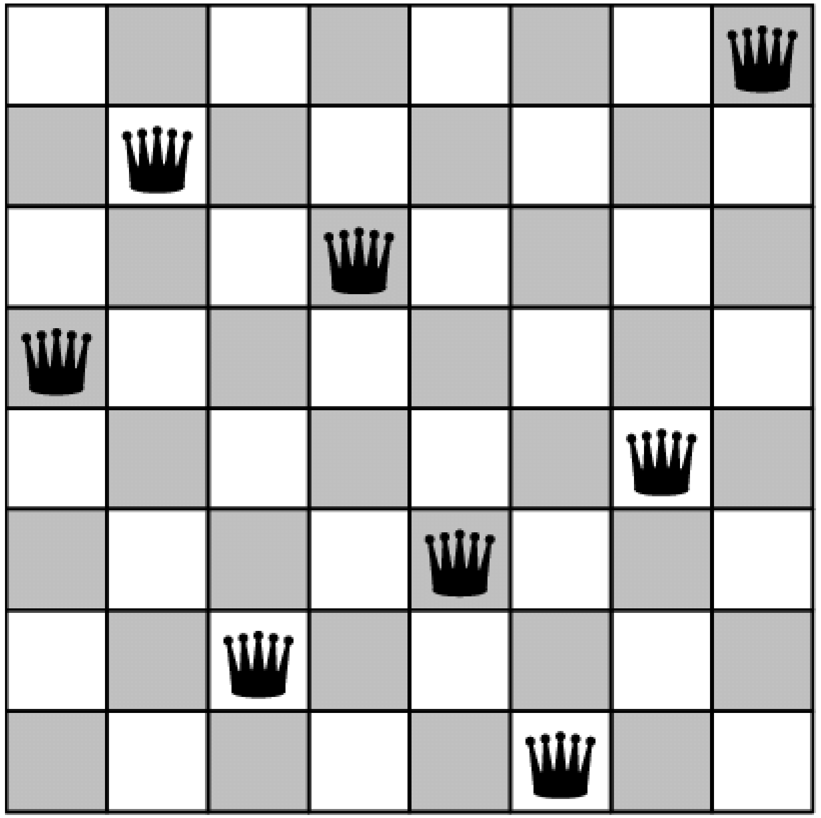
- Let us then consider the problem from another point of view.
- Instead of mathematically calculating all the constraints, let's do something else
- Let us consider the matrix of the quadratic contributions of the QUBO problem
- This matrix has as elements all the possible pairs of squares on the chessboard (NxNxNxN)
- To implement our constraints, we do the following: we analyze the matrix of the quadratic contributions and, for each pair that is "forbidden", we increase the value of its weight
- The weight, by definition, is activated only if both qubits, or squares, are in state 1, i.e. both host a queen



# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

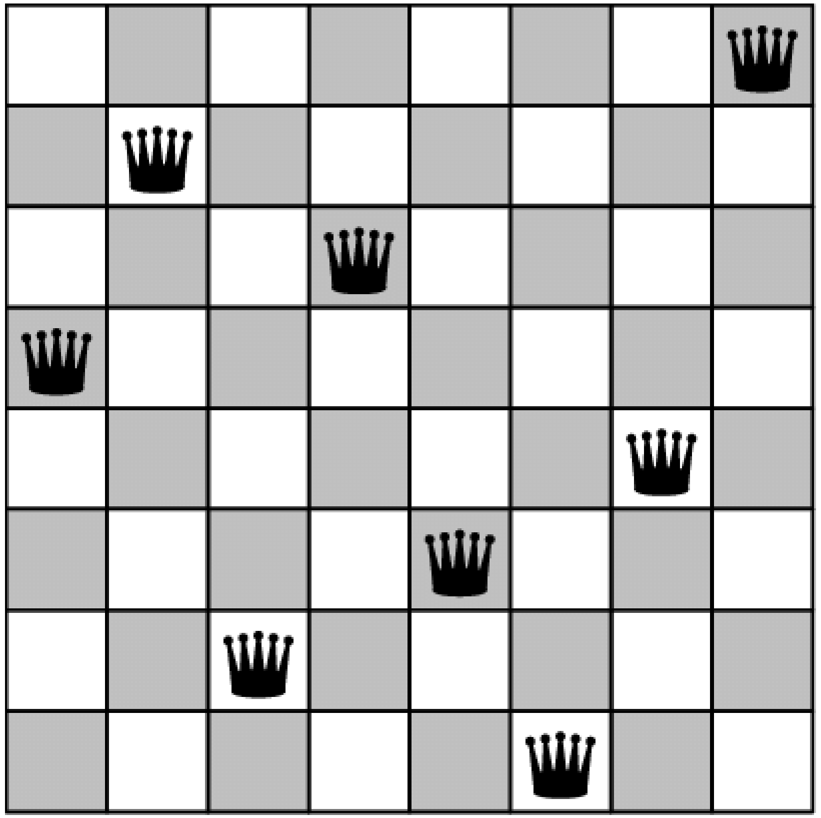


# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

- This function calculates a linearized vector containing all the possible pairs of squares on the board

