POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di Laurea Magistrale

Solving the binary optimization problem for linear regression with Grover Adaptive Search

Relatori:
Prof. Maurizio Zamboni
Prof. Mariagrazia Graziano
Prof. Giovanna Turvani

Candidato:
Luigi Giuffrida

Dicembre 2021

To my loved ones

Summary

Quantum computing can solve in some cases optimization problems faster than classical one, since quantum mechanics phenomena could speed up the optimum search; in particular, it suits the formulation of Quadratic Unconstrained Binary Optimization (QUBO) problems, which can model many scenarios, such as finance and machine learning. The problem to be solved is mapped onto a cost function over binary variables, with weights involving at most couples of bits.

QUBO problems can be solved with both quantum annealers and general-purpose gate-array-based quantum computers. The former mimics the behaviour of simulated annealing in a quantum framework; the latter exploits quantum superposition to explore the solution space concurrently, as the Grover Adaptive Search (GAS) algorithm, which is the object of this thesis, does through Grover Search (GS) algorithm.

GS algorithm allows to find one or more items in an unordered database, mapped onto a superposition of qubit states, by labelling the elements to be found and amplifying their probability amplitude with interference mechanism. In the context of interest, a quantum dictionary circuit, entangling a first set of qubits (keys register) to a second one (values register), is exploited for obtaining a superposition of all domain-image couples of the QUBO function, and GS finds negative values of it.

GAS is a hybrid quantum-classical algorithm which iteratively exploits GS for finding the optimal value. In particular, after the initialization of the QUBO function at the beginning of the algorithm, two steps are repeated until no negative values are present in the function and the optimal solution has a null value: execution of GS to sample a negative value of the cost function and translation of it by subtracting the value of the last measured sample.

The goals of this thesis are the design and the implementation of GAS and its exploitation for solving linear regression problems, described as a QUBO model, identifying the quadratic error between the points and the interpolating line as the cost function. The quantum circuit for GS has been implemented in Python using the ETH Zurich's framework ProjectQ and it has been tested on a noiseless qubits classical simulator, as the large number of the needed qubits precludes the execution on the available quantum computers.

The obtained results have shown that, in order to maximize the probability of

sampling a negative value, the central routine of GS must be iterated a number of times depending on the search domain and that, unfortunately, this is not known a priori. Moreover, another crucial point of the GAS algorithm is identifying the achievement of the minimum, i.e. the end of the algorithm. Different strategies to optimize both the degrees of freedom have been studied. It has been ascertained that the most reasonable approaches are choosing randomly the GS iterations and evaluating multiple times the GS result, in the same iteration of GAS, to avoid a false minimum.

Ultimately, the GAS results for linear regression problems have been compared with those of Quantum and Simulated Annealing. GAS has been proven to get higher success probability than Quantum Annealing.

It is expected that GAS could be improved by establishing rules to forecast the best parameters for the algorithm, also taking into account the QUBO problem to be solved. Exploiting statistical calibrations or formulating machine learning models are possible approaches for obtaining these.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

1.1 Motivation

Linear regression [1] is a linear method to find a relation between a scalar response and a set of variables. In particular, a single variable linear regression can be used to find the straight line interpolating a set of points (like the one in Figure 1.1).

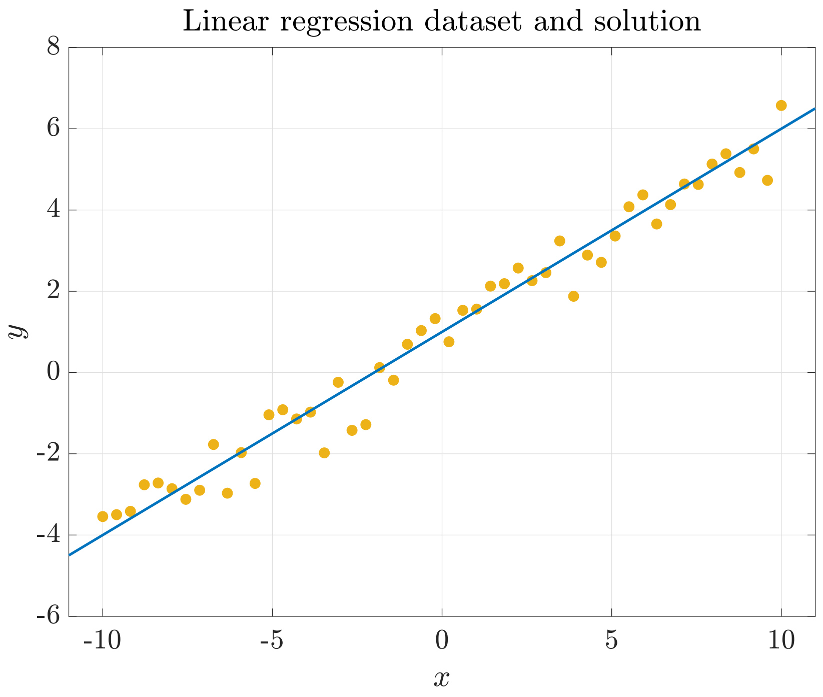


Figure 1.1: Linear regression example

Due to its diffusion and conceptual simplicity, it is a fundamental tool in many scientific applications, so it is very important to have the possibility of efficiently solving this problem. In particular, linear regression problems can be used in many different applications, like machine learning models. Moreover, it can be used to study many real-life scenarios, from security [2], to chemistry [3].

The definition of the interpolating line parameters has an analytical solution, but its calculation is extremely expensive from a computational point of view, indeed it requires many matrix multiplications and inversions, which are known to belong to the set of the most resource demanding numerical operations. Some alternatives must be found.

The training of a linear regression model can be studied as an optimization problem. In fact its target is to find the line that has the minimum distance to the dataset points. In particular, a **combinatorial optimization** formulation of this problem was proposed by P. Date in [4]. **Quantum computing** is a promising paradigm to solve combinatorial optimization problems.

The most promising approaches are based on **quantum annealing** and **quantum-gate-array** based techniques. Quantum annealing mimics what **simulated annealing** does, but it exploits quantum fluctuations instead of thermal perturbations to explore the cost function and find the optimum. **Grover Adaptive Search** (GAS) algorithm, the main focus of this thesis, is a mixed approach that exploits the Grover's algorithm [5], a quantum search algorithm implemented on Quantum Gate Array, using its output in a classical context to find the solution to an optimization problem.

Quantum annealing presents some limitations, which are analysed in this thesis. GAS algorithm could overcome them. Moreover, industries are moving in the direction of general-purpose quantum-gate-array computing [6] since this presents a more promising scaling trend than quantum annealing hardware. Therefore it is important to test and validate techniques that follow this direction and move the optimization problem solution from a special-purpose hardware to a general-purpose one.

This thesis has the objective to implement and validate the GAS algorithm as a possible solver of the linear regression training problem and in general combinatorial optimization problems. The effect of different algorithm parameters is tested and some strategies to select them are presented. Ultimately, the results obtained by the GAS algorithm are compared to the one of quantum and simulated annealing, to prove how effective this technique can be.

1.2 Workflow

In order to implement the GAS algorithm, the workflow reported in Figure 1.2 have been followed. Each step is detailed in the following:

1. **State of art analysis**: different optimization techniques to solve combinatorial optimization problems have been studied and analysed, with a particular focus on **Simulated Annealing**, **Quantum Annealing** and **Grover Adaptive Search**. Then, different **QUBO** formulations of machine learning related problems have been analysed and studied. The linear regression training problem has been chosen as a use case and target problem.

2. **Problem formulation validation**: Once that the target problem has been selected, a software description of it has been produced and validated. During this phase, simulated annealing has been used as the preferred solver to test the correctness of the produced formulation.

3. **GAS implementation**: The Grover Adaptive Search algorithm has been implemented. A stop policy is proposed, to determine that a minimum has been achieved.

4. **GAS validation**: The implemented version of the algorithm has been tested to solve the linear regression formulation.

5. **GAS parameter test**: The design parameters of the Grover Adaptive Search algorithm have been modified to test their effect on the result and to find the

optimum to have a tradeoff between the correctness of the solution and time required to obtain it.

6. **GAS benchmark and comparison**: In the end, the GAS algorithm has been benchmarked against simulated and quantum annealing ones to find if it can introduce an effective improvement in solving optimization problems.

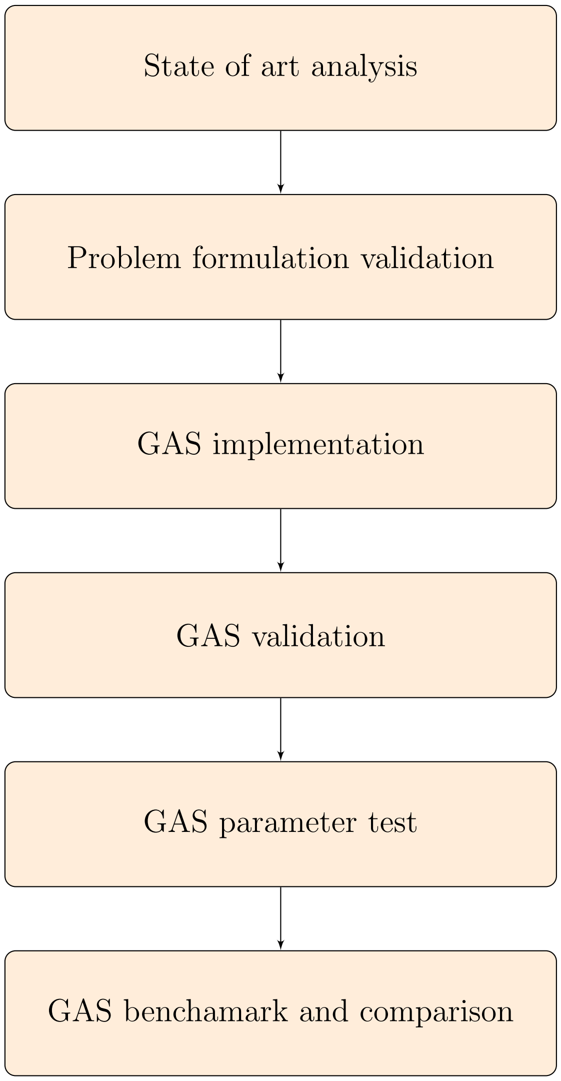


Figure 1.2: Thesis workflow

Chapter 2

Quantum computing

In the last years the importance of supercomputers, which can perform computation faster than other computing systems, has grown significantly. For some applications the speed-up introduced by these is not enough, because computational complexity grows too rapidly with the amount of data involved, thus the exploration of a new computational paradigm is required.

Quantum computing is a promising approach to computation problem. It was firstly conceptualized by Richard Feynman in [7], who proposed to use a computer based on quantum phenomena to simulate quantum systems. Now, after 40 years, quantum computing is possible and it can be used not only to simulate physical systems but to solve problems with a computational cost too high to be handled by classical supercomputers.

This has lots of possible applications, from pure physical simulations [8] to cryptography [9] and optimization [10].

In order to better understand the fundamentals of quantum computation, an overview on qubits is presented, this includes some notes on their representation and properties. Then quantum gates and some algorithms are presented, to show the advantages of this computational paradigm.

2.1 Qubits

A classical deterministic computing system relies on the fact that information is either 0 or 1, while quantum computers exploit quantum superposition and **qubits** (quantum equivalent of bits) can assume potentially infinite states given by the linear combination of two basis states:

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{2.1}$$

thus a qubit is represented as:

$$|q\rangle = c_0 |0\rangle + c_1 |1\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}, \tag{2.2}$$

where c_0 and c_1 represent complex probability amplitudes of the qubit to be in the state 0 or 1 respectively.

They reveal their effective value after a measurement event, which makes them collapse in a classical deterministic state 0 or 1 with a probability of $|c_0|^2$ and $|c_1|^2$.

A vector, in which the probabilities of being 1 and 0 are 1 and 0 respectively, corresponds to the *classical* 1, while *classical* 0 is represented by a vector with complementary probabilities. The deterministic nature of this representation is encapsulated in the fact that probabilities are always 1 or 0.

The state of a qubit is formally represented by a vector in a two-dimensional complex vector space, the states $|0\rangle$ and $|1\rangle$ are the **eigenstates** of a qubit and build an orthonormal basis of the space, so every qubit can be represented as a linear combination of these two states, they are called **computational basis states**.

Due to this peculiar information representation, it is possible to outperform classical computers. In fact, quantum computers can represent a problem in a multidimensional space given by the combination of different qubits and, after some manipulation, extrapolate the solution to this.

The vector composed by c_0 and c_1 represents the state of a qubit and it is called **state vector**. To represent qubits, the Dirac notation is adopted, where $|\cdot\rangle$ represents the state vector of a quantum system (a qubit here) as a column vector and $\langle\cdot|$ represents its conjugate transpose.

More details about the mathematical model for the description of qubits can be found in [11] and [12].

2.1.1 Bloch sphere

A quantum state can be represented in a graphical and geometrical way using the Bloch sphere, depicted in Figure 2.1. Each point on the surface of this is a possible state of the qubit.

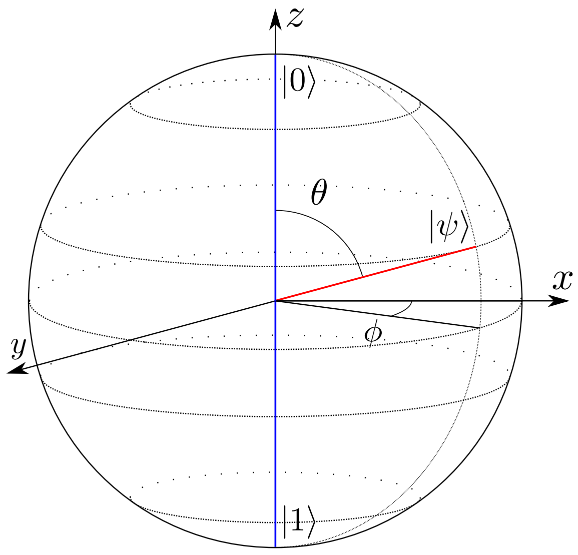


Figure 2.1: Bloch sphere

Expressing the complex numbers c_0 and c_1 in polar form, is possible to rewrite the state of a generic qubit as:

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle = r_0 e^{j\phi_0} |0\rangle + r_1 e^{j\phi_1} |1\rangle \,, \tag{2.3}$$

where r_0 and r_1 are the magnitudes of c_0 and c_1 respectively, while ϕ_0 and ϕ_0 are their phases.

It is possible to demonstrate that $r_0^2 + r_1^2 = 1$, so Equation (2.3) can be rewritten to:

$$|\psi\rangle = \cos(\theta)|0\rangle + \sin(\theta)e^{j\phi}|1\rangle, \tag{2.4}$$

where:

$$\cos(\theta) = r_0, \tag{2.5}$$
$$\sin(\theta) = r_1, \tag{2.6}$$
$$\phi = \phi_1 - \phi_0. \tag{2.7}$$

It is possible to describe the state of a qubit by two angles, θ and ϕ in Figure 2.1, acting on θ will affect the probability of the measurement to be $|1\rangle$ or $|0\rangle$, while acting on ϕ will change the phase of the qubit not affecting the probability of measurement.

Two notable states are:

$$\begin{aligned} |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \\ |-\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned} \tag{2.8}$$

$|+\rangle$ is the point on the Bloch's sphere that intercepts the positive x axis, while $|-\rangle$ intercepts the negative x axis. In these states the probability to collapse to $|1\rangle$ or $|0\rangle$ is the same.

2.1.2 Representing more than one qubit

The importance and potential of superposition are more evident considering n qubits together, which can be described as a linear combination of the 2^n measurable states.

The state vector can be used to represent more than one qubit too. In a system composed of n qubits, it is the tensor product of the state vectors of each qubit, so the state is represented by 2^n complex probabilities whose magnitude square values are the probability of the system to collapse to a particular state after a measurement event (one of the 2^n possible combinations), as shown in Equation (2.9):

$$|\psi\rangle = \begin{array}{r} \ldots\mathbf{0000} \\ \ldots\mathbf{0001} \\ \ldots\mathbf{0010} \\ \ldots\mathbf{0011} \\ \vdots \\ \ldots\mathbf{1111} \end{array} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{2^n} \end{bmatrix}, \tag{2.9}$$

where the notation is the same adopted by [11].

A quantum state can be represented by a discrete random variable. In fact, every possible combination of basis states $|1\rangle$ and $|0\rangle$ has its own probability ($|c_i|^2$) to appear after a parallel measurement of the qubits. From a high-level point of view, a quantum algorithm aims to influence the probability distribution of states in order to increase the probability of those that satisfy specific requirements, dependent on the target of the algorithm, with respect to the probability of the others.

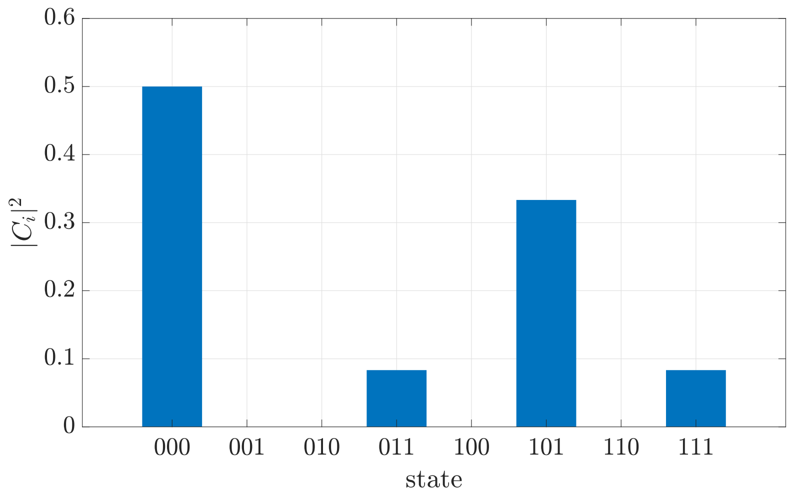


Figure 2.2: An example probability distribution

Figure 2.2 represents an example probability distribution in which on the x axis there are the basis states and the y axis represents the probability that the quantum system collapses in each basis state.

2.2 Qubit properties

Quantum computing exploits different quantum effects, mainly *superposition* and *entanglement*.

Superposition is an intrinsic property of quantum nature. In contrast with classical physics, the state of a quantum system is described by the superposition of all its eigenstates ($|1\rangle$ and $|0\rangle$ in the case of a single qubit).

This is the reason why multi-qubit systems are particularly powerful. Since with n qubits it is possible create a superposition of 2^n states, it is substantially possible to handle simultaneously all the possible states and to execute an operation on multiple data, each one associated with a state, in parallel. This concept is sometimes expressed as "quantum parallelism".