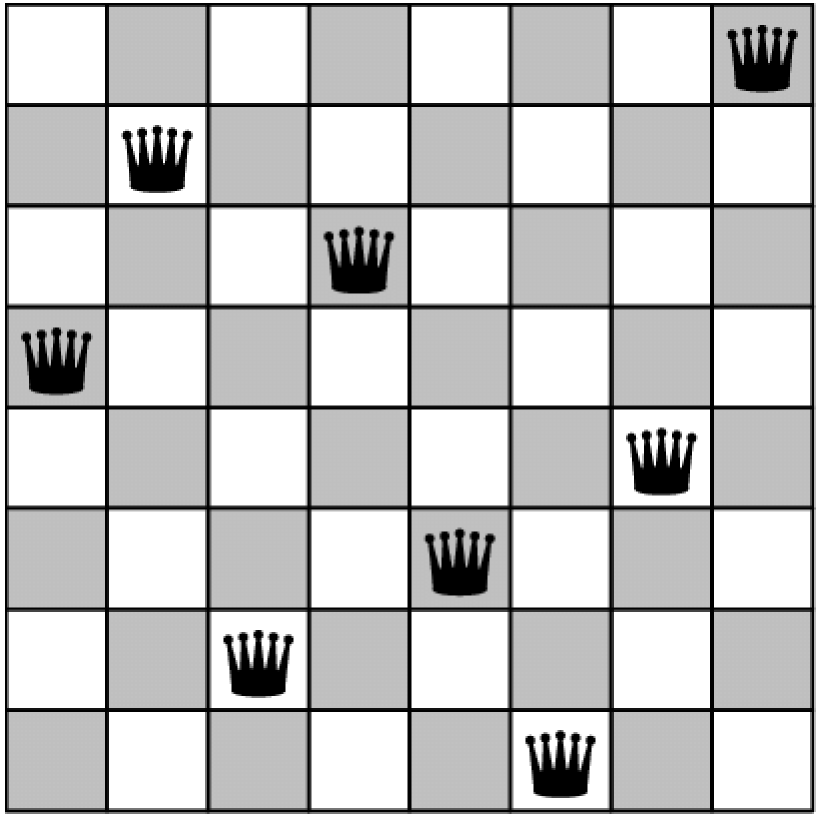


# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

- This function calculates a linearized vector containing all the possible pairs of squares on the board
- Once a specific row has been chosen, the vector will have the value 1 if the pair of squares belongs to the same row, 0 otherwise



# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```python
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

- This function calculates a linearized vector containing all the possible pairs of squares on the board
- Once a specific row has been chosen, the vector will have the value 1 if the pair of squares belongs to the same row, 0 otherwise
- With this definition, I can start building the penalty matrix like this

```python
N=4

w=1

B=np.zeros((N*N,N*N), dtype=float)
for row in range(N):
    R=ROW(row,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+R[i]*R[j]*w
print(B)
```

```
[[0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

- This function calculates a linearized vector containing all the possible pairs of squares on the board
- Once a specific row has been chosen, the vector will have the value 1 if the pair of squares belongs to the same row, 0 otherwise
- With this definition, I can start building the penalty matrix like this
- Basically I'm saying: if two squares are part of the same row, it increases their weight by a factor w

```
N=4

w=1

B=np.zeros((N*N,N*N), dtype=float)
for row in range(N):
    R=ROW(row,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+R[i]*R[j]*w
print(B)
```

```
[[0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

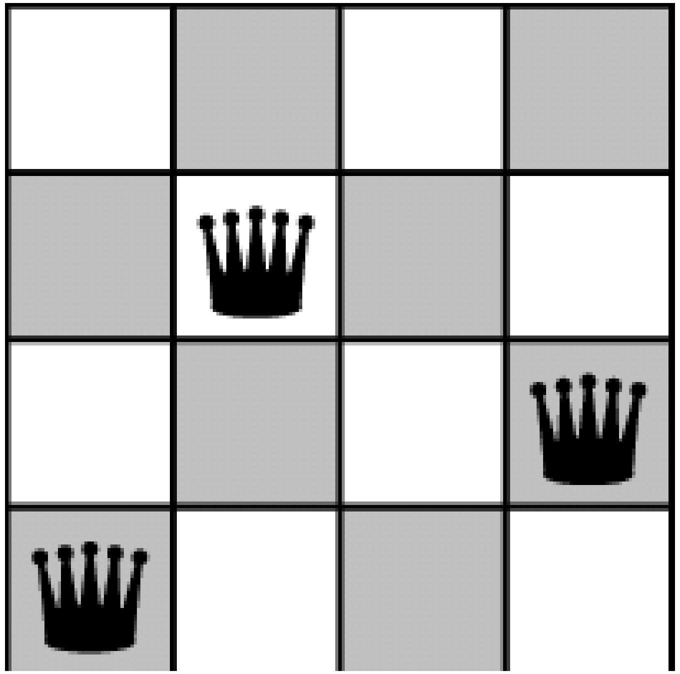
# Exercise 3: N-Queens Puzzle

- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

```
R=ROW(0,4)
```

```
[[1 1 1 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```



```
N=4

w=1

B=np.zeros((N*N,N*N), dtype=float)
for row in range(N):
    R=ROW(row,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+R[i]*R[j]*w
print(B)
```

```
[[0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

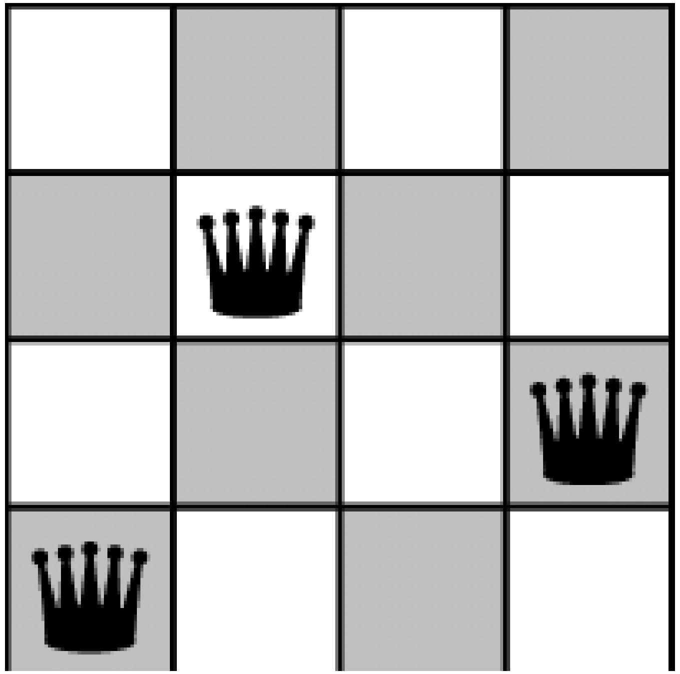
- One way to do this, is to define a function in this way

```
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

```
R=ROW(0,4)
```

```
[[1 1 1 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

```
[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
```



```
N=4

w=1

B=np.zeros((N*N,N*N), dtype=float)
for row in range(N):
    R=ROW(row,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+R[i]*R[j]*w
print(B)
```

```
[[0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

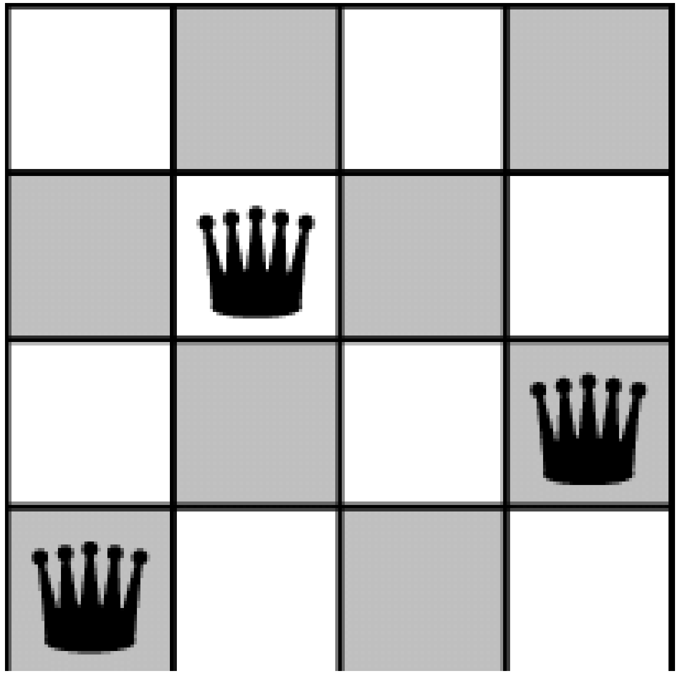
- One way to do this, is to define a function in this way

```python
def ROW(row,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[row,:] = 1
    return C.flatten()
```

```python
R=ROW(1,4)
```

```
[[0 0 0 0]
 [1 1 1 1]
 [0 0 0 0]
 [0 0 0 0]]


[0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0]
```



```python
N=4

w=1

B=np.zeros((N*N,N*N), dtype=float)
for row in range(N):
    R=ROW(row,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+R[i]*R[j]*w
print(B)
```

```
[[0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

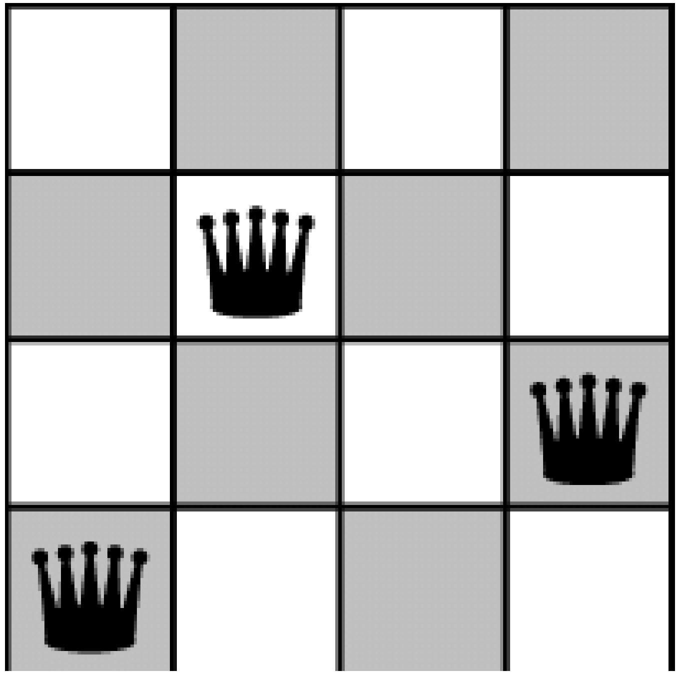
- Same thing for columns

```python
def COL(col,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[:,col] = 1
    return C.flatten()
```

```python
C=COL(0,4)
```

```
[[1 0 0 0]
 [1 0 0 0]
 [1 0 0 0]
 [1 0 0 0]]


[1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0]
```



```python
for col in range(N):
    C=COL(col,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+C[i]*C[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

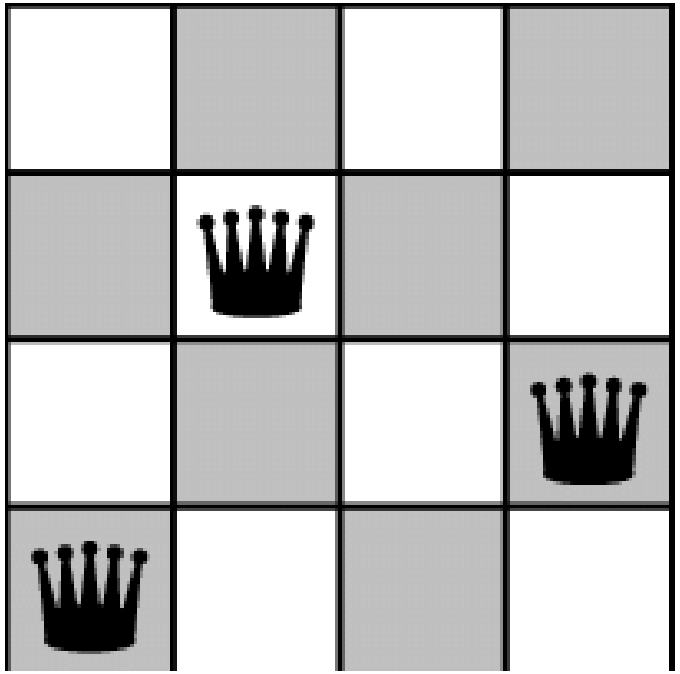
- Same thing for columns

```python
def COL(col,dim):
    C=np.zeros((dim,dim), dtype=int)
    C[:,col] = 1
    return C.flatten()
```

```python
C=COL(1,4)
```

```
[[0 1 0 0]
 [0 1 0 0]
 [0 1 0 0]
 [0 1 0 0]]


[0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0]
```



```python
for col in range(N):
    C=COL(col,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+C[i]*C[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