Entanglement is a type of correlation that has no correspondence in the classical world. When two quantum systems are *entangled*, if one of them collapses to a particular state after a measurement, the state of other will instantaneously collapse too.

A pair of entangled qubits is represented by the state vector in Equation (2.10).

$$|\psi\rangle = \begin{array}{c} \mathbf{00} \\ \mathbf{01} \\ \mathbf{10} \\ \mathbf{11} \end{array} \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix}, \tag{2.10}$$

where the probabilities of the couple of being $|00\rangle$ or $|11\rangle$ are $1/2$, while those of $|01\rangle$ or $|10\rangle$ are 0. In this case, when a qubit is measured, the other will collapse to the same value.

Another important property of quantum mechanics is the **no cloning principle**. It asserts that a quantum state cannot be copied without destroying it [13], in the sense that it is not possible to have two independent copies of the same state. This is a limitation of quantum computing: it is not possible to read a state, manipulate it and then reuse directly the original state to perform others manipulation of it.

2.3 Quantum gates

As classical computation relies on a set of logic gates, qubit states can be changed by applying quantum gates. In the single qubit case, they act on θ and ϕ in Equation (2.3), or, alternatively, on c_0 and c_1 in Equation (2.2).

In general, a quantum gate can be modelled as a unitary square matrix U modifying a quantum state vector $|\psi_0\rangle$ according to the following expression:

$$|\psi_1\rangle = U |\psi_0\rangle . \tag{2.11}$$

Due to the mathematical properties of unitary matrices, it is possible to define the inverse of a quantum gate as its Hermitian conjugate, which is simply obtainable by taking the complex conjugate of the transpose matrix. If U is a gate, its inverse is denoted as $U^\dagger$, so:

$$|\psi\rangle = U^\dagger U |\psi\rangle . \tag{2.12}$$

Due to their intrinsic linearity, given two gates, U_0 and U_1, a third gate, U_f, is equivalent to the combination of them if it is equal to the multiplication of them:

$$U_0 U_1 |\psi\rangle = U_f |\psi\rangle \qquad \text{if } U_f = U_0 U_1. \tag{2.13}$$

Another property of quantum gates derives from Equation (2.13): applying the same gate more than once is equivalent to raising it to power:

$$U_0 U_1 \dots U_{n-1} |\psi\rangle = U^n |\psi\rangle . \tag{2.14}$$

Here some gates employed in the following of this thesis are presented. A more comprehensive discussion on this can be found in [14].

2.3.1 Single qubit gates

Hadamard gate

One of the most basic gates is the Hadamard gate. Given $|\psi\rangle$ the state of a qubit:

$$H|\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \frac{c_0+c_1}{\sqrt{2}} \\ \frac{c_0-c_1}{\sqrt{2}} \end{bmatrix} \tag{2.15}$$

This is particularly useful when superposition is needed, starting from a basis state of $|0\rangle$ or $|1\rangle$. In fact, after the application of this gate, the probability of measuring one of the two basis states is equal, since it transforms $|0\rangle$ and $|1\rangle$ in $|+\rangle$ and $|-\rangle$ respectively.

X gate

The X gate operates a reflection of the state along the x axis of the Bloch sphere.

$$X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} \tag{2.16}$$

As one can notice doing calculations, this gate swaps the amplitudes of the state.

Pairing with X gate there are the Y and Z gates, they have the same effect as X gate, but reflection is done around the other two axes. Their matrices and applications are:

$$Y|\psi\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} -ic_1 \\ ic_0 \end{bmatrix} \tag{2.17}$$

$$Z|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0 \\ -c_1 \end{bmatrix} \tag{2.18}$$

Rotation gates

Rotation gates are a generalization of the three aforementioned gates. In fact, they can only operate reflections (rotations of π radians), while rotation gates are parametric with respect to the rotation angle around the chosen axis. They are $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$, the subscript indicates the axis around which the rotation is performed. Their applications are reported:

$$R_y(\theta)|\psi\rangle = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0\cos\left(\frac{\theta}{2}\right) - c_1\sin\left(\frac{\theta}{2}\right) \\ c_0\sin\left(\frac{\theta}{2}\right) + c_1\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.19}$$

$$R_x(\theta)|\psi\rangle = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0\cos\left(\frac{\theta}{2}\right) - ic_1\sin\left(\frac{\theta}{2}\right) \\ -ic_0\sin\left(\frac{\theta}{2}\right) + c_1\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.20}$$

$$R_z(\theta)|\psi\rangle = \begin{bmatrix} \mathrm{e}^{-i\frac{\theta}{2}} & 0 \\ 0 & \mathrm{e}^{i\frac{\theta}{2}} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0\mathrm{e}^{-i\frac{\theta}{2}} \\ c_1\mathrm{e}^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.21}$$

S gate

The S gate, also known as $\sqrt{\mathrm{Z}}$ gate, applies a rotation of $\pi/2$ around the Z axis.

$$S|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0 \\ ic_1 \end{bmatrix} \tag{2.22}$$

The name derives from the fact that a sequence of two S gates has the same effect as a single Z gate.

T gate

The T gate applies a rotation of $\pi/4$ around the Z axis, similarly to S gate, it is called $\sqrt[4]{\mathrm{Z}}$. Its application is reported here:

$$T|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1\mathrm{e}^{i\frac{\pi}{4}} \end{bmatrix} \tag{2.23}$$

2.3.2 Multi-qubit gates

Single qubit gates are useful, but in a quantum algorithm it is important to have the possibility to build a relation between qubits and obtain a result given by the interaction of these. State-of-the-art technology permits to properly implement only two-qubit gates.

In a two-qubit gate, one of the two qubits usually controls the application of a gate to the second one.

Given a unitary operation, it is always possible to get a controlled version of it, as reported in [11]. In particular, if

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{2.24}$$

the controlled version of U is:

$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \tag{2.25}$$

The application of CU to a two-qubit system is reported in Figure 2.3.

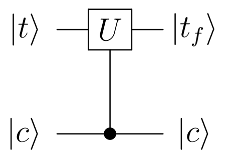


Figure 2.3: Contrelled-U gate

The control qubit ($|c\rangle$) will be unaffected, while the target qubit ($|t\rangle$) will be modified according to the value of the control qubit.

$$|t_f\rangle = |t\rangle \qquad \text{if } |c\rangle = |0\rangle \tag{2.26}$$

$$|t_f\rangle = U\,|t\rangle \qquad \text{if } |c\rangle = |1\rangle \tag{2.27}$$

It is possible to control a gate in the complementary way, this is called anti-control and is reported in Figure 2.4.

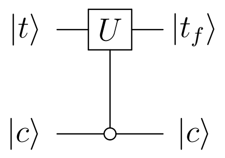
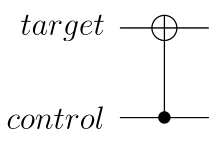


Figure 2.4: Anticontrolled-U gate

In this case U is applied if only if $|c\rangle$ is in the state $|0\rangle$.

CNOT gate

The controlled-NOT (CNOT) gate is a controlled variant of the X gate. In particular, the inversion is applied to the target qubit only if the control qubit is in state $|1\rangle$, otherwise it is not affected.



Since the gate is applied on two qubits, the vector, which describe the affected quantum system, is composed of four elements, which are, respectively, the probability amplitudes of the four basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. Denoting the state of the quantum system as $|\psi\rangle$:

$$CNOT\,|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_3 \\ c_2 \end{bmatrix}. \tag{2.28}$$

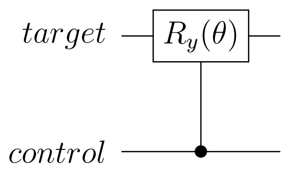
In order to better understand the gate effect, it is possible to consider a spacial case in which only one of the probability amplitudes is equal to 1, while others are 0:

$$CNOT\,|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{2.29}$$

in this case, where only $c_2 = 1$, the control qubit is in the state $|1\rangle$, while the target qubit is in the state $|0\rangle$, the output state vector, after the application of the CNOT matrix, represents a state in which the two qubits are both in the state $|1\rangle$, this means that the control qubit remains untouched, while the target qubit is flipped.

Controlled rotation gates

Moreover, it is possible to perform a controlled version of rotation gates, which are applied on the target qubit if and only if the control qubit is in the state $|1\rangle$.

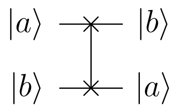


The application of this gate to two qubits is reported here:

$$CR_y(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2\cos\left(\frac{\theta}{2}\right) - c_3\sin\left(\frac{\theta}{2}\right) \\ c_2\sin\left(\frac{\theta}{2}\right) + c_3\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \quad (2.30)$$

Swap gate

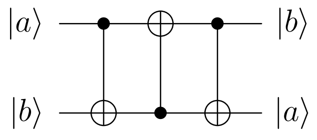
This gate simply logically swaps two qubits.



The application of the swap gate to the state vector of a two-qubit system is:

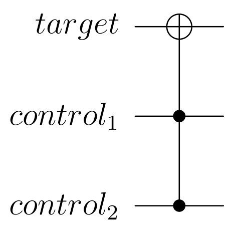
$$SWAP\,|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ c_1 \\ c_3 \end{bmatrix}. \quad (2.31)$$

This gate can be implemented by the following series of CNOT's.



Toffoli gate

Toffoli gate is an example of three-qubit gate. In particular, it acts as a *CNOT* with two control qubits, indeed it is also known as *CCNOT*, and it implements the exclusive-OR (XOR) operation between the target qubit and the result of a logical AND between the two control qubits.



2.4 Algorithms

From a high abstraction level perspective, a quantum algorithm acts on the states probability distribution and maximizes the probability of the states representing the solution.

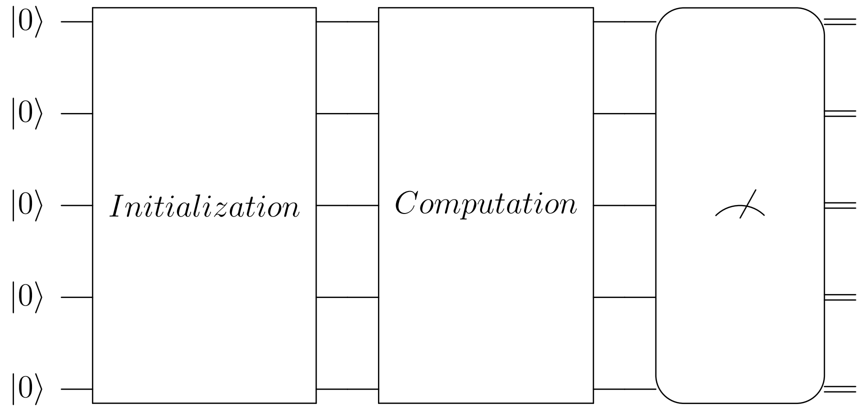


Figure 2.5: Example circuit topology

Figure 2.5 represents the flow of a quantum algorithm. Usually, the initial qubits state is the $|00\dots00\rangle$ state. Afterwards a preparation phase is required in which the state is modified by the initialization circuit (Figure 2.5) to model the problem, finally the solution is computed by the computation circuit (Figure 2.5) and the final state is measured obtaining the result.

Here, a review of some important algorithms is presented.

2.4.1 Grover's algorithm

Given a database of N unordered items, it is possible to identify one or more items exploiting a search algorithm. It is necessary to examine every single item and check if it corresponds to the desired one, this requires $O(N)$ tests.

Grover proposed in [5] a quantum algorithm to perform this task by only $O(\sqrt{N})$ queries, thus introducing a quadratic speed-up in time needed to find the desired item.

Grover algorithm is composed of a series of quantum operations with the aim of amplifying the probability amplitude of the states associated with the items to be searched.

Assuming that a quantum register of n qubits is initialized to the state $|0\rangle^{\otimes n}$, a uniform superposition of all the states is obtained applying Hadamard gates, as shown in Equation (2.32).

$$|\psi\rangle = H^{\otimes n} |0\rangle^{\otimes n} . \tag{2.32}$$

In this way, every amplitude of the states in the computational basis will be $1/\sqrt{2^n}$, as shown in Figure 2.6, which reports the amplitude of the states of a two-qubits system taken as example. The amplitudes are equal to $1/\sqrt{4}$, while their mean (the dashed line in Figure 2.6) will be $1/\sqrt{4}$.

Then the phase of the amplitude of the target state ($|01\rangle$ in the considered example) is inverted, this can be done by the application of a unitary operator O, called **oracle**:

$$|\psi'\rangle = O |\psi\rangle . \tag{2.33}$$

Figure 2.7 shows that the target state ($|01\rangle$) has a negative amplitude and the average amplitude has a lower value.

Afterwards, the **diffusion operator** (D), defined as

$$D = 2 |\psi\rangle \langle\psi| - \mathbb{1}, \tag{2.34}$$

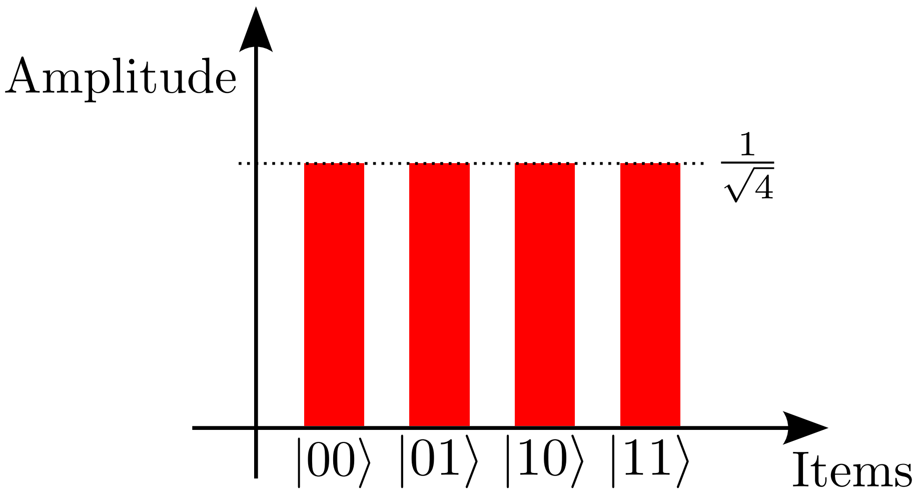


Figure 2.6: Uniform superposition

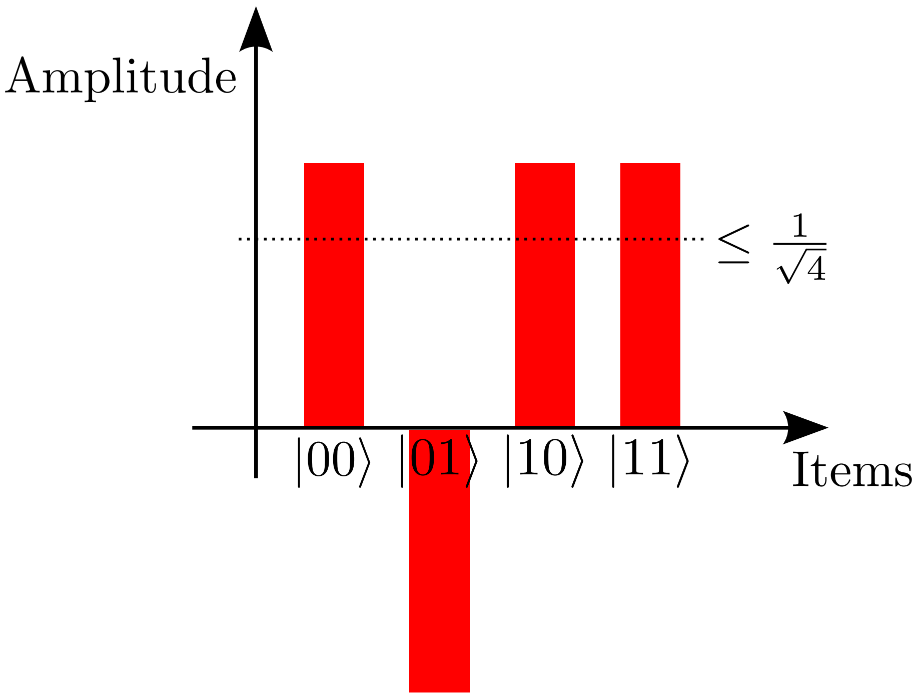


Figure 2.7: Target state inversion

where $|\psi\rangle$ is the state of Equation (2.32) and $\mathbb{1}$ is the identical matrix, is applied. This has the effect of inverting the probability amplitudes around their average, and amplifying the amplitude of the target item.

Figure 2.8 reports the resulting amplitude graph for the considered example, it is possible to notice that the amplitude of the target state ($|01\rangle$) is amplified.

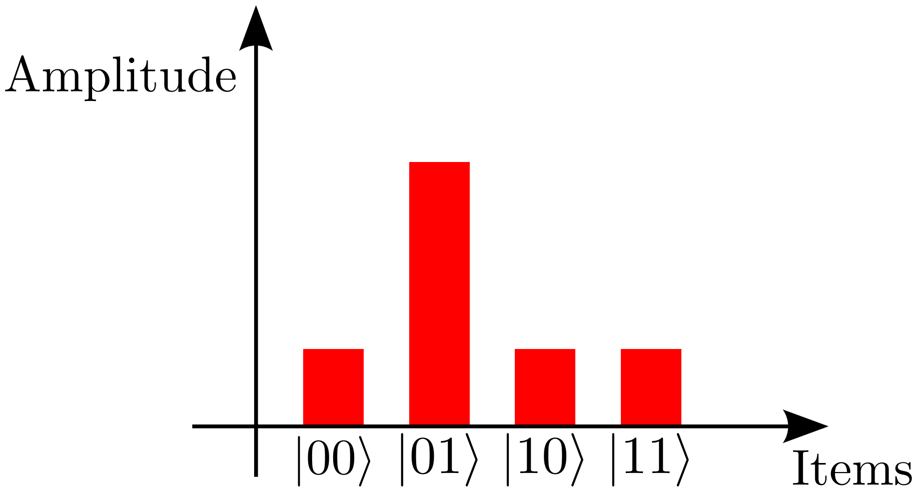


Figure 2.8: Inversion around the average

To further amplify the amplitude of the target state, the oracle and the diffusion operators have to be iterated $\sqrt{N}$ times.

It is possible to define as **Grover operator** G, the combination of the oracle and the diffusion operator. In a more compact form, the resulting state $|\psi_f\rangle$ is:

$$|\psi_f\rangle = (DO)^r |\psi\rangle = (G)^r |\psi\rangle \,, \tag{2.35}$$

where, r is the number of applications of the Grover operator.
The circuit that implements this algorithm is reported in Figure 2.9.