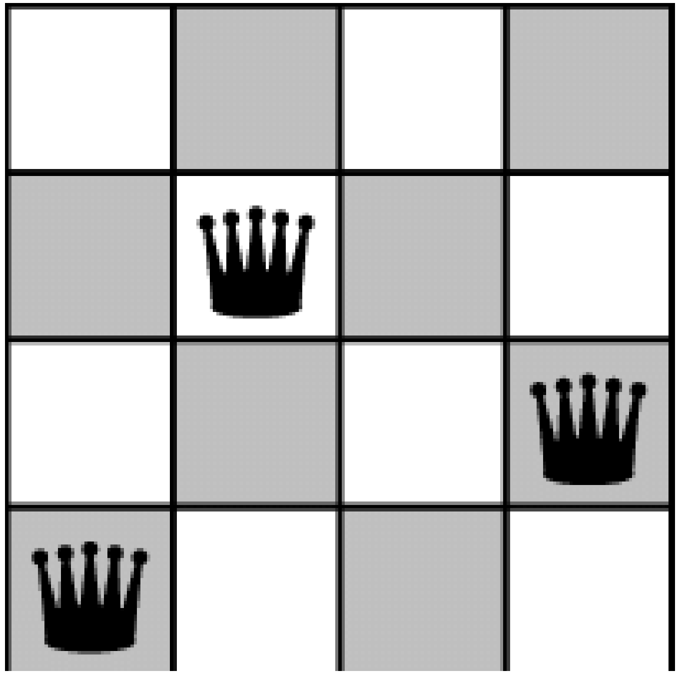
- Same thing for diagonals

```python
def DIAG1(k,dim):
    d=np.ones(dim-abs(k), dtype=int)
    C=np.diag(d,k=k)
    return C.flatten()
```

```python
D=DIAG1(0,4)
```

```
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]


[1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1]
```



```python
for diag1 in range(-(N-2),N-1):
    D1=DIAG1(diag1,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+D1[i]*D1[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1.]
 [0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

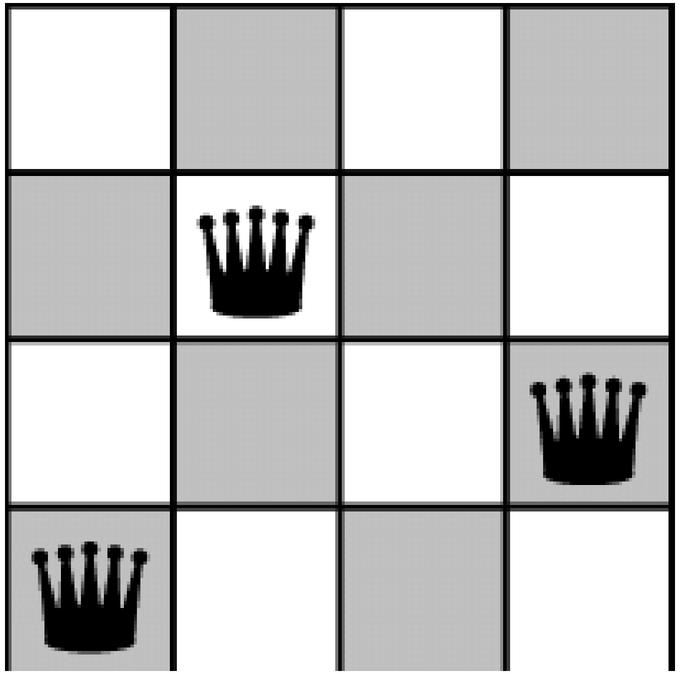
- Same thing for diagonals

```python
def DIAG1(k,dim):
    d=np.ones(dim-abs(k), dtype=int)
    C=np.diag(d,k=k)
    return C.flatten()
```

```python
D=DIAG1(1,4)
```

```
[[0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]
 [0 0 0 0]]


[0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0]
```



```python
for diag1 in range(-(N-2),N-1):
    D1=DIAG1(diag1,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+D1[i]*D1[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1.]
 [0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

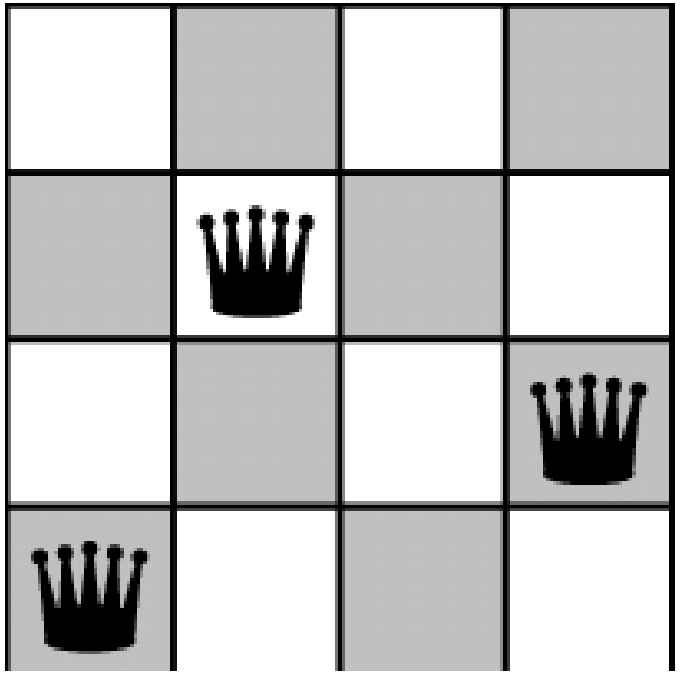
- Same thing for diagonals

```python
def DIAG1(k,dim):
    d=np.ones(dim-abs(k), dtype=int)
    C=np.diag(d,k=k)
    return C.flatten()
```

```python
D=DIAG1(-1,4)
```

```
[[0 0 0 0]
 [1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]]


[0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0]
```



```python
for diag1 in range(-(N-2),N-1):
    D1=DIAG1(diag1,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+D1[i]*D1[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1.]
 [0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

# Exercise 3: N-Queens Puzzle

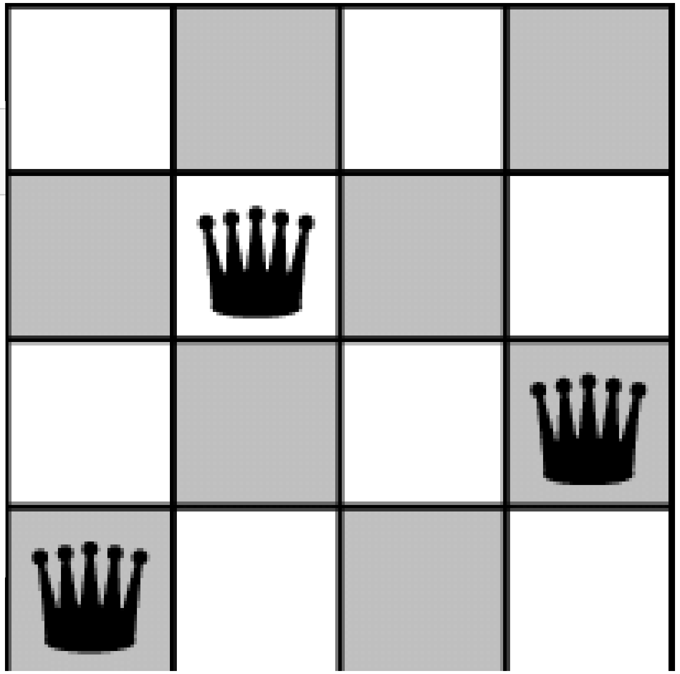
- Same thing for diagonals

```python
def DIAG2(k,dim):
    d=np.ones(dim-abs(k), dtype=int)
    C=np.fliplr(np.diag(d,k=k))
    return C.flatten()
```

```python
D=DIAG2(0,4)
```

```
[[0 0 0 1]
 [0 0 1 0]
 [0 1 0 0]
 [1 0 0 0]]


[0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0]
```



```python
for diag2 in range(-(N-2),N-1):
    D2=DIAG2(diag2,N)
    for i in range(N*N):
        for j in range(i+1,N*N):
            B[i][j]=B[i][j]+D2[i]*D2[j]*w

print(B)
```

```
[[0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1.]
 [0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

SuperComputing Applications and Innovation

**Quantum Annealing with continuous variables:**
**Low-Rank Matrix Factorization**

Daniele Ottaviani
CINECA

**Quantum Computing and High Performance Computing**
CINECA Casalecchio di Reno, Bologna, 18-12-2018

# QUBO Problems with real variables

**We define a QUBO problem with real variables as a Quadratic Unconstrained Optimization problem with unknown variables expressed as:**

$$x = c \cdot \sum_{e=0}^{N-1} 2^e q_e, \quad c = 10^{-a}, \text{ for some } a \in \mathbb{N}$$

For example, the QUBO problem associated with the simple equation $x - b = 0$ is:

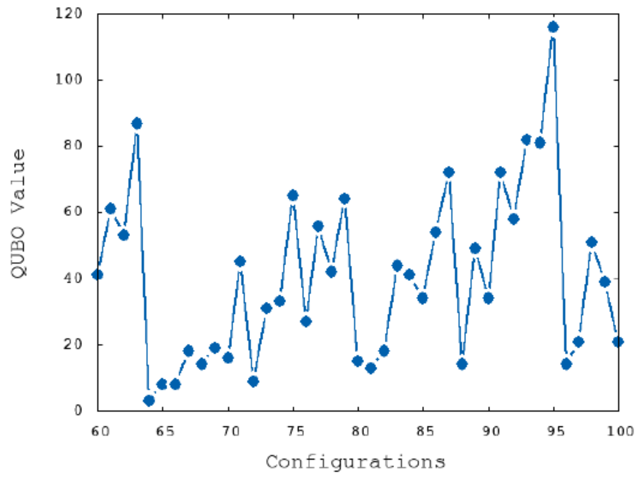
$$\min_{\mathbf{q}=(q_0,\ldots,q_{N-1})} \left( \sum_{e=0}^{N-1} \left(c^2 2^{2e} - bc2^{e+1}\right) q_e + \sum_{e<f} \left(c^2 2^{e+f+1}\right) q_e q_f \right)$$

Considering $x - b = 0$ as $\min_{x \in \mathbb{R}} (x - b)^2$

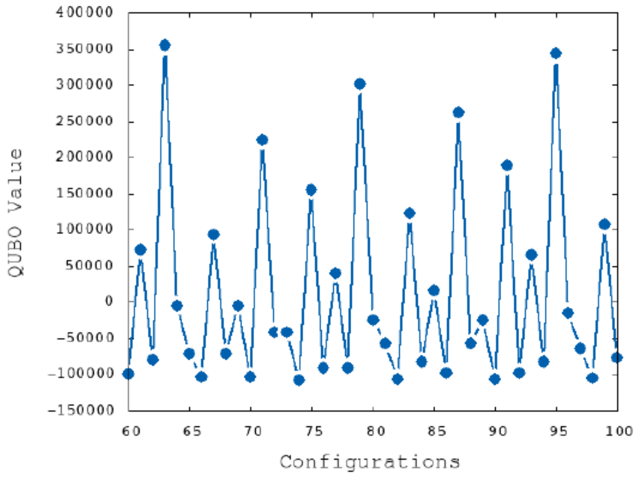
# Graphical representation

**QUBO problems of this kind are particularly difficult to solve.**
**Especially with annealing techniques.**

**This is due to the exponential dependence of the coefficients from the binary variable indices, which create numerous local minima very similar to the global minimum.**




**"Normal" QUBO landscape**




**"Real-variables" QUBO landscape**

# Solving a linear system

We have chosen to solve a linear system $A\mathbf{x} = b$, where

$\mathbf{x} = (x_1, x_2, x_3)$ and $x_i \in [0,1]$.

We represent $x_i = c \cdot \sum_{e=0}^{9} 2^e q_e, \quad c = 10^{-3}$ $(N = 10,\ a = 3)$.