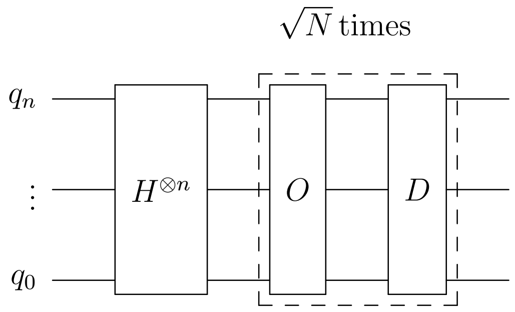


Figure 2.9: Grover algorithm

Oracle

The oracle, as presented in [12], can be seen as a unitary operator that applies a piecewise-defined function, that is 1 if the tested item is the target one, 0 otherwise:

$$f(x) = \begin{cases} 1 & \text{if } x = w \\ 0 & \text{otherwise} \end{cases} \tag{2.36}$$

where x is the tested item and w is the target one.

It is possible to define the oracle O by its action on an item as:

$$|x\rangle\,|q\rangle \xrightarrow{O} |x\rangle\,|q \oplus f(x)\rangle\,, \tag{2.37}$$

where $|x\rangle$ is the quantum register of n qubits that contains the tested item, $|q\rangle$ is an ancillary single qubit, while $\oplus$ is the XOR operation.
Thanks to the XOR operation, if $|x\rangle$ corresponds to the target item ($f(x) = 1$) the state of $|q\rangle$ is changed, if it is in the state $|0\rangle$ it is flipped to $|1\rangle$.

If $|q\rangle$ is in the state $|-\rangle$, starting from $|0\rangle$ this can be obtained applying an H and a Z gate, Equation (2.37) becomes:

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{O} |x\rangle\,(-1)^{f(x)} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), \tag{2.38}$$

if $|x\rangle$ is not the target nothing happens to $|q\rangle$ and its state remains $|-\rangle$, otherwise, since $|0\rangle$ and $|1\rangle$ are flipped, the state of $|q\rangle$ becomes $-\,|-\rangle$.

The oracle effect is substantially to flip the sign of the probability amplitude of the target states.

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}\,|x\rangle\,, \tag{2.39}$$

where $|q\rangle$ has been omitted for practice.

A general methodology for designing a Grover oracle, based on Boolean algebra and Z gate for basis-to-amplitude translation of information, is available in [15].

Diffusion operator

Equation (2.34) reports the definition of the diffusion operator, this can be better understood recalling that

$$|\psi\rangle = H^{\otimes n} |0\rangle^{\otimes n}, \tag{2.40}$$

and

$$\langle\psi| = \langle 0|^{\otimes n} H^{\otimes n}, \tag{2.41}$$

i.e., the conjugate transpose of $|\psi\rangle$, according to Dirac notation.

So,

$$D = H^{\otimes n} 2 \left(|0\rangle^{\otimes n} \langle 0|^{\otimes n} - \mathbb{1} \right) H^{\otimes n}. \tag{2.42}$$

This operator is implemented by the quantum circuit in Figure 2.10.

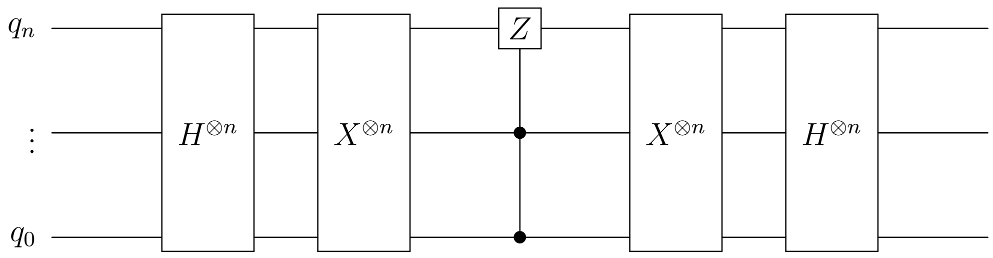


Figure 2.10: Diffusion operator

2.4.2 Quantum Fourier Transform

Given a vector of complex numbers, $[x_0, x_1, \ldots, x_N]$, the discrete Fourier transform associates to it a second vector of complex numbers, $[y_0, y_1, \ldots, y_N]$, such that:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}, \tag{2.43}$$

where $i = \sqrt{-1}$. This is one of the most useful transformations of signal processing.

Its quantum computing counterpart is the quantum Fourier transform (QFT).

The quantum Fourier transform applies the same transformation, but it acts as a linear operator that given a state $|\phi\rangle$ produces a state $|\psi\rangle$ such that the amplitudes of $|\psi\rangle$ are the discrete Fourier transform of the amplitudes of $|\phi\rangle$.

A unitary matrix and a relative operator can be defined by its $j-k^{th}$ element:

$$QFT_{jk} = e^{2\pi ijk/N}. \tag{2.44}$$

This operation can be imagined as a basis exchange, in fact the QFT permits to pass from an amplitude encoding of a number to its phase encoding. This means that the basis will not be expressed as a combination of $|0\rangle$ and $|1\rangle$, but as a combination of $|+\rangle$ and $|-\rangle$.

An Inverse Quantum Fourier Transform can be also defined taking the Hermitian conjugate of the QFT matrix. It will have the effect to go back to the computational basis from the phase encoding.

2.4.3 Quantum Phase Estimation

The aim of the quantum phase estimation (QPE) algorithm is to find the phase of a unitary operator U, i.e. θ in $U|u\rangle = e^{2\pi i\theta}|u\rangle$, where $|u\rangle$ and $e^{2\pi i\theta}$ are an eigenvector-eigenvalue pair of U.

To perform this algorithm, two quantum registers are required (Equation (2.45)). The first one contains the eigenvector $|u\rangle$ and the second one contains n $|0\rangle$, where n depends on the accuracy required to estimate θ:

$$\begin{aligned} |\psi\rangle &= |u\rangle \\ |z\rangle &= |0\rangle^{\otimes n}. \end{aligned} \tag{2.45}$$

A set of n Hadamard gates is applied to $|z\rangle$, so the resulting state will be:

$$|r\rangle = H^{\otimes n}|z\rangle|\psi\rangle = \frac{1}{2^{n/2}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle \tag{2.46}$$

Then n controlled version of U, raised to successive powers of two, are applied to $|\psi\rangle$, these are controlled by every qubit of $|z\rangle$ as shown in Figure 2.11.

At this point it is possible to calculate the state of $|z\rangle$ after the application of the set of U:

$$|f\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i\theta k} |k\rangle , \tag{2.47}$$

where k is the integer value of the corresponding binary string.
Equation (2.47) can be obtained remembering that $e^{2\pi i\theta}$ is an eigenvalue of U.

In the end, to pass from the phase encoding of θ to its computational basis encoding, the inverse QFT is applied to $|f\rangle$. A measurement of the state in the computational basis will give the binary encoding of θ expressed as a fixed point number.

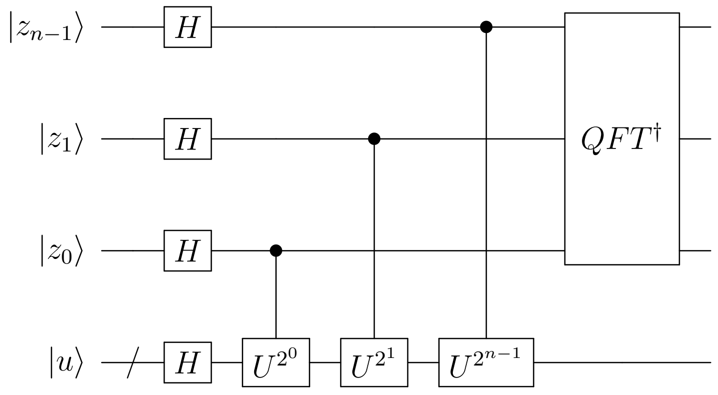


Figure 2.11: Quantum phase estimation circuit

2.4.4 Quantum Function encoding

To develop effective computation, it is necessary to modify a quantum state in such a way that it represents a problem, many times, this is expressed as a function in terms of input/output or domain/image couples.

Quantum dictionaries, presented in [16], are introduced as the quantum counterpart to the software ones.

This is built by entangling two quantum registers, $|k\rangle$ and $|v\rangle$, to represent respectively key/value couples, like in its software version. The key/value pair can also describe a domain/image couple like a lookup table does.

The idea behind quantum dictionaries is to encode in the phase of an ancillary qubit the desired value, doing something similar to Quantum Phase Estimation, and reconstructing the value in computational basis with the inverse quantum Fourier transform.

Values are mapped onto multiples of a base angle. Given m qubits of $|v\rangle$ it is possible to represent 2^m integer values as multiples of $2\pi/M$ where $M = 2^m$, Figure 2.12 reports a visualization of it.

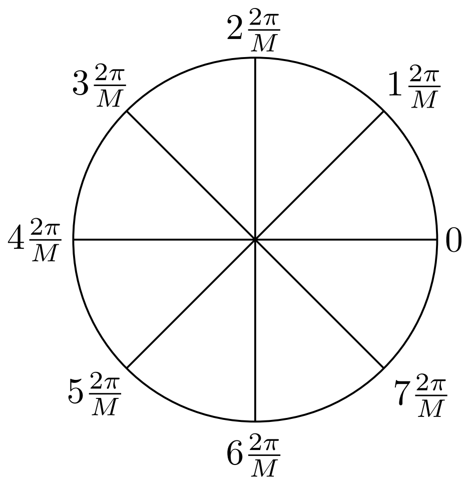


Figure 2.12: Angular representation of 8 positive numbers

Negative multiples of the base angle (clockwise rotations) can be used to represent negative numbers in two's complement notation.

Different types of representations can be adopted:

- Complete: Every key of the dictionary is explicitly mapped to a value.

- Partial: Just a set of keys is mapped to values, others are computed "automatically".

And different definitions are possible:

- Classical: key/value pairs are precomputed by a classical system.
- Quantum: key/value pairs are computed by the same quantum system that performs the encoding of the dictionary.

Independently from the selected representation/definition chosen, the key $|k\rangle$ and the value $|v\rangle$ registers must be put in equal superposition by means of a set of Hadamard gates, while the ancilla qubit $|a\rangle$ stores an eigenvector of the operator that encodes values in the phase, like in the Quantum Phase Estimation algorithm. This is implemented by the circuit in Figure 2.13.

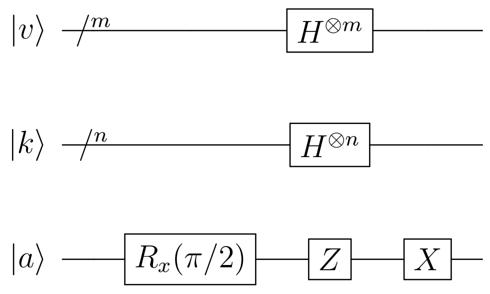


Figure 2.13: Quantum dictionary initialization circuit

Then the encoding operator is applied, performing controlled rotations of $|a\rangle$, depending on the selected representation/definition.

In the end, the inverse quantum Fourier transform is applied to $|v\rangle$ and the initialization of $|a\rangle$ is reversed to simplify measurements. These operations are definition-independent, the quantum circuit for this is reported in Figure 2.14.

Classically defined function

This method will be analysed by following an example.

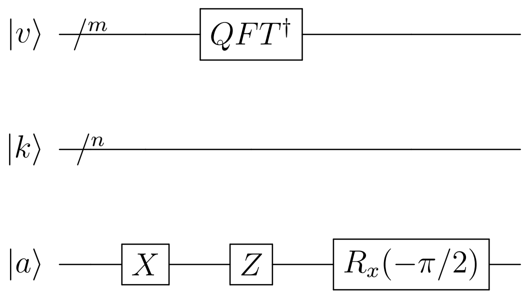


Figure 2.14: Quantum dictionary termination circuit

Given a complete, classically defined function $f(x)$:

$$f(x) = \begin{cases} 3 & \text{if } x = 0 \\ 5 & \text{if } x = 1 \\ 2 & \text{if } x = 2 \\ 1 & \text{if } x = 3 \end{cases} . \tag{2.48}$$

Adopting a purely binary representation, 2 qubits are required to represent input the domain, while 3 qubits are sufficient to represent the image. The binary lookup table of this function is the one reported in Table 2.1.

key	value
00	011
01	101
10	010
11	001

Table 2.1: Example's LUT

Every qubit in the two quantum registers corresponds to a bit of the key/value pairs.

To effectively encode the dictionary, a sequence of controlled rotations is applied to the ancilla. Circuits in Figure 2.15 and Figure 2.16 report the encoding of the key/value pairs in the lookup table. To every key/value pair coincides a set of controlled rotations, this contains a rotation for every qubit of $|v\rangle$. A qubit of $|v\rangle$

and qubits of $|k\rangle$ corresponding to bits of the key that are 1 control every rotation in the aforementioned set, while qubits corresponding to bits of the key that are 0 anti-control them. The angle of rotation of the $i-th$ rotation gate in the set is:

$$\theta = V\frac{2\pi}{2^n}2^{i+1}, \tag{2.49}$$

where: V is the value to be encoded, n is the number of bits in the $|v\rangle$ register and i is the index of the controlling qubit of the $|v\rangle$ register. Then the circuit in

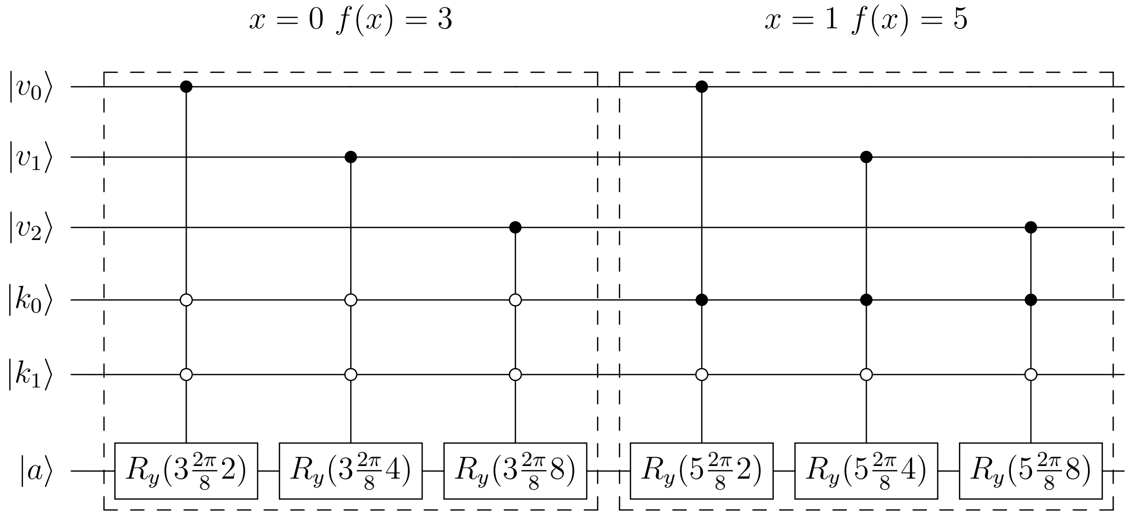


Figure 2.15: Circuit to encode the first and the second values

Figure 2.14 is applied to allow the measurement in computational basis.

After the application of the total dictionary circuit, the state encodes the function in Equation (2.48) and the final state is:

$$|\psi\rangle = \frac{|00\rangle_k |011\rangle_v + |01\rangle_k |101\rangle_v + |10\rangle_k |010\rangle_v + |11\rangle_k |001\rangle_v}{\sqrt{4}}. \tag{2.50}$$

Applying repeated simulations and measurements it is possible to plot the probability distribution of the state over the computational basis states.

Figure 2.17 reports the probability distribution of states with a probability different from 0 obtained after the application of the example dictionary. Keys are

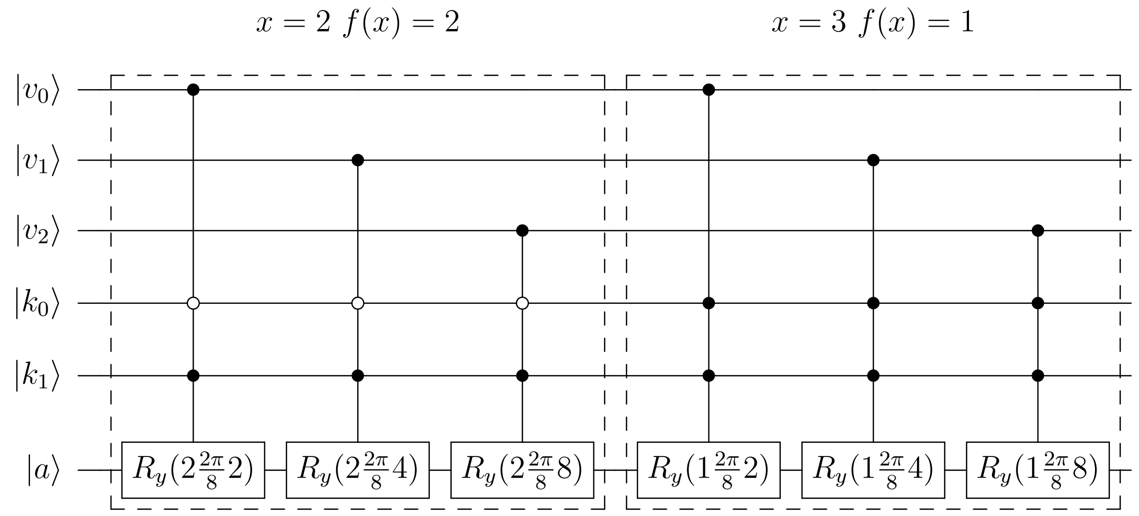


Figure 2.16: Circuit to encode the third and the fourth values

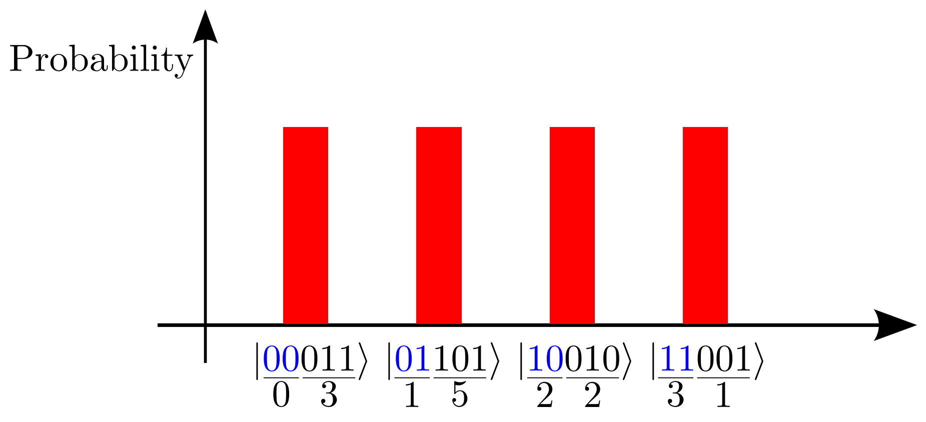


Figure 2.17: Probability distribution of the final state

highlighted in blue, values are black and the integer value of the bit string is reported. As it is possible to notice, the obtained distribution is uniform over the states of $|k\rangle\,|v\rangle$ that represents key/value pairs with a value of $1/2^n$.