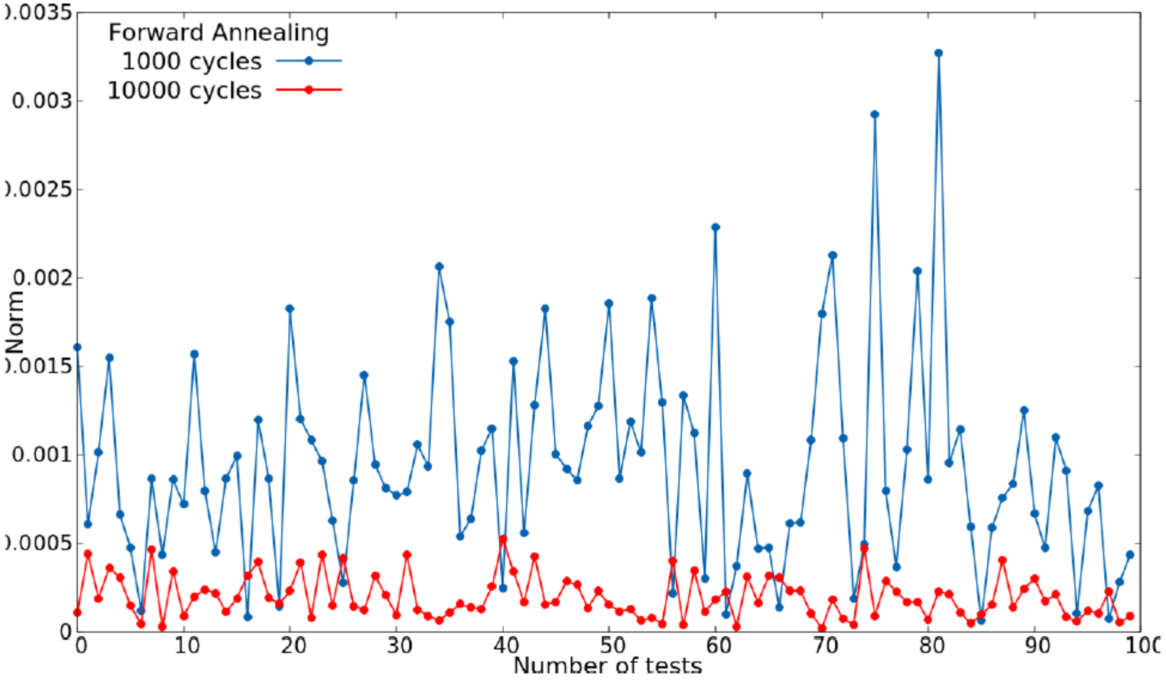
We will find $\mathbf{x}$ solving $\min_{\mathbf{x}\in[0,1]^3} ||A\mathbf{x} - b||_2^2$

$$\begin{bmatrix} 1.301 & 0.125 & 0.187 \\ 0.440 & 0.342 & 0.082 \\ 0.672 & 0.709 & 0.802 \\ 0.218 & 0.427 & 0.520 \\ 0.024 & 0.036 & 0.038 \end{bmatrix} \cdot \begin{bmatrix} 0.178 \\ 0.333 \\ 0.489 \end{bmatrix} = \begin{bmatrix} 0.365 \\ 0.232 \\ 0.748 \\ 0.435 \\ 0.035 \end{bmatrix}$$

# Advanced Annealing Techiques

## Attempt number 1: Forward Annealing

100 attempts with 1,000 and 10,000 annealing cycles




$$\begin{bmatrix} 0.179 \\ 0.338 \\ 0.483 \end{bmatrix}$$

$$\left(\begin{bmatrix} 0.178 \\ 0.333 \\ 0.489 \end{bmatrix}\right)$$

# Local refinement of solutions:

## Reverse Annealing

**Introduced with the last D-Wave model, DWAVE2000Q**

**Starting point chosen by the user**

**Backward Annealing**

**Forward Annealing**

**During the Backward Annealing phase, the transverse field slowly increases up to a value chosen by the user (*Reversal Distance)***

**The last Forward Annealing phase is a LOCAL quantum annealing search: how much local depends on the reversal distance value.**

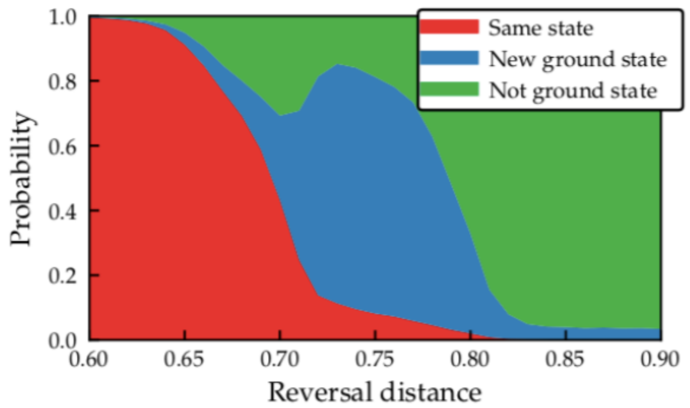
Image taken from *Reverse Quantum Annealing for Local Refinement of Solutions,* D-Wave White Papers, 2017

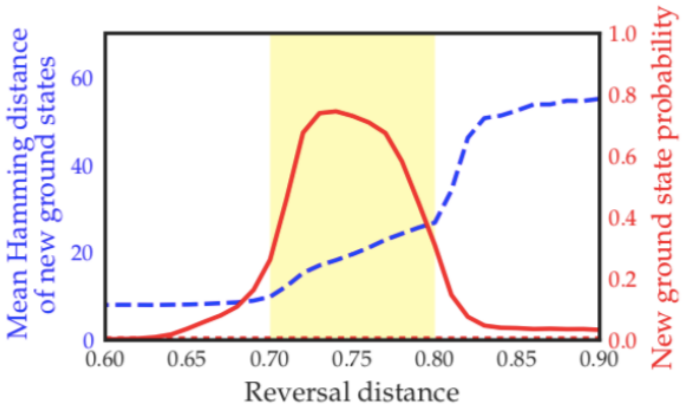
# Tuning the reversal distance

D-WAVE
The Quantum Computing Company™

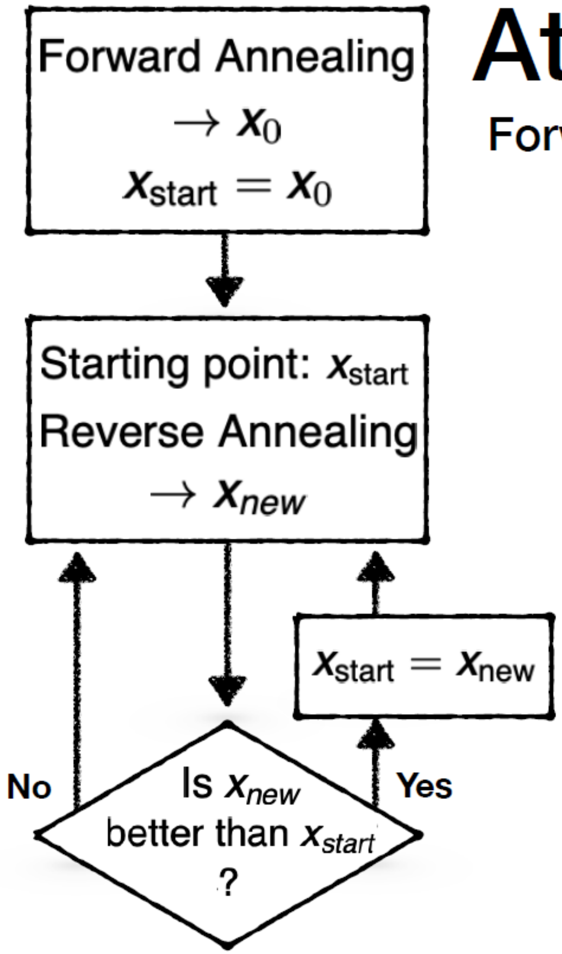
Reverse Quantum Annealing for Local Refinement of Solutions

WHITEPAPER



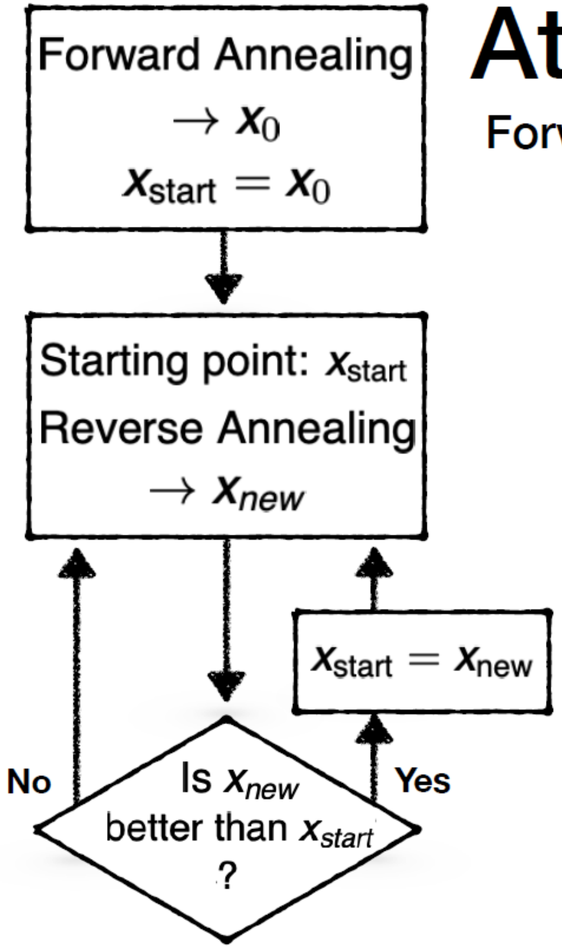
# Advanced Annealing Techiques




## Attempt number 2:

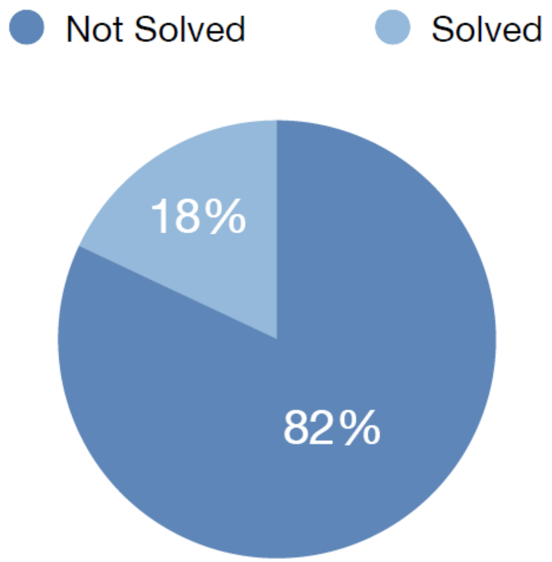
Forward Annealing + Reverse Annealing

# Advanced Annealing Techiques




## Attempt number 2:

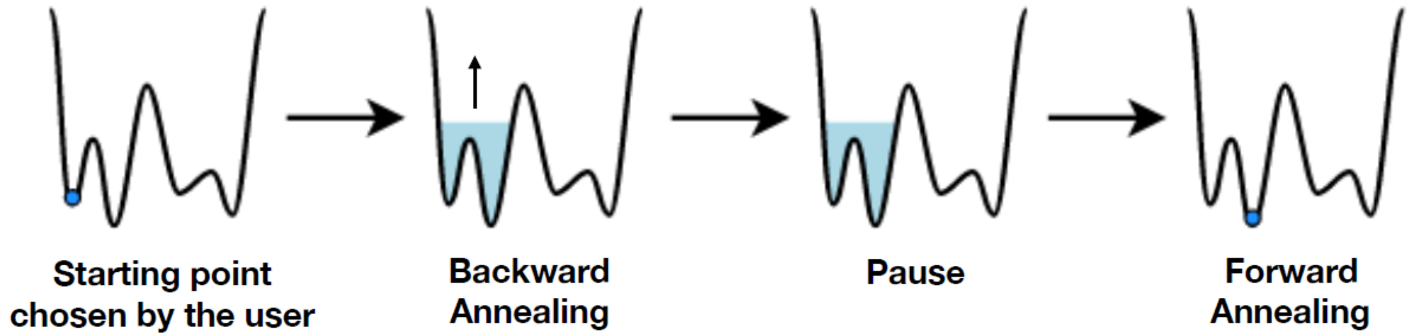
Forward Annealing + Reverse Annealing



# Pausing the annealing process

Being able to pause the annealing process is another of the new features introduced with the latest D-WAVE quantum annealer.

We can use the pause during a Reverse Annealing search in this way:




Why pause? Because pausing the annealing process means better exploration of the selected zone, increasing the chances of obtaining a new global minimum.

But pay attention: pause can't be too long. For two main reasons:
1) it increase the computational time of each annealing cycle.
2) if it is too long, it may also risk to increase the search radius more than desired.