Quantum defined function

It is also possible to encode directly a function in a quantum state computing its values directly on quantum hardware.

Let $f(x)$ be a function of n binary variables that maps $\mathbb{B}^n$ to a subset of $\mathbb{Z}$:

$$f(x) = 3x_0 - 2x_0x_1 + 2x_1, \tag{2.51}$$

where x_0 and x_1 are binary variables.

Here the function is not computed explicitly: every term of the function has its set of rotations, not every domain/image couple. $|k\rangle$ contains two qubits, one per binary variable, and $|v\rangle$ contains three qubits because a two's complement is adopted.

Rotations are applied with the same strategy as the classically defined case, but the control qubits are the one corresponding to the binary variables involved in each term and θ (the angle of every rotation) is the same as Equation (2.49), but V is the coefficient of the term. Circuits in Figure 2.18 and Figure 2.19 report the encoding of the three terms of the function.

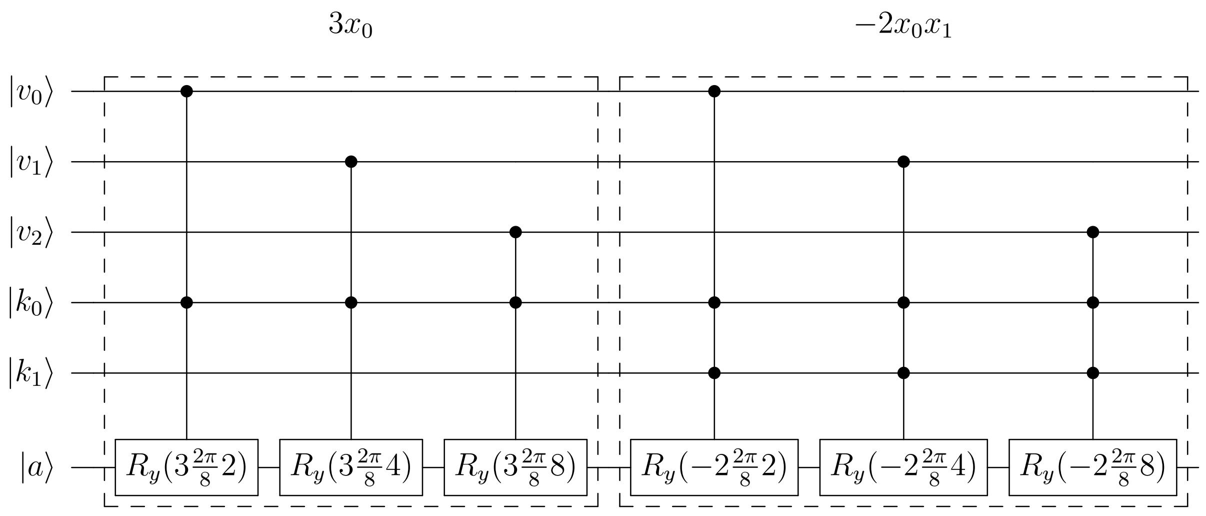


Figure 2.18: Circuit to encode the first and the second terms

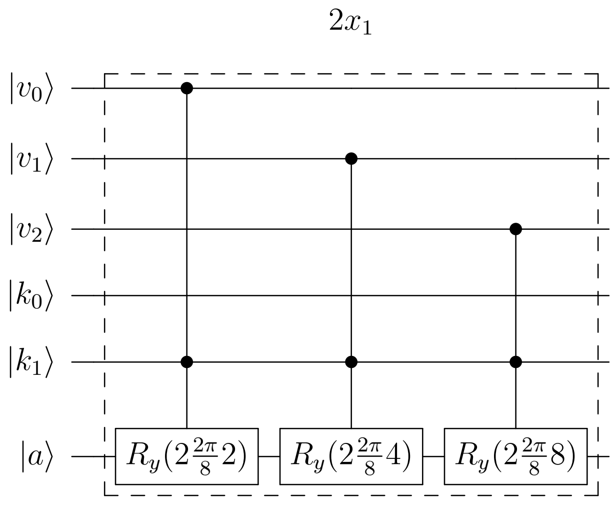


Figure 2.19: Circuit to encode the third term

How it is possible to notice, coefficients of the function are encoded, not its values. This permits to obtain a shorter circuit, since the classical description of the function would need 2^m, where m is the number of qubits in the key register, sets of rotations.

The state after the application of the dictionary will be something like:

$$|\psi\rangle = \begin{array}{c} \mathbf{00000} \\ \mathbf{00001} \\ \vdots \\ \mathbf{01011} \\ \vdots \\ \mathbf{10010} \\ \vdots \\ \mathbf{11011} \\ \vdots \\ \mathbf{11111} \end{array} \begin{bmatrix} 1/\sqrt{2^5} \\ 0 \\ \vdots \\ 1/\sqrt{2^5} \\ \vdots \\ 1/\sqrt{2^5} \\ \vdots \\ 1/\sqrt{2^5} \\ \vdots \\ 0 \end{bmatrix} \tag{2.52}$$

Figure 2.20 reports a plot of the probability distribution of the state, i.e. the P vector in Equation (2.52). In this case the distribution is uniform on the computational basis states obtained by the concatenation of the binary representations of key/value pairs, while it is null on the others. The advantage of this technique is that the

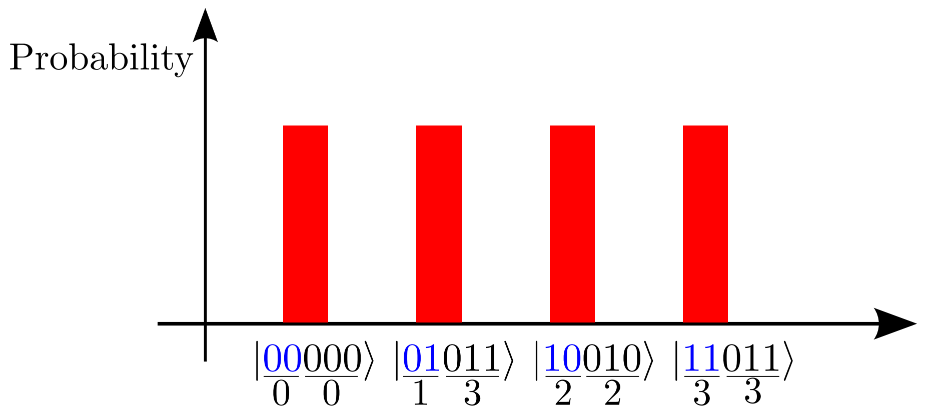


Figure 2.20: Probability distribution of the final state associated with the binary function in Equation (2.51)

function is not computed a priori and that the circuit "automatically" computes each value of the function exploiting its terms, indeed, the probability distribution in Figure 2.20 can be used to write the lookup table of this function (Table 2.2).

key	value
00	000
01	011
10	010
11	011

Table 2.2: LUT of Equation (2.51)

Replacing keys in Equation (2.51) it is possible to check that the obtained encoding is correct.

2.4.5 Grover's algorithm extension

Grover's algorithm, as presented in Section 2.4.1, starts with the production of a uniform superposition of all the computational basis states, obtained applying a set of Hadamard gates, but this can be replaced by any other state preparation circuit producing a uniform distribution over the states that are relevant to the problem.

If P is a state preparation operator that encodes in the state a database, Equation (2.32) becomes:

$$|\psi\rangle = P\,|0\rangle^n \,, \tag{2.53}$$

then the oracle O is applied as reported in Equation (2.33).
The diffusion operator, defined in Equation (2.34), becomes:

$$D = P2(|0\rangle^n \langle 0|^n - \mathbb{1})P^\dagger \tag{2.54}$$

This is the same as Equation (2.42) where $P = H^{\otimes n}$, since the Hermitian conjugate of the Hadamard gate is the Hadamard gate itself.

In a practical scenario, quantum dictionary can help in defining a P operation for a generic problem to be solved with Grover's search algorithm. In the following chapters, a dictionary-based Grover's search will be employed as a subroutine for solving binary optimization problems.

2.4.6 More on Grover algorithm

Section 2.4.1 presents Grover's algorithm in the case that just an item in a database is the target to be found, iterating the Grover operator $\sqrt{N}$ times, it is possible to obtain a success percentage of almost 100%. In the context of Grover Adaptive Search it is used to find a negative value of the cost function, so in this case the Grover algorithm produces samples of the search domain following the probability distribution proposed in [17].

In general, it is possible to model this problem as the search of an item in an unordered database that contains more than an item that corresponds to the search criterion.

Let S be the searching domain, containing N unordered items, and M the subset of the searching domain that contains all the t point corresponding to the search criterion (labelled points). Let $p = t/N$ be the portion of labelled points.

Applying r times the Grover operator defined in Equation (2.35), the final measurement outputs a value in S with the following probability distribution:

$$\gamma(\{x\}) = \begin{cases} \frac{g_r(p)}{t} & \text{if } x \in M \\ \frac{1-g_r(p)}{N-t} & \text{if } x \in S \setminus M \end{cases}, \tag{2.55}$$

where

$$g_r(p) = \sin^2\left[(2r+1)\arcsin\sqrt{p}\right]. \tag{2.56}$$

The parameter r is a degree of freedom for the algorithm and it must be chosen to maximize g_r. Unfortunately, choosing the optimal value for it is not easy, because it depends from the problem characteristic.

To better understand Equation (2.55), a quantitative example is proposed:

$$N = 100, \tag{2.57}$$

$$t = 30, \tag{2.58}$$

$$p = \frac{3}{10}, \tag{2.59}$$

$$g_r(p) = \sin^2\left[(2r+1)\arcsin\sqrt{\frac{3}{10}}\right]. \tag{2.60}$$

As it can be noticed, $g_r(p)$ is parametric with respect to r. Figure 2.21 reports the plot of it for different values of r, which is an integer number, so an interpolation with real values is added for representation reasons.

From Figure 2.21, it is possible to notice that the maximum of g_r is obtained for $r = 9$.

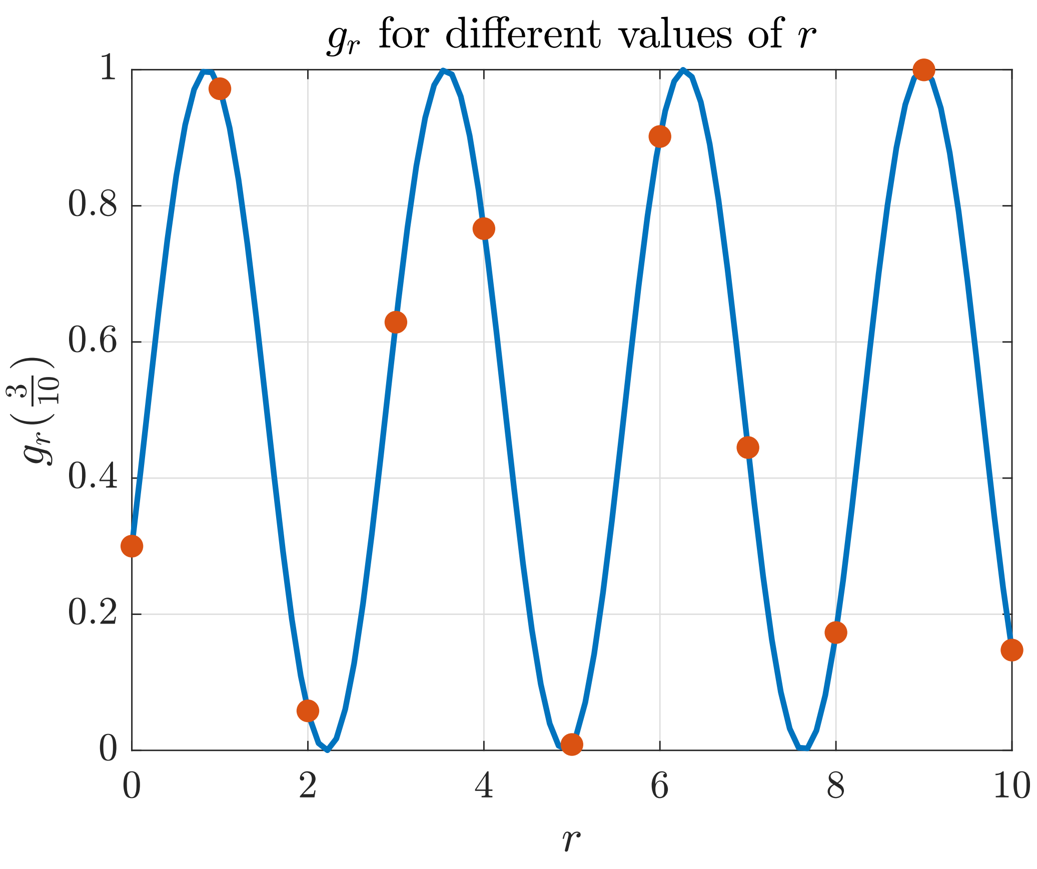


Figure 2.21: Plot of g_r

With this choice:

$$g_r(p) = \sin^2\left[(18+1)\arcsin\sqrt{\frac{3}{10}}\right] = 0.9997, \tag{2.61}$$

$$\gamma(\{x\}) = \begin{cases} \frac{0.9997}{30} & \text{if } x \in M \\ \frac{1-0.9997}{100-30} & \text{if } x \in S \setminus M \end{cases} = \tag{2.62}$$

$$= \begin{cases} 0.0333 & \text{if } x \in M \\ 4.2857 \times 10^{-6} & \text{if } x \in S \setminus M \end{cases} .$$

It is interesting to consider three particular cases:

- $p = 0$: in this case there are no marked items, $M = \emptyset$, and $\gamma(\{x\})$ is uniform over the whole search domain, therefore every point of the domain is equally probable, but there are no correct results.

- $p = 0.5$: in this case $t = N/2$ and $g_r(p) = 0.5$ independently of the chosen r, so the probability is uniform over the total searching domain, so each $\gamma(\{x\})$ is $1/N$, this means that every item in the search domain is equally probable and the probability of obtaining a correct result is $1/2$, that is p.

- $p = 1$: in this case all the search domain is marked, $M = S$ and $S \setminus M = \emptyset$, while $g_r(p) = 1$ for every choice of r. This means that the output probability distribution is uniform over all the search domain, so each value has the same probability of being measured and every result is correct.

Chapter 3

Optimization problems formulation and solving methodologies

This chapter presents base concepts of combinatorial optimization with a particular focus on **PCBO** and **QUBO** formulations since quantum computation particularly can solve these. Moreover, an overview on techniques to solve this class of problems, in particular classical and quantum techniques, are presented.

This thesis focuses on Grover Adaptive Search, which is a hybrid algorithm explained in detail in Chapter 4, and its application to QUBO formulations for machine learning, presented in Section 3.3. In particular, the use case of linear regression was studied.

An optimization problem is expressed as:

$$\text{find} \quad x \in S \mid C(x) = \min \{C(x)\}\,, \tag{3.1}$$

where $C(x) : S \to \mathbb{R}$ is the cost function of the optimization problem, S is the solution space and x is the solution which minimizes $C(x)$.

If S in Equation (3.1) is a finite and discrete set, the optimization problem is defined **combinatorial** [18].

3.1 PCBO model

PCBO [19] is one of the possible mathematical formulations of combinatorial optimization problems. PCBO is an acronym that stands for **Polynomial Constrained Binary Optimization** and this model encodes the problem in a polynomial cost function. The solution is a combination of binary variables and the solution space is $\mathbb{B}^n$ where n is the number of binary variables involved in the cost function. This model includes the possibility of representing constrained optimization problems. Since the cost function is a real-valued polynomial function of binary variables, the name is justified.

3.2 QUBO model

Many combinatorial optimization problems can be modelled with **QUBO** formulation [20]. QUBO is an acronym that stands for **Quadratic Unconstrained Binary Optimization**. In general, this model has the following formulation:

$$\text{minimize/maximize} \quad f(x) = x^T Q x, \tag{3.2}$$

where $x \in \mathbb{B}^n$ is a column vector of n binary variables and $Q \in \mathbb{R}^{n \times n}$ is a square matrix of constants. Without losing in generality, the coefficient matrix Q can be expressed in **symmetric** or **upper triangular** form. In the case of an upper triangular Q matrix, the cost function $f(x)$ in Equation (3.2) becomes:

$$f(x) = \sum_{i>j}^{n} x_i x_j Q_{ij} + \sum_{i}^{n} Q_{ii} x_i, \tag{3.3}$$

the cost function written in this form reveals its quadratic nature. In fact, it is composed by a quadratic part where the elements out of the diagonal of Q are multiplied by couples of elements of x and a linear part, where the elements in the diagonal of Q are multiplied by the elements of x. The linear part comes from the fact that, due to the binary nature of the variables, $x_i x_i = x_i^2 = x_i$.

Given a Q matrix, it is possible to obtain its upper triangular form and the symmetric one. In particular:

- To obtain a symmetric form: substitute every q_{ij} except $i = j$ in Q with $(q_{ij} + q_{ji})/2$.
- To obtain an upper triangular form: substitute every q_{ij} for $i > j$ with $(q_{ij} + q_{ji})$ and 0 for $i < j$, do not touch elements with $i = j$.

An example follows:

$$Q = \begin{bmatrix} 8 & 5 & 8 & 0 \\ 3 & 1 & 8 & 0 \\ 0 & 6 & 2 & 7 \\ 0 & 2 & 7 & 4 \end{bmatrix} \Longleftrightarrow \begin{bmatrix} 8 & 4 & 4 & 0 \\ 4 & 1 & 6 & 1 \\ 4 & 6 & 2 & 7 \\ 0 & 1 & 7 & 4 \end{bmatrix} \Longleftrightarrow \begin{bmatrix} 8 & 8 & 8 & 0 \\ 0 & 1 & 12 & 2 \\ 0 & 0 & 2 & 14 \\ 0 & 0 & 0 & 4 \end{bmatrix}. \tag{3.4}$$

The cost function associated with the considered example is:

$$f(x) = x^t Q x = \begin{bmatrix} x_0 \, x_1 \, x_2 \, x_3 \end{bmatrix} \begin{bmatrix} 8 & 8 & 8 & 0 \\ 0 & 1 & 12 & 2 \\ 0 & 0 & 2 & 14 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \tag{3.5}$$
$$= 8x_0x_1 + 8x_0x_2 + 12x_1x_2 + 14x_2x_3 + 8x_0 + 1x_1 + 2x_2 + 4x_3,$$

where the x vector of Equation (3.2) is composed by four binary variables.

It is possible to notice that there are no explicit constraints applied to variables. Indeed, QUBO is a model for unconstrained optimization problems. However, some constraints can be written as quadratic penalities and, in this way, they can be added to the cost function. Table 3.1 reports some quadratic penalty associated with some important constraints.

Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

Table 3.1: Some example constraint/penalty pairs

All variables in Table 3.1 are to be intended as binary variables, while P, that is a positive penalty parameter, is sufficiently high to avoid constraints violations.

QUBO models can be used to describe different optimization problems, an example is reported in following.

3.2.1 The Max-Cut Problem

The **Max Cut** problem [20] is a combinatorial optimization problem, whose objective is to divide the vertices of a weighted undirected graph in two sub-sets maximizing the sum of the weights of edges between them.

To model the problem, a binary variable per vertex is used, which is 1 if the corresponding vertex is in a set, 0 if it is in the other. To model if the edge from vertex i to j is in the cut, the quantity $x_i + x_j - 2x_ix_j$ is defined, this is 1 if $x_i \neq x_j$, 0 otherwise. So the cost function will be:

$$f(x) = \sum_{i,j} w_{ij} \left(x_i + x_j - 2x_ix_j\right), \tag{3.6}$$

where the couples i,j are the index of vertices connected by an edge in the graph and w_{ij} is the weight of the edge between the vertices i and j.

Considering the graph in Figure 3.1 composed by five vertices and six edges, all with weight equal to 1, the corresponding QUBO model for the max-cut problem

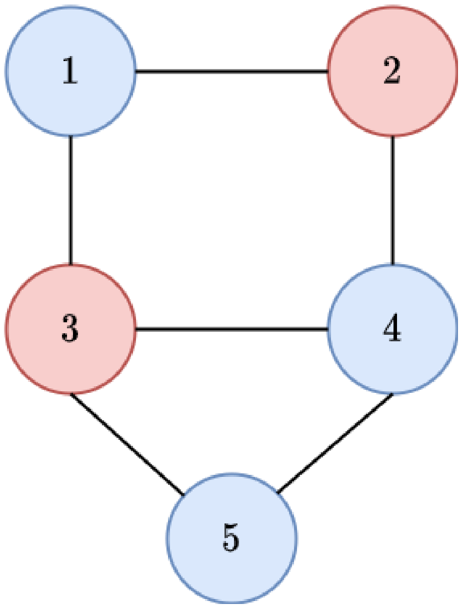


Figure 3.1: Reference example for the max-cut problem

has five binary variables and the cost function is:

$$\begin{aligned} f(x_1,x_2,x_3,x_4,x_5) &= (x_1 + x_2 - 2x_1x_2) + (x_1 + x_3 - 2x_1x_3) + \\ &+ (x_2 + x_4 - 2x_2x_4) + (x_3 + x_4 - 2x_3x_4) + \\ &+ (x_3 + x_5 - 2x_3x_5) + (x_4 + x_5 - 2x_4x_5) = \\ &= 2x_1 + 2x_2 + 3x_3 + 3x_4 + 2x_5 - 2x_1x_2 - \\ &- 2x_1x_3 - 2x_2x_4 - 2x_3x_4 - 2x_3x_5 - 2x_4x_5 \end{aligned} \tag{3.7}$$

This corresponds to a symmetric Q matrix:

$$Q = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{3.8}$$

In this case, the optimal solutions are $x = [0,1,1,0,0]$ or $x = [1,0,0,1,1]$, which means that the first set contains the nodes 2 and 3 (the red vertices in Figure 3.1) and the second one contains the nodes 1, 4 and 5. The total number of cuts is equal to 5.

3.3 QUBO formulations for machine learning

Different machine learning problems can be modelled by means of a QUBO representation [21]. In particular, two formulations for training clustering [22] and linear regression [4] are presented.

3.3.1 Clustering

Clustering is an unsupervised learning model that permits to divide a set of points in some subsets (clusters) grouping them by the proximity to a common centroid, Figure 3.2 reports an example dataset for a clustering problem, where every cluster has a different colour.

Assuming that every cluster has the same dimension, this problem can be expressed as:

$$\min_{\Phi} \sum_{i=1}^{k} \sum_{x,y \in \Phi_i} \|x-y\|^2, \tag{3.9}$$

where:

- $\Phi = \{\Phi_1, \ldots, \Phi_k\}$ is the set of clusters;
- k is the number of clusters;
- x and y are elements of the training dataset $X \in \mathbb{R}^{N\times d}$ (N is the number of elements of the dataset, d is the dimension of every element of the dataset).

The objective is to find k clusters each containing the N/k nearest points.

To model this problem with a QUBO formulation, a matrix $D \in \mathbb{R}^{N\times N}$ where

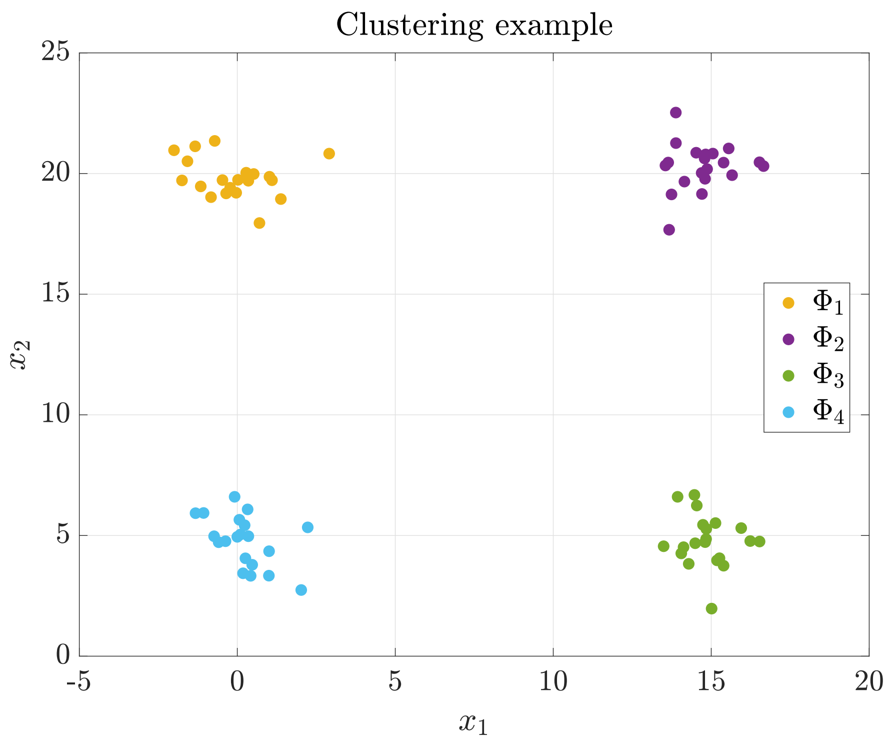


Figure 3.2: Clustering dataset

every element is given by:

$$d_{ij} = \|x_i - x_j\|^2 \quad \forall \quad x_i, x_j \in X, \tag{3.10}$$

and a matrix $\hat{W} \in \mathbb{B}^{N\times k}$, with elements:

$$\hat{w}_{ij} = 1 \quad \text{iff} \quad x_i \in \Phi_j, \tag{3.11}$$

are defined. It is possible to define a vector of binary variables $\hat{w}$ obtained by vertically stacking the elements of $\hat{W}$. This is the vector of binary variables of Equation (3.2), while the Q matrix is $\mathbb{1}_k \otimes D$, where $\mathbb{1}_k$ is the k-dimensional identity matrix and $\otimes$ is the Kronecker product.

Since every point is exactly in one cluster, the $\hat{W}$ matrix contains only a 1 for every row. This constraint has to be added to the model, so a penalty on the sum of the elements of each row of $\hat{W}$ has to be added. This can be done by exploiting the relations in Table 3.1. Moreover, the sum of the columns elements of $\hat{W}$ has to be equal to N/k to guarantee that every cluster has the same number of elements. In order to do so something similar is done. The mathematical steps are explained in detail in [22].

3.3.2 Linear regression

The linear regression model is one of the most used statistical machine learning techniques [4]. The objective of this technique is to find a straight line that interpolates a dataset guaranteeing that the distance between its points and the dataset is minimised. Figure 3.3 reports a dataset and its interpolating line.

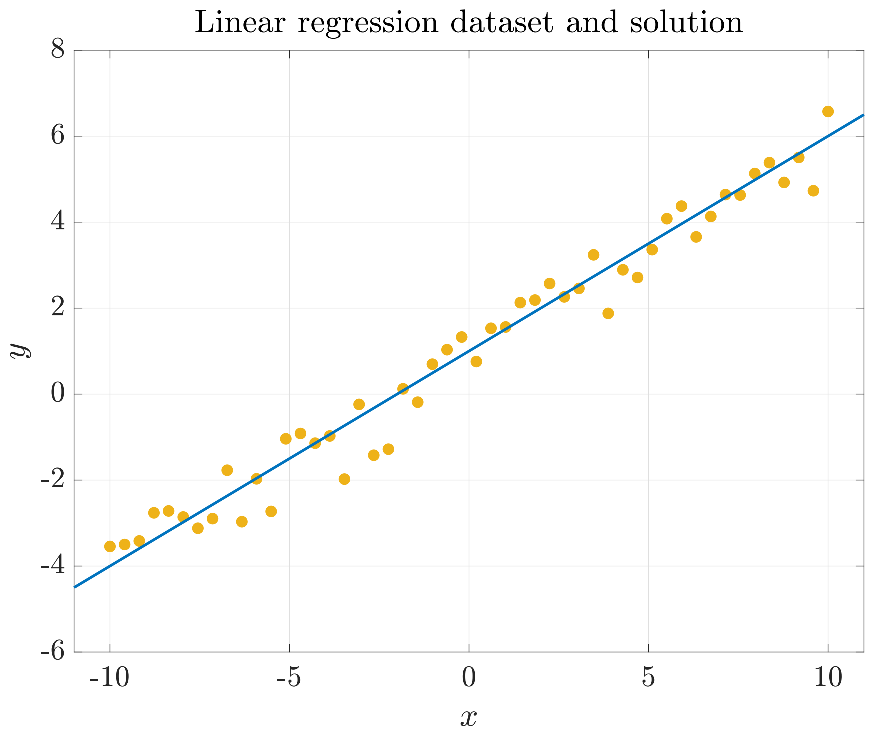


Figure 3.3: Linear regression problem, dataset and solution

Let $X \in \mathbb{R}^{N\times(d+1)}$ be the augmented dataset and let $Y \in \mathbb{R}^N$ be the vector of labels. X is obtained vertically stacking the N d-dimensional points in the dataset, each followed by a one, so every row of X contains a point and a 1. This notation introduces some mathematical simplifications.

In the case of points in a plane (as the problem in Figure 3.3) d is 1, so X contains the abscissa of every point plus a 1, while Y contains its ordinate. The interpolating line is completely defined by a vector of regression weights $w \in \mathbb{R}^{d+1}$. In the $2-D$ case of Figure 3.3, the entries of w represent the slope and the intercept of the interpolating line and d is equal to 1. If $d > 1$, w represents a hyperplane. Under this notation, the problem can be written as:

$$\min_{w} \quad E(w) = \|Xw - Y\|^2, \tag{3.12}$$

where $E(w)$ represents the Euclidean error function.

The QUBO formulation can be defined starting from a rewriting of Equation (3.12) as:

$$\min_{w} \quad E(w) = w^T X^T X w - 2w^T X^T Y + Y^T Y. \tag{3.13}$$

Furthermore, a precision vector P is defined as $[p_1,p_2,\ldots,p_k]^T$ with $k \in \mathbb{N}$, where every p_i is a signed integral positive or negative power of 2. P has to be sorted. Then a vector $\hat{w} \in \mathbb{R}^{k(d+1)}$ containing k binary variables per each entry of w is defined such that:

$$w_i = \sum_{j=1}^{k} p_j \hat{w}_{ij} \quad \forall \;\; w_i \in w, \tag{3.14}$$

where w_{ij} is a binary decision variable that, depending on if it is 0 or 1, ignores or select the j-th entry of P to approximate w_i. Thus, if P contains only positive powers of 2, w_i can assume one of 2^k possible values, while, if P contains negative numbers too, the number of possible values for w_i reduces. For example, if $P = \left[-1,-\frac{1}{2},\frac{1}{2},1\right]^T$, w_i is in $\left\{-\frac{3}{2},-1,-\frac{1}{2},0,\frac{1}{2},1,\frac{3}{2}\right\}$ and 8 binary variables are required for $\hat{w}$.

Equation (3.14) can be rewritten in a matrix form as:

$$w = \mathscr{P}\hat{w}, \tag{3.15}$$

where $\mathscr{P} = \mathbb{1}_{d+1} \otimes P^T$ is the precision matrix. Substituting Equation (3.15) in Equation (3.13), the linear regression problem becomes:

$$\min_{\hat{w} \in \mathbb{B}^{k(d+1)}} \quad E\left(\hat{w}\right) = \hat{w}^T \mathscr{P}^T X^T X \mathscr{P} \hat{w} - 2\hat{w}^T \mathscr{P}^T X^T Y, \tag{3.16}$$

that is a proper QUBO formulation of the linear regression problem, indeed a quadratic and a linear part can be recognized in the cost function $E\left(\hat{w}\right)$.

Some considerations are necessary, the possible solutions to the QUBO model for the linear regression are discrete and bounded. In fact, not all slopes and intercepts can be obtained by the precision matrix $\mathscr{P}$. In the aforementioned example, the maximum slope and intercept are ± 1, while the minimum, apart from 0, are $\pm \frac{1}{2}$. Moreover, the number of possible lines is discrete and so it is not possible to obtain an intermediate line between the admissible one.

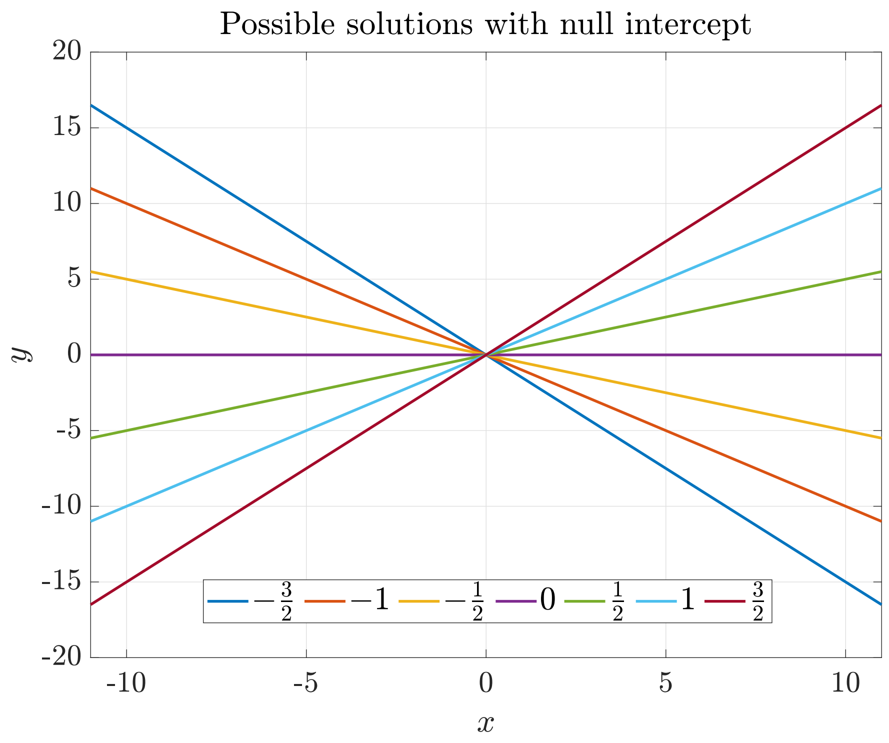


Figure 3.4: Possible solutions to the linear regression problem, $P = \left[-1, -\frac{1}{2}, \frac{1}{2}, 1\right]^T$ and null intercept. The legend contains the slope of the solutions

Figure 3.4 reports a plot of the possible slopes obtainable using a precision vector $P = \left[-1, -\frac{1}{2}, \frac{1}{2}, 1\right]^T$.

In order to increment the number of representable lines, k has to be incremented and so also the number of involved binary variables, which scales as $k(d+1)$. Therefore, the addition of a resolution step to the considered example ($\pm\frac{1}{4}$) adds 4 binary variables obtaining a problem with 12 total variables.

3.4 Solvers

Combinatorial optimization problems can be solved by different techniques and algorithms. This section reports some among the plethora of optimization algorithms, to have a deeper overview on these algorithms see the Master thesis [23].

3.4.1 Classical solvers

Optimization problems are among the most studied computer science problems. During years different approaches to find the optimal solution have been proposed. One of the main classical techniques is simulated annealing, which is used in this thesis as reference approach for validating Grover Adaptive Search behaviour.

Simulated annealing

Simulated annealing [24] is an optimization technique which bases its main idea on annealing in metallurgy, which is a process requiring to heat and then slowly cool down a metal alloy so that its crystalline structure is reorganized to modify alloy's hardness.

In the simulated annealing algorithm, the solution domain is explored, exploiting artificial thermal fluctuation to escape from local minima. A neighbour solution is always accepted if it introduces an improvement, otherwise, if it does not introduce any improvement, i.e. corresponds to a higher cost, it can be accepted if:

$$\rho < e^{\frac{\Delta C}{T}}, \tag{3.17}$$

where ρ is a random number between 0 and 1, ΔC is the difference between the cost of the considered neighbouring solution and the cost of the old solution, T is the temperature, which is a parameter that decreases at every iteration of the annealing algorithm. The acceptance mechanism simulates thermal fluctuation and it permits to escape from local minima. Therefore, the probability of accepting a worse solution decreases at every iteration until temperature is zero.

This algorithm is well-tested and efficient, but it could require high computational times. Moreover, the escape mechanism is efficient only with cost functions that does not have too high derivative, so that does not present high and narrow peaks.

More details can be found in the Master thesis [23].