## Correlation between pause and search radius

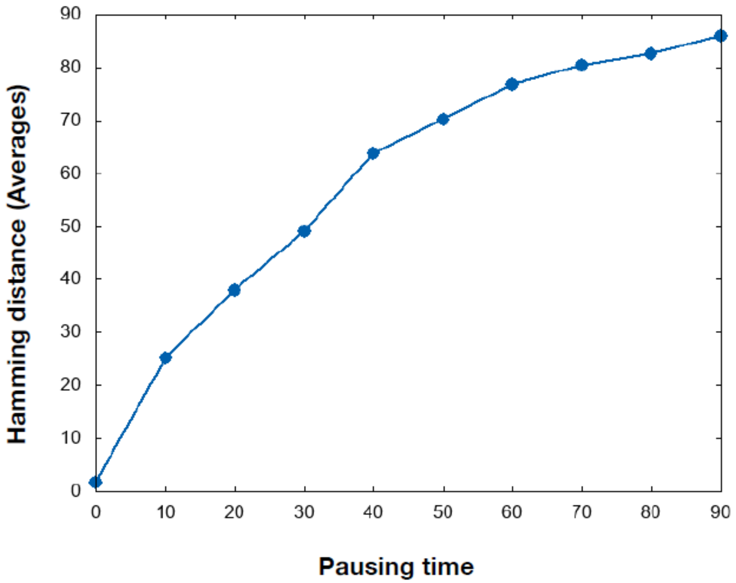
**We can realize a posteriori the search radius of a reverse annealing search by analyzing the average distance between the solutions found by each cycle.**

**To do this, we choose the Hamming distance, a function written to calculate the distance between vectors of binary numbers.**

**We have observed that there is a correlation between the pause time and the average distance between the solutions obtained with each annealing cycle**

**As with the reversal distance, here too we have to be careful about the right break time:**

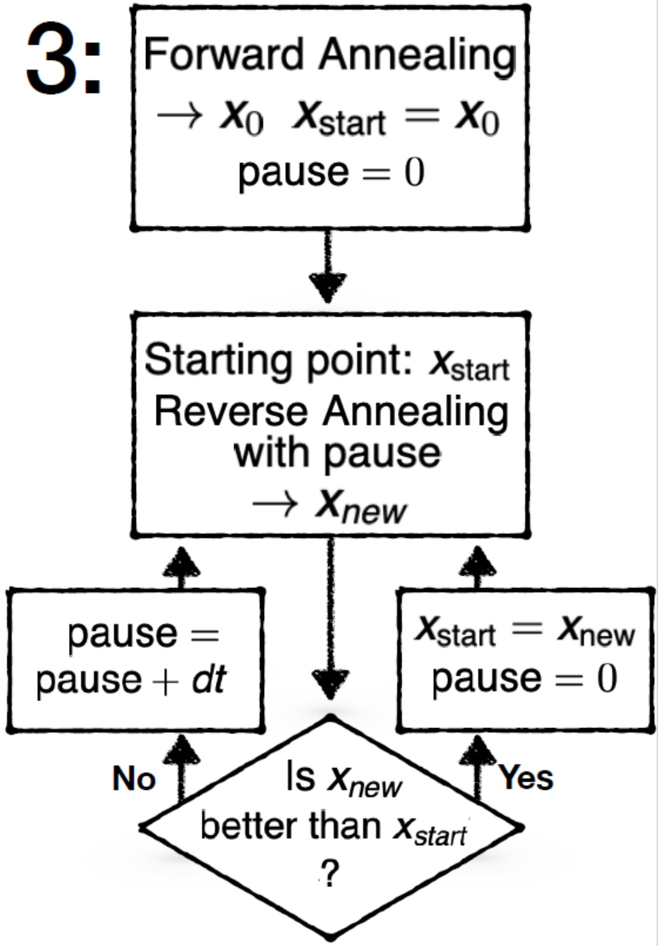
**too little is not enough,**
**too much can lead to wrong results**



# Advanced Annealing Techiques

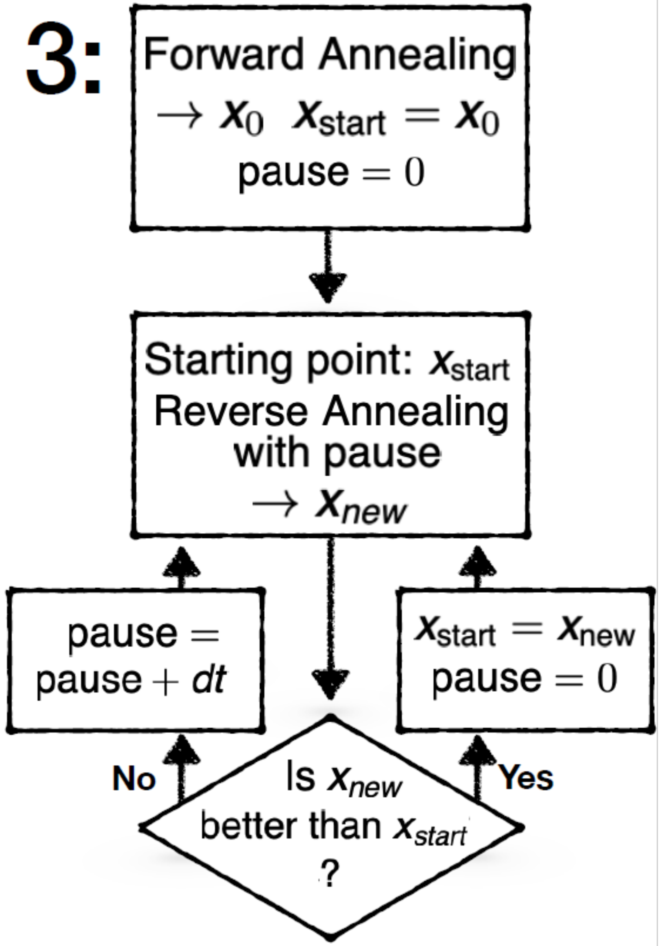
## Attempt number 3:

Forward Annealing +
Reverse Annealing with pause



# Attempt number 3:

Forward Annealing +
Reverse Annealing with pause



# Advanced Annealing Techiques

## Attempt number 3:

**Forward Annealing +**
**Reverse Annealing with pause**