3.4.2 Quantum solvers

Quantum computing has proven to be a promising computational paradigm to solve optimization problems. In fact quantum phenomena and quantum parallelization can be used to speed up the optimum search.

Quantum annealing

Quantum annealing was proposed to overcome limits of simulated annealing [25], exploiting quantum fluctuations instead of thermal one.

The problem cost function is mapped onto an Ising Hamiltonian [26]. It can be demonstrated that a strong relation exists between the QUBO formulation and the Ising Hamiltonian and it is possible to go from a formulation to the other one without loss of information in the problem description [23].

A time-dependant transverse field is applied to a spin system that embeds the Ising model of the problem. The transverse field permits tunnelling between eigenstates of the system doing a **global search** in the solution domain, in contrast to what simulated annealing does, i.e. **local search**.

The transverse field in quantum annealing plays a role similar to the temperature parameter in simulated annealing algorithm. In order to assure the convergence of the spin configuration to the minimum energy state, which is the problem solution, it is required a slow decrease of the transverse field. At the beginning, the transverse field is high to obtain the quantum superposition of all possible states, then it is decreased sufficiently slow to follow in each time instant the Hamiltonian ground state.

Adiabatic quantum computation is the type of computation performed on D-Wave's quantum annealers [27], the one used as a reference model for quantum annealing in this thesis, and it is based onto the basic ideas presented before. To have a better understanding of this topic refer to the Master thesis [23]. The main problems of this approach are the limited connectivity of qubits, requiring embedding technique to represent fully-connected problems, and no idealities of quantum hardware.

Grover Adaptive Search

Grover Adaptive Search (GAS) is a hybrid quantum/classical algorithm exploiting the quantum Grover's algorithm, which efficiently searches items in a database meeting certain requirements, to find the solution of an optimization problem. Next chapter provides more information about GAS and its implementation.

Chapter 4

Grover Adaptive Search

Grover Adaptive Search (GAS) is a quantum-classical mixed algorithm firstly proposed in [28]. It permits to find the solution of an optimization problem by exploiting the Grover's algorithm, introduced in Section 2.4.1. In particular, this algorithm is exploited as internal routine of an iterative algorithm and its results are processed in a classical way.

This chapter contains a high-level overview of the algorithm, with some considerations about the types of problems suitable for this technique. Furthermore, design choices are motivated. In conclusion an implementation of the algorithm is presented with the discussion of the relevant design parameters. A significant example is also analysed step by step in order to explain clearly the algorithm behaviour.

4.1 Algorithm overview

Let $f(x)$ be the objective function of a minimization problem, e.g. the one plotted in Figure 4.1, where the x axis represents the solution space of the function and the y axis contains the cost space. Every x in the solution space is mapped onto a y. This problem can be expressed as:

$$\text{find } x \mid x = \arg\left[\min\left(f(x)\right)\right] \tag{4.1}$$

A possible approach to obtain the minimum value of an objective function $f(x)$

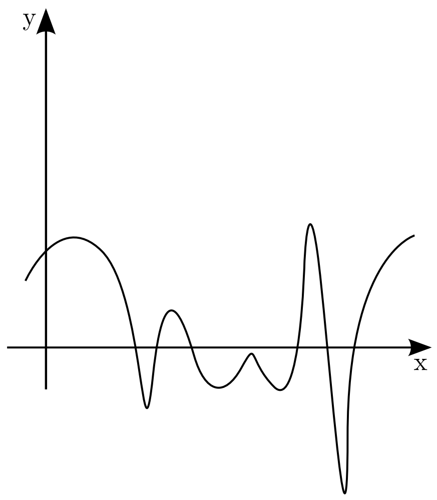
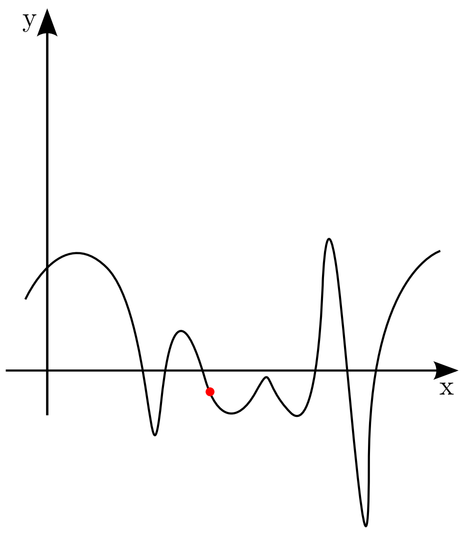
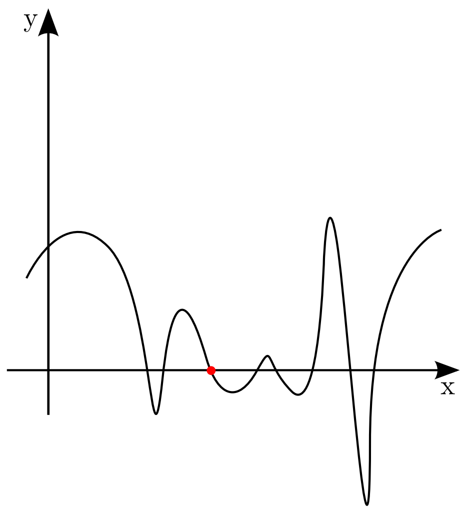


Figure 4.1: Objective function example

(Figure 4.1) is based on sampling a negative value of it ($f(x_{neg})$) and translating it vertically of the sampled amount, so that $f(x_{neg})$ becomes equal to zero (Figure 4.2).



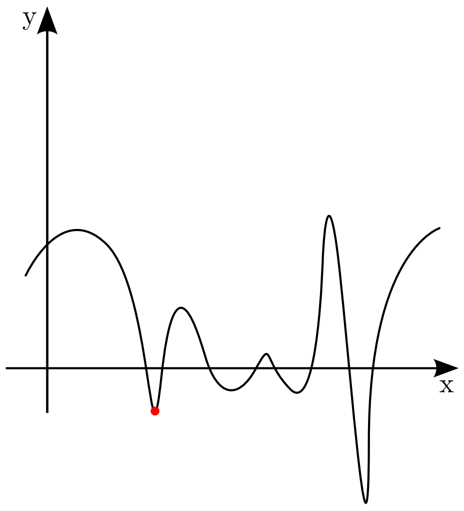
(a) *First negative sample*



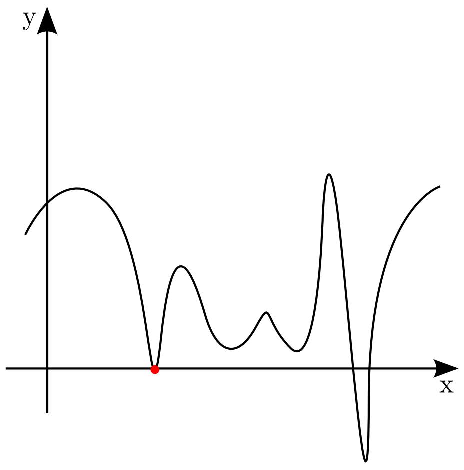
(b) *First shift*

Figure 4.2: Example procedure

Sampling and shift procedure is iterated (Figure 4.3) until the cost function has no negative values and the minimum has a null value, as reported in Figure 4.4.



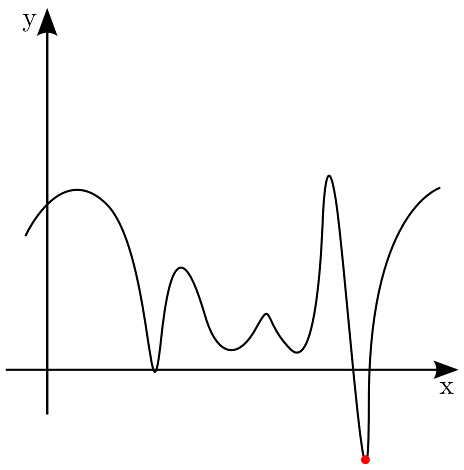
(a) *Second negative sample*



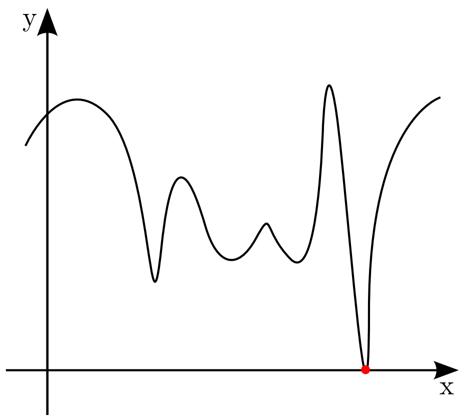
(b) *Second shift*

Figure 4.3: Example procedure (cont'd)

Quantum computing can be used to obtain a fast and efficient method to sample negative values.



(a) *Last sample*



(b) *Last shift*

Figure 4.4: Example procedure (cont'd)

In particular, the Grover's algorithm, presented in Section 2.4.1, can be used to find one ore more item that satisfies a certain constraint (in this case to be lower than zero), in a faster way than a classical counterpart could do. A quantum dictionary, introduced in Section 2.4.4, can be used to encode in a quantum state the cost function of the optimization problem. On the other hand, classical computing can be used for states preparation, required by Grover's algorithm and for shifting the cost function.

Therefore, it is possible to implement an efficient method to find the minimum of an objective function, which exploits both quantum and classical computing paradigms, this is called Grover Adaptive Search. Figure 4.5 summarizes this mixed approach.

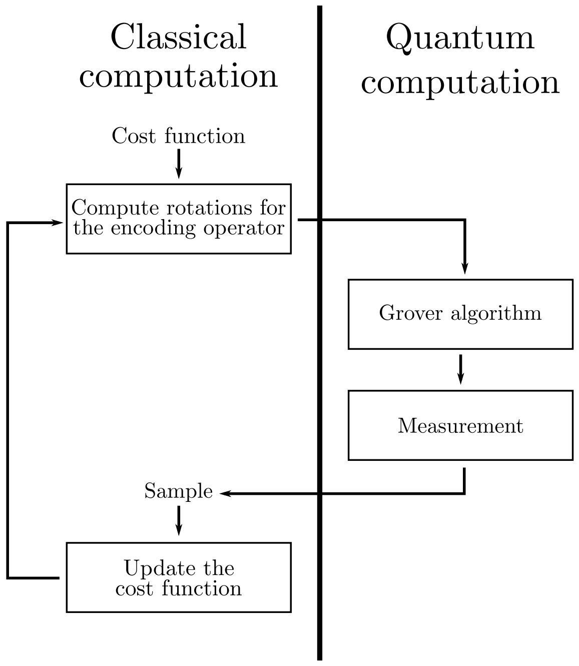


Figure 4.5: Algorithm scheme

4.1.1 GAS compliant optimization problems

As presented in [29], Grover Adaptive Search can be used to solve Constrained Polynomial Binary Optimization (PCBO) problems. The binary nature of the cost function of this class of problems permits a natural encoding of the cost function with a quantum dictionary. Constraints can be added acting on the oracle of the Grover Algorithm, indeed it is possible to search for solutions that are both negative and satisfy a certain constraint.

QUBO problems are a subset of PCBO problems, therefore they can be solved by using the Grover Adaptive Search algorithm, naturally the dictionary can be simplified and the oracle is responsible of labelling the negative values of the cost function.

4.2 Algorithm design

As already mentioned, the GAS algorithm exploits both quantum and classical computation. The Grover's quantum search algorithm samples the negative values of the cost function according to the probability distribution reported in Equation (2.55)

As noted above, Grover Adaptive Search algorithm is a mixed algorithm, in particular Grover algorithm is used to sample the objective function, in fact the final measurement produces a sample that follows the probability distribution in Equation (2.55), then it is used to update the state preparation circuit, a quantum dictionary, to shift the cost function.

Shifting operation is performed subtracting the last outcome of the Grover's algorithm, so that, if the cost function initially does not contain negative values (it is in the case in which $p = 0$ in Section 2.4.6), at the second step there will be at least a negative value.

Moreover, it is important to remind that the probability of measuring a positive value is not null. For this reason, multiple samples in the same condition could be required for obtaining a negative value, otherwise the management of a *stop condition* after a given number of consecutive positive measures must be done.

At the end of the algorithm, the cost function has no negative values and sampling gives a random value from the cost function, that at this point is all positive.

To summarize, Grover's algorithm produces a positive sample of the cost function in three cases:

- The starting cost function has no negative values;
- There is an *unlucky error* in the Grover's algorithm;
- The absolute minimum has been found and no negative values remain in the cost function.

The first two cases are handled checking whether the last sample introduces an improvement or not, while the third case must be identified to determine the stop condition.

To do this, the fact that in this regime the probability distribution of Equation (2.55) is uniform, because of the presence of only positive values, is exploited. The number of obtained subsequent positive samples is measured and if it reaches a certain threshold t the algorithm ends.

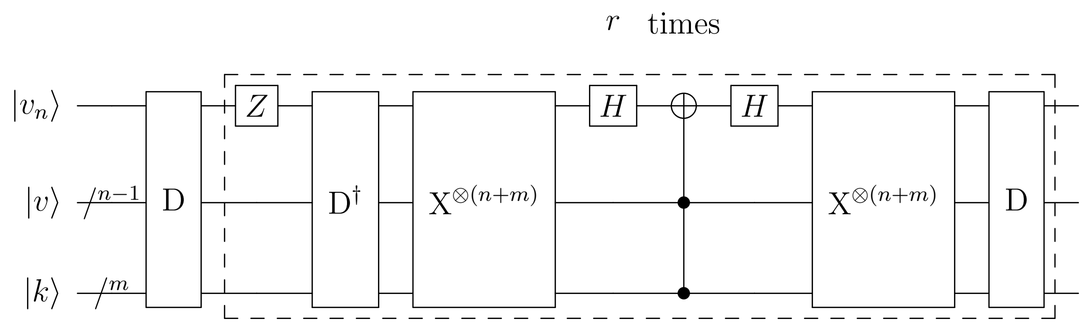


Figure 4.6: Grover algorithm

The implementation of the quantum part requires the design of some circuits to implement the Grover's algorithm: the quantum dictionary for the state preparation has been designed by following the methodology presented in Section 2.4.4. The oracle label the states with negative values by inverting their phase and this is performed by applying a Z gate on the qubit associated with the most significant bit of the value register, representing the sign of a number in two's complement encoding. The diffusion operator is standard, i.e. the same reported in Section 2.4.5. Figure 4.6 reports the circuit that realizes the Grover algorithm.

Shifting can cause overflow. In fact, the image of the cost function increases and the selected number of qubits to represent it could become not sufficient. In particular, out-of-range values are considered as negative values by Grover's algorithm because in the two's complement encoding, overflow flips the most significant bit, i.e. the sign of the number. In order to prevent this condition, an higher number of qubits can be used. The aforementioned check on the validity of the solution guarantees no errors, simply more sampling will be needed.

4.2.1 Non-integer encoding

As described in Section 2.4.4, the quantum dictionary circuit encodes integer-valued functions, mapping their coefficients to integer multiples of a base angle.

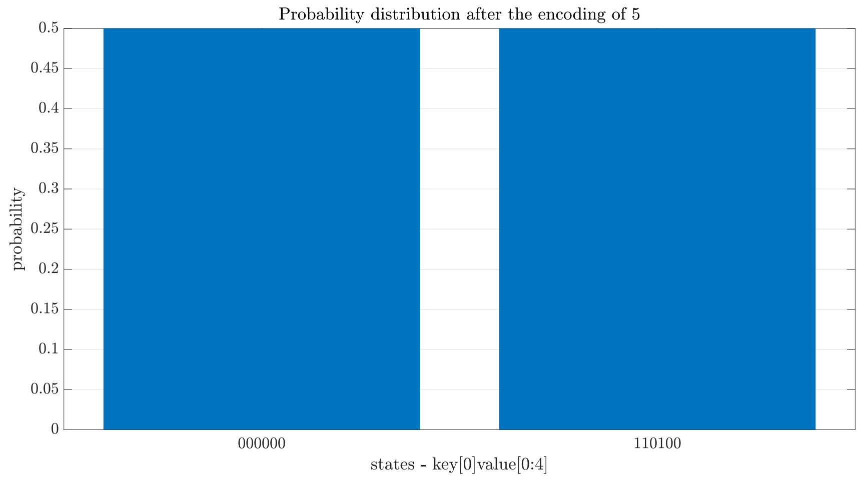


Figure 4.7: Probability distribution of the state after the encoding of an integer number

Unfortunately, the cost function could have real coefficients. These can be mapped to non-integer multiples of the base angle giving a certain acceptable error, spreading the probability of that state to near integers.

In order to better explain the adopted strategy, the encoding of an integer coefficient (5) and of two real coefficients (5.5 and 5.2) are commented. Figure 4.7 reports the probability distribution of the monomial $5x_0$, where on the x axis there are the states, with probability amplitude different from zero, as strings of bits composed by the concatenation of the key and the value registers with the least significant bit first. As it is possible to notice, the probability of the state in which x_0 is 0 or 1 are equal and when $x_0 = 1$, the value register contains 5 represented on 5 bits.

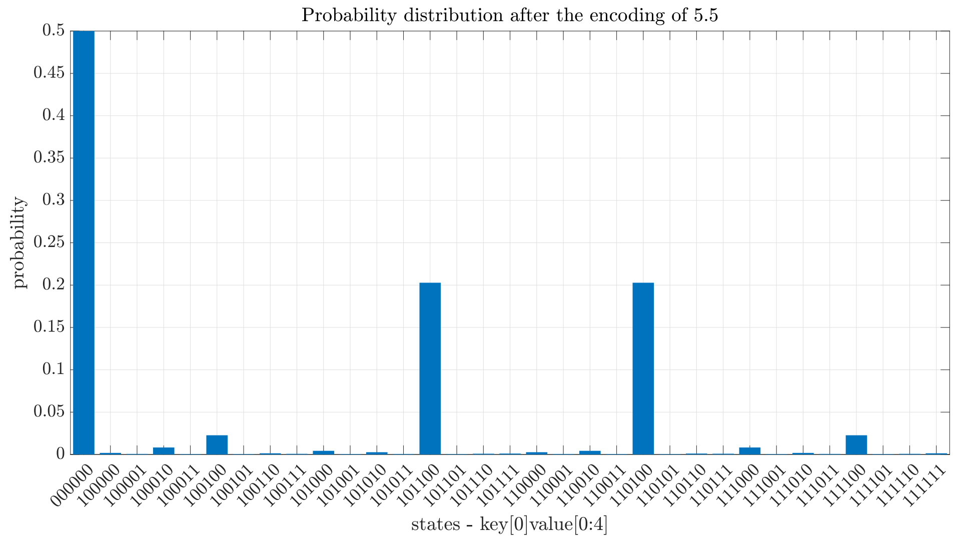


Figure 4.8: Probability distribution of the state after the encoding of a non-integer number

Figure 4.8 and Figure 4.9 report the encoding of $5.5x_0$ and $5.2x_0$ respectively. It is interesting to notice that the first monomial gives equal probability of obtaining 5 and 6, while in the second one the probability of obtaining 5 is higher than that of obtaining 6.

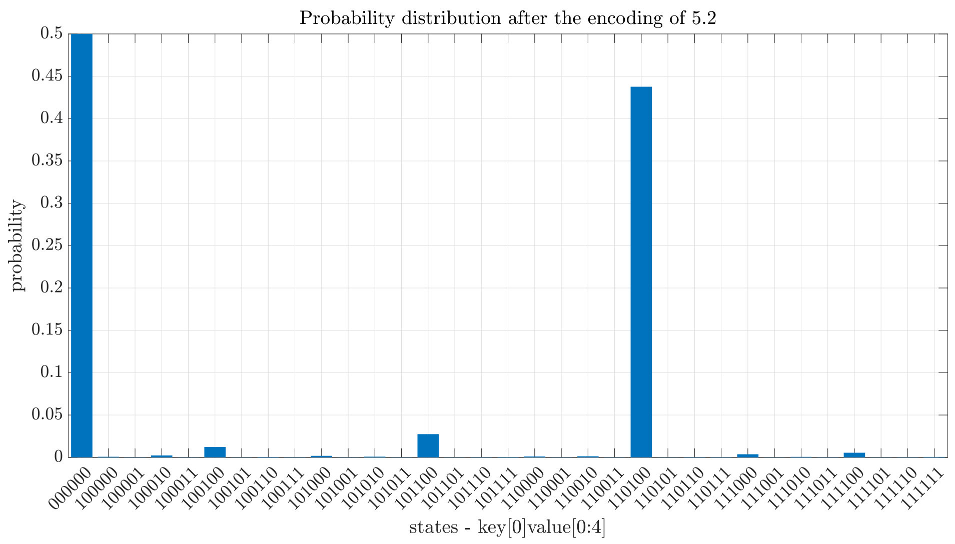


Figure 4.9: Probability distribution of the state after the encoding of a non-integer number

4.2.2 Handling constraints

As mentioned in Section 4.1.1, Grover Adaptive Search supports also problems with explicit constraints [29].

In order to add constraints to the target optimization problem, it is sufficient to modify the Grover oracle adding conditions to the key register. For example, if the number of bits at 1 in the solution has to be fixed to a certain value, the obtained solution could not correspond to the global minimum of the cost function, but to the lowest value which satisfies the applied condition.

Naturally, adding constraints requires a different design of the oracle to include them, indeed it has to label costs that are both negative and satisfy the constraints.

More complex constraints can be added using arithmetic circuits and auxiliary registers, so more than one ancilla qubits are required to maintain the information about different constraints. If they are all satisfied, the most significant bit of the value register is flipped, as in the previously described oracle.

4.3 Implementation

The flowchart of the implemented Grover's algorithm is reported in Figure 4.10.

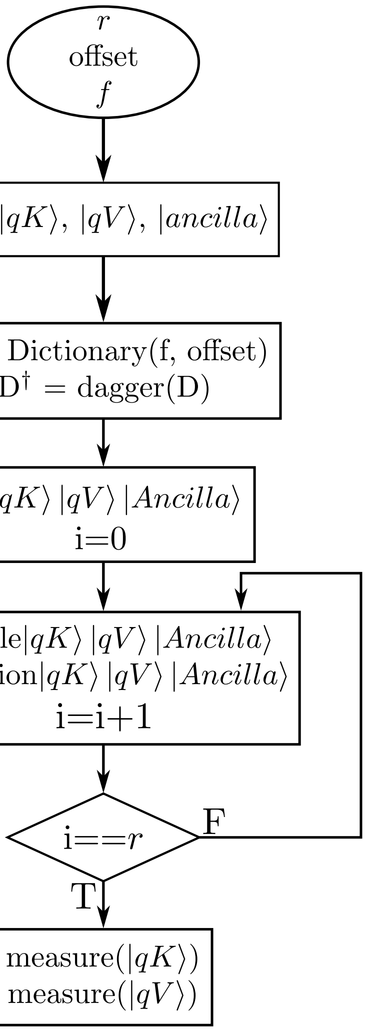

r
offset
f
Init $|qK\rangle$, $|qV\rangle$, $|ancilla\rangle$
D = Dictionary(f, offset)
D† = dagger(D)
D$|qK\rangle\,|qV\rangle\,|Ancilla\rangle$
i=0
Oracle$|qK\rangle\,|qV\rangle\,|Ancilla\rangle$
Diffusion$|qK\rangle\,|qV\rangle\,|Ancilla\rangle$
i=i+1
i==r
F
T
measure($|qK\rangle$)
measure($|qV\rangle$)

Figure 4.10: Flowchart of the Grover's algorithm

The implemented Grover's algorithm takes as input r, the number of iterations of the Grover's operator, the cost function f and the offset to apply. $|qK\rangle$ and $|qV\rangle$ are two quantum registers with a number of qubits sufficient to contain the domain/image pairs of the cost function (n and m respectively), $|ancilla\rangle$ is an ancilla qubit to permit the encoding of the dictionary, they are initialized to $|0\rangle^{n+m+1}$

at the beginning of the procedure. A unitary operator D is defined as the quantum dictionary starting from the cost function and the shift to be applied to it (it is an incremental shift, so it is relative to the previous shift). Its inverse $D^\dagger$ is then computed.

A first application of D prepares the state, then the Grover operator is applied r times. In the end the measurement of the quantum registers is returned.

Figure 4.11 reports the flowchart of the GAS algorithm, the algorithm takes as input the cost function of the optimization problem (f) and t that is the aforementioned threshold, *cnt* is the counter of the positive values returned by the Grover's algorithm, *minX* and *minV* are the solution to the optimization problem and its cost respectively.

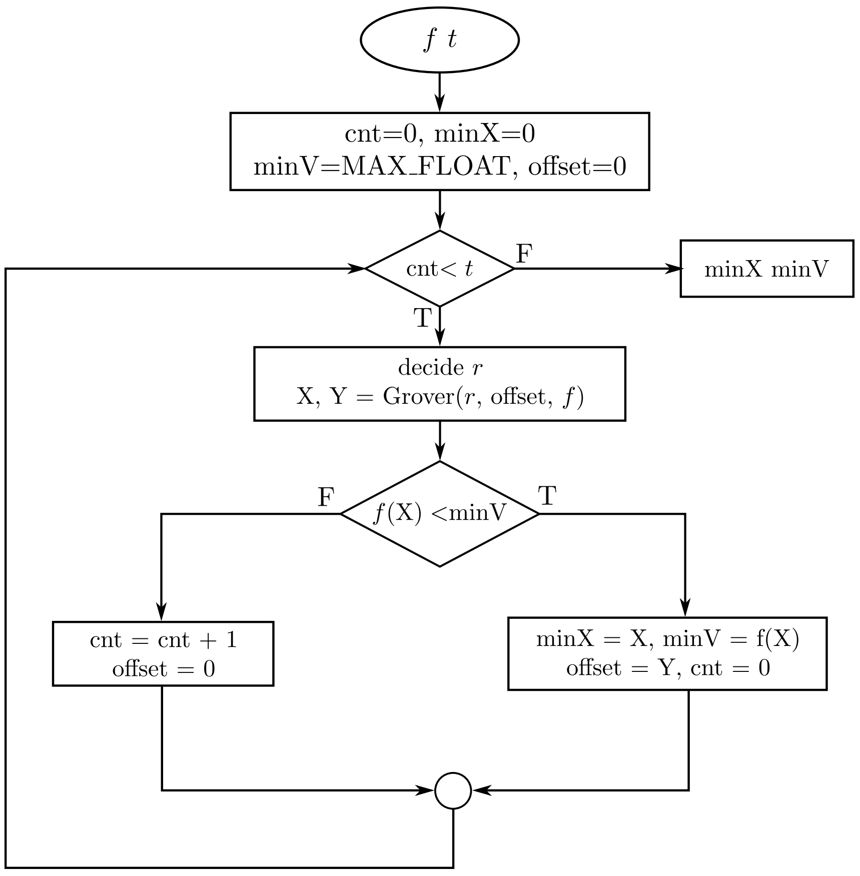


Figure 4.11: Flowchart of the GAS algorithm

The main loop is composed by different operations:

1. decision of the Grover's operator iterations number r, that, as stated before, has to be chosen at every iteration of the Grover Adaptive Search algorithm;

2. application of the Grover's algorithm.

Then a check on the value of the cost function in the point returned by the Grover's algorithm is present:

+ if the value of the cosT has a lower value than the last minimum value, the minimum is updated, the offset to apply as a shift to the cost function is updated and *cnt* is put to 0;

- otherwise, the positive counter is incremented and the offset is put to 0.

As stated before, the offset is incremental, in fact it is added to the dictionary by the encoding of a new term, so if it is required to repeat the Grover algorithm with the same function it is put to 0, while if a shift has to be applied this has to be the value of the previously shifted function, not the value of the original function.

The described algorithm was implemented in **Python** language.

The Grover's algorithm has been implemented with the open-source framework **ProjetQ** [30], which permits to integrate a quantum routine in a classical programming environment. Its methods permit seamlessly to run a quantum algorithm on different types of simulators/real hardware and to use the obtained result in a classical context.

Moreover, QUBO formulation of a given optimization problem can be also obtained with **Qubovert** Python library [31].

4.4 Parameters

As reported in Section 4.2, t and r are the two fundamental GAS parameters. In particular, the former is a parameter of the classical part of the algorithm, while the other is peculiar to the quantum part.

This section contains a discussion on some strategies to choose these two parameters, considering the fact that t is a *static* parameter decided at design time, while r can be tuned and modified *dynamically* during the algorithm execution.

4.4.1 How to select t

As already mentioned, the algorithm parameter t is a threshold which determines the stop condition of the GAS algorithm. In particular, it corresponds to the number of consecutive positive samples required to be sure that, at that moment, the function is all positive. This strategy has to be adopted because, in a real scenario, there is a non-null probability to obtain a positive value from Grover's algorithm, even if negative ones are still present in the function (see Section 2.4.6).

A higher value of t gives more possibility to the Grover algorithm to output a negative value, remember that if a negative sample is produced, it will have surely a lower value than the previous sample. So this will give a more robust algorithm against false minima, but, when the real minimum is found there are lots of useless samplings. On the other hand, a lower value of t is weaker against false minima, but permits to end the algorithm in a shorter time when the actual minimum is found. In conclusion, a trade-off is required.

4.4.2 How to choose r

As it is possible to notice from Equation (2.55), the value of g_r, which depends on the number of Grover's operator iterations (r), directly affects the probability distribution of the output state. Therefore, to maximize the probability of measuring a labelled state, g_r has to be as near to 1 as possible and it is necessary to properly choose the value of r.

The value of g_r depends on p, which is the fraction of states labelled by the oracle, too. It is important to remind that p depends on the particular problem and changes at every iteration of the Grover Adaptive Search algorithm. In an ideal scenario, it decreases since the number of negative values decreases while the algorithm proceeds.

	$p = 0.1$	$p = 0.3$
$r = 9$	0.0286	0.9997
$r = 2$	0.9986	0.0581

Table 4.1: Values of $g_r(p)$ for different combinations of r and p (Equation (2.56))

According to the previous observations, the optimal r cannot be known a priori and a way to choose its optimal value at every iteration of the GAS algorithm must be found.

Different approaches can be adopted:

- Chose r a priori and apply the same number of iterations at every step of the Grover Adaptive Search algorithm;
- Define a fixed pattern for r and select a different value from the pattern every time the Grover algorithm is executed;
- Chose r randomly at every step of the Grover Adaptive Search algorithm.

r fixed

With this strategy, r is chosen at the beginning, so Grover operator is applied the same number of times at each GAS iteration. This is not particularly efficient since the marked set dimension changes at every iteration. In fact, every time a shift is applied to the cost function, the number of negative values of the cost function decreases, so p in Equation (2.56) changes accordingly and it is not ensured that the chosen r maximizes the probability of measuring a negative value.

In a practical scenario, t must be sufficiently high to compensate the higher probability of sampling with the Grover's algorithm a positive value. This would avoid the wrong evaluation of stop condition achievement, the wrong evaluation of stop condition achievement, but the algorithm would require more iterations for its completion.

r from a fixed pattern

Defining a fixed pattern, from which the total number of iterations of the Grover's operator is chosen, is another strategy proposed for the first time by [17].

The idea is to define a sequence of iterations that can fit every problem submitted to Grover Adaptive Search algorithm. The one proposed in [17] is:

$$[0,0,0,1,1,0,1,1,2,1,2,3,1,4,5,1,6,2,7,9,11,13,16,5,20,24,28,34,2,41,49,4,60]\,, \tag{4.2}$$

Where a 0 iteration count must be interpreted as randomly sampling the searching domain without applying the Grover operator.

The mathematical derivation of this sequence here is omitted, but is reported in the reference paper [17].

This method introduces better results than the previously presented method, considering the number of required iterations of the Grover Adaptive Search algorithm. In particular, it guarantees higher probability of measuring a negative value at each GAS iteration.

The main problem is that it is based on heuristics and assumptions on the evolution of p at every shift operation, so, when these assumptions are not valid, it loses in generality. From a practical point of view, it would require a total number of iterations that can be very high. Moreover, the presented pattern contains some iteration counts that requires a circuit depth that is too high to be handled by simulators.

r taken randomly

The third alternative is to select r randomly in a domain $[0,R]$ that is adapted during the execution of the algorithm. In particular, R is initialized to 1 and is incremented every time the Grover algorithm gives a sample not belonging to the marked set, while it is re-initialized to 1 whenever a solution is obtained.

This technique permits to adaptively tune the iteration count, depending on the results provided by the Grover algorithm, in fact, if the measured state is out of the marked set, next sampling is performed with a different iteration count.

The assumption behind this method is that the result is wrong because of a wrong choice of the iteration count. On the other hand, if the output is in the labelled set, the iteration count is correct and next sampling can be performed restarting from $R = 1$.

The power of this technique resides in the fact that, even if R becomes very high, there is always the chance that the Grover's algorithm is applied with a low r providing a better result. This permits to guarantee a relatively low number of total iterations of the Grover Adaptive Search algorithm and a reasonable circuit depth for every application of the Grover's algorithm.

4.5 Example

To better understand what the designed algorithm does, an example follows. In particular, the cost function in Equation (4.3) is employed:

$$f(x) = x_0 + 3x_1 + 2x_2 - 5x_3 - 6x_1x_2 + 2x_3x_0. \tag{4.3}$$

This cost function does not represent any problem in particular and the plot of it is reported in Figure 4.12. It is possible to see that there are different negative values and the minimum is -6.

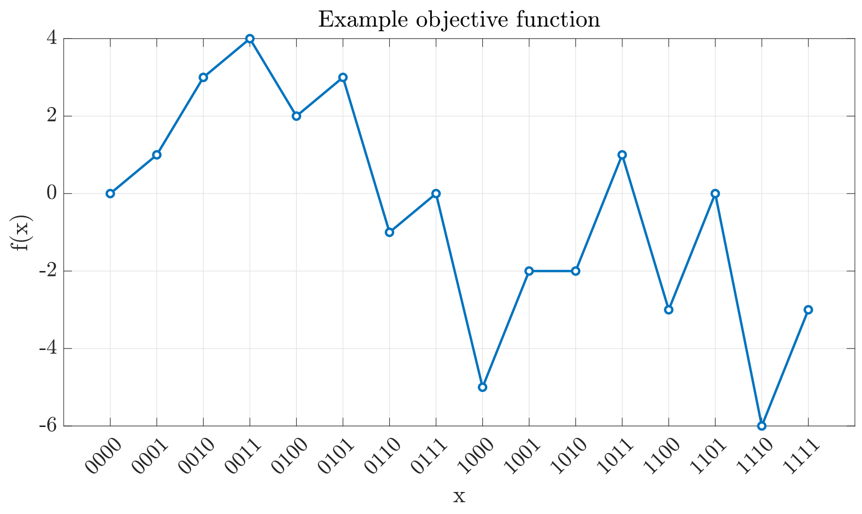


Figure 4.12: Example objective function

The presented example follows a simulation carried out on a classical simulator of qubits without noise, in particular the one integrated in *ProjectQ*, specification on this can be found in [30].

The Grover Adaptive Search algorithm is applied to $f(x)$, setting t equal to 3 and r fixed to 1. Two quantum registers, $|domain\rangle$ and $|image\rangle$, are composed by four and five qubits respectively, that are sufficient to represent the domain/image pairs of the cost function.

The probability distribution created with the quantum dictionary is reported in

Figure 4.13, where on the x axis there are the measurable states as strings of bits composed by the concatenation of key/value pairs with the LSB's first, probabilities of the states with negative values, i.e. with MSB = 1, are highlighted.

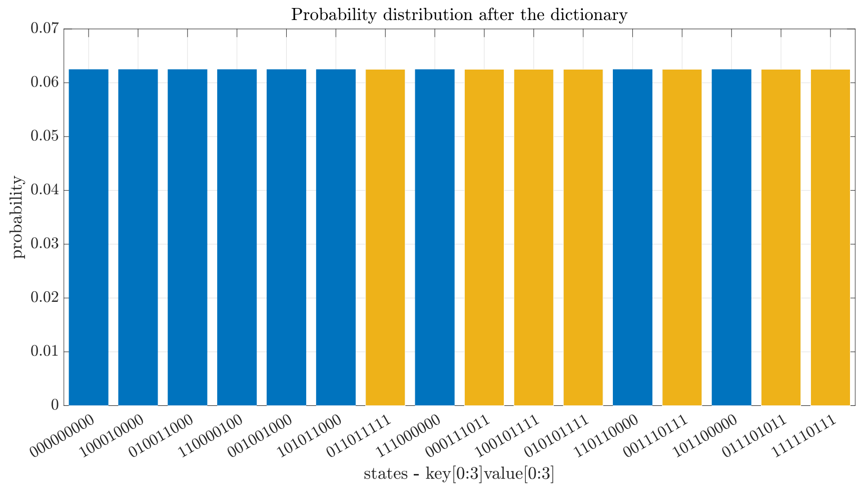


Figure 4.13: Probability distribution of the state after the application of the dictionary encoding the example cost function

After the execution of Grover's algorithm, the probability distribution reported in Figure 4.14 is obtained.

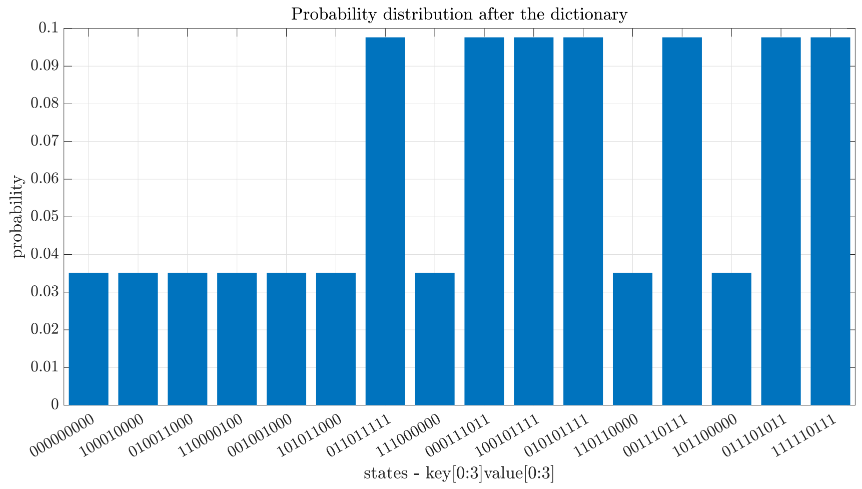


Figure 4.14: Probability distribution of the state after the application of the Grover algorithm

$|domain\rangle$ and $|image\rangle$ registers are so measured obtaining respectively 1001 and 01111. The cost function is then evaluated in the sampled point and shifted as shown in Figure 4.15.

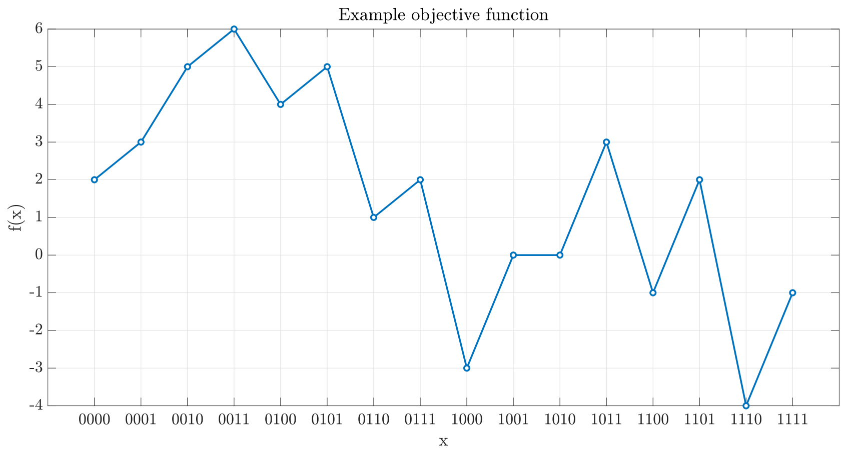


Figure 4.15: The objective function after the first shift

At this point, Grover's algorithm is applied on the new cost function, whose encoding is reported in Figure 4.16.

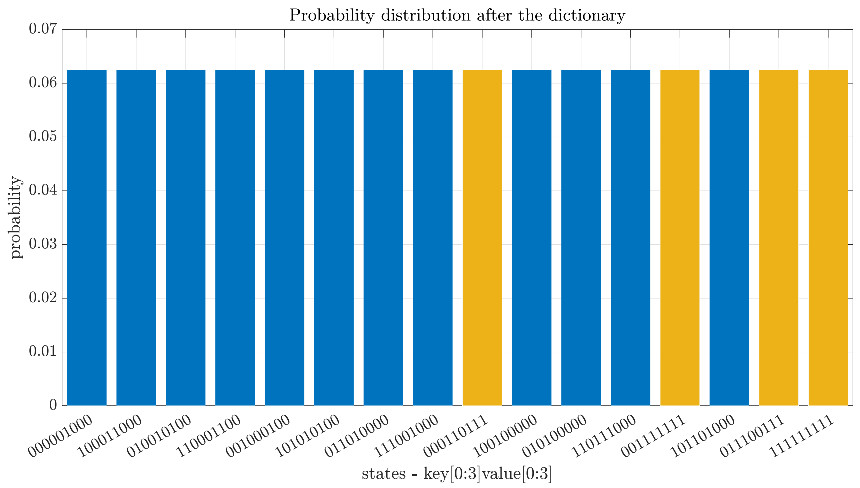


Figure 4.16: Probability distribution of the state after the application of the dictionary encoding $f(x) + 2$

It is possible to notice that, as expected, the number of negative values is lower than in the previous case.

The state probability distribution after the Grover operator is reported in Figure 4.17. In this case, the parameter r is set to the optimal value to maximize the probability of sampling a negative value. Indeed the probaility of sampling a positive value is equal to zero.

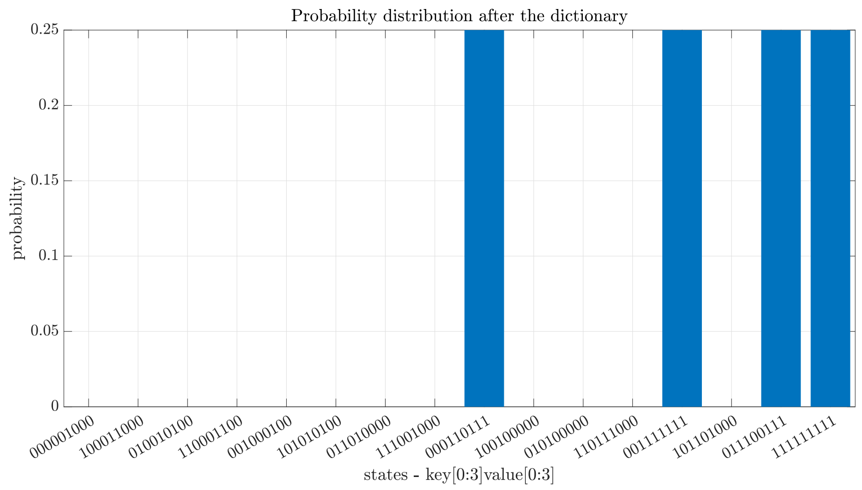


Figure 4.17: Probability distribution of the state after the application of the Grover algorithm

In this case, measurement produces the image/domain couple $0111 - 00111$ and the function is shifted again accordingly, as shown in Figure 4.18. As it is possible to

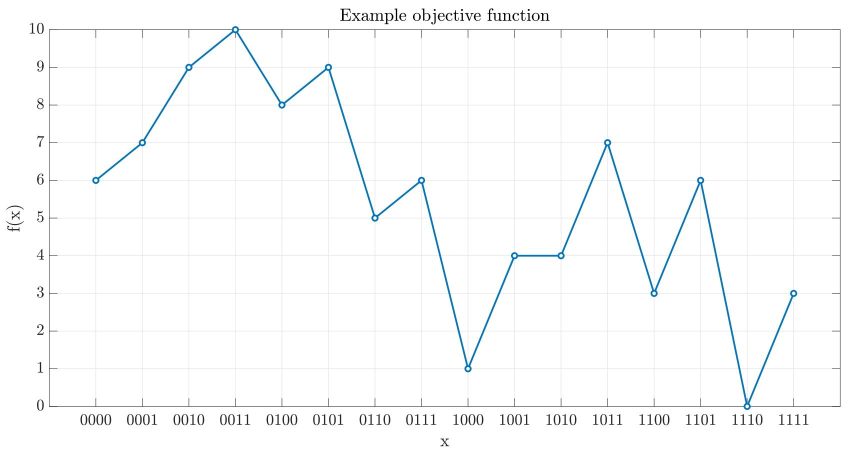


Figure 4.18: The objective function after the second shift

notice, the current function does not contain negative values. Figure 4.19 reports the probability distribution of the state after the dictionary application. It is possible to notice that the probability of measuring a state encoding a negative value is null. Since there are no more negative values in the function, it is the third case of the

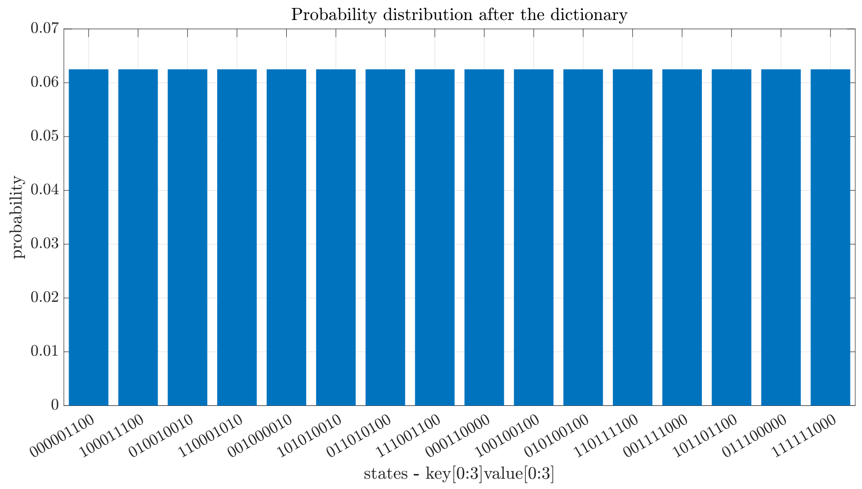


Figure 4.19: Probability distribution of the state after the application of the dictionary encoding $f(x) + 6$

list in Section 2.4.1, so the distribution at the output of the Grover algorithm is uniform over all the searching domain Figure 4.20.

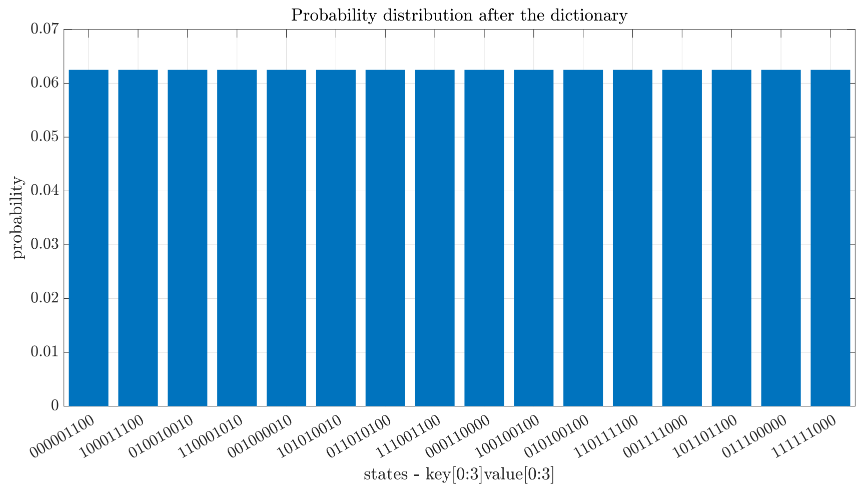


Figure 4.20: Probability distribution of the state after the application of the Grover algorithm

However, it is important to remind that this is not known at runtime, so sampling is repeated until three successive positive values are obtained.

Chapter 5

Grover Adaptive Search Verification

The linear regression problem, presented in [4], has been used to test the validity of the Grover Adaptive Search algorithm implementation.

This chapter presents some considerations about possible strategies to chose parameters of Grover Adaptive Search. Moreover, the solution of a problem is reported to show the effective functionality of the designed algorithm with a reasonable pair of parameters.

5.1 Parameters tuning

First of all, different parameters settings are tested to evaluate the best combination of t and r. In particular the three methodologies presented in Section 4.4 to select r have been tested for different values of t.

The implemented algorithm was applied many times on the same problem, varying the parameters. The obtained value of the cost function is evaluated to estimate the quality of the considered t-r pair.

In particular, a linear regression problem was considered, the samples were obtained starting from the samples of a straight line with a slope equal to 0.5 and an offset equal to 0.1, on which a white noise with amplitude equal to 0.2 was applied. The precision vector contains 6 elements, hence 12 binary variables are necessary to represent the problem. The key register contains 12 qubits and the values one contains 6 qubits, which are sufficient to represent the cost function, considering also the ancilla qubit of the dictionary, the total number of involved qubits is 19. This number is too high to run the quantum part of the algorithm on the via-cloud free accessible quantum hardware [32], so a simulator has been employed.

5.1.1 r fixed

In the first analysed strategy, the parameters t and r are maintained constant during the execution of the algorithm, so every time the Grover algorithm is executed its main operator is executed r times, while the algorithm ends when t successive positive samples are obtained.

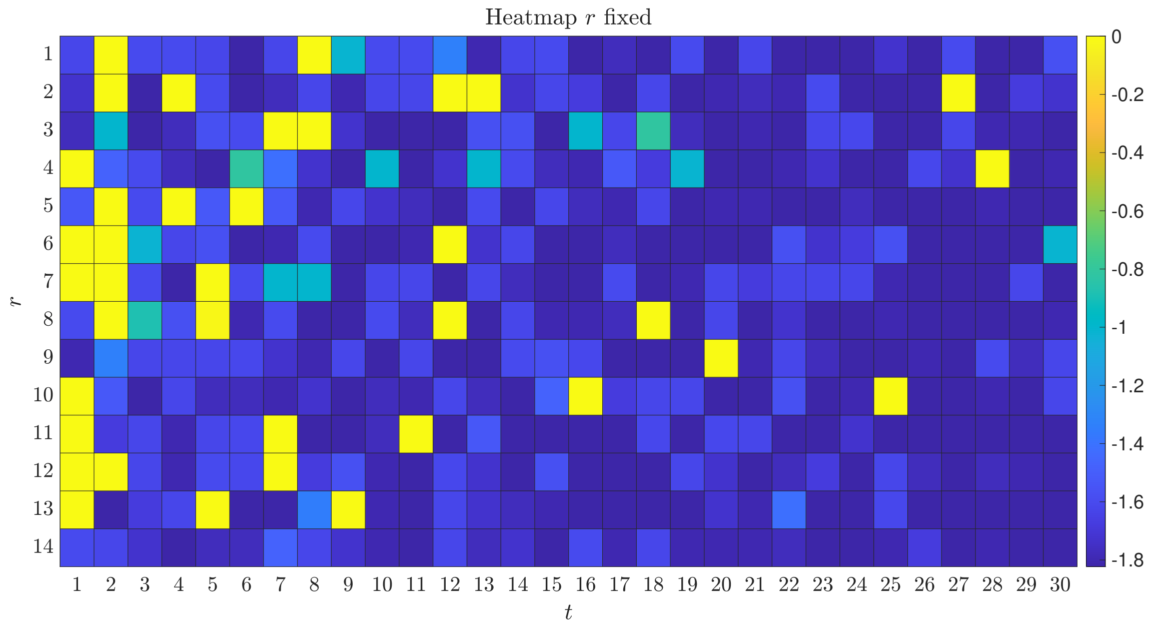


Figure 5.1: Heatmap for the r fixed method

Figure 5.1 reports a heatmap where are reported the cost obtained by executing the algorithm with many couple of parameters t and r. Colours nearer to yellow represent higher costs, while colours towards blue are associated to lower costs, i.e. more optimal results.

As expected, lower costs were obtained for high values of t. It is possible to notice that, increasing the threshold t, the choice of a suboptimal value of r can be partially compensated, Figure 5.2 highlights this.

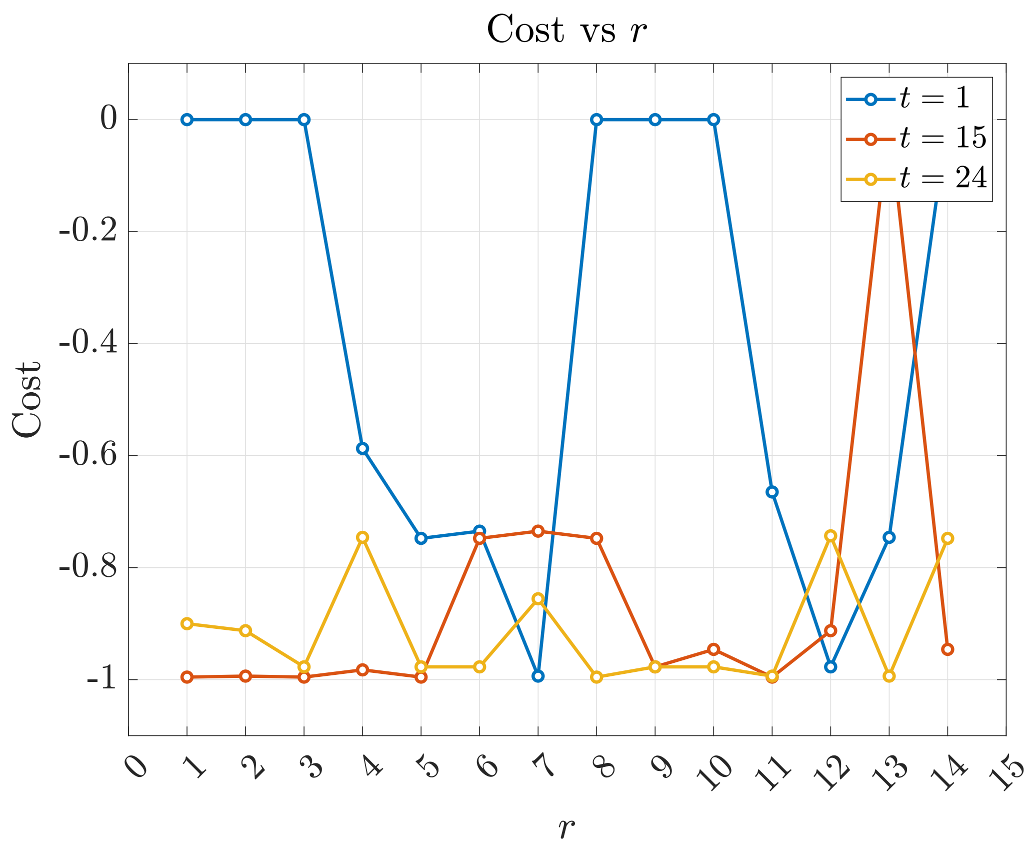


Figure 5.2: Cost vs r for different t

5.1.2 r from a pattern

As mentioned in Section 4.4.2, it is possible to define an array of iteration counts and select one of its elements every time the Grover algorithm is executed. Figure 5.3 reports the final cost obtained executing GAS with different values of t.

The range considered for t variation is limited between 1 and 20, because the used pattern (Equation (4.2)) involves also high values of r which cannot be currently handled by the used simulator, thus it is necessary to chose a stop condition which avoids these.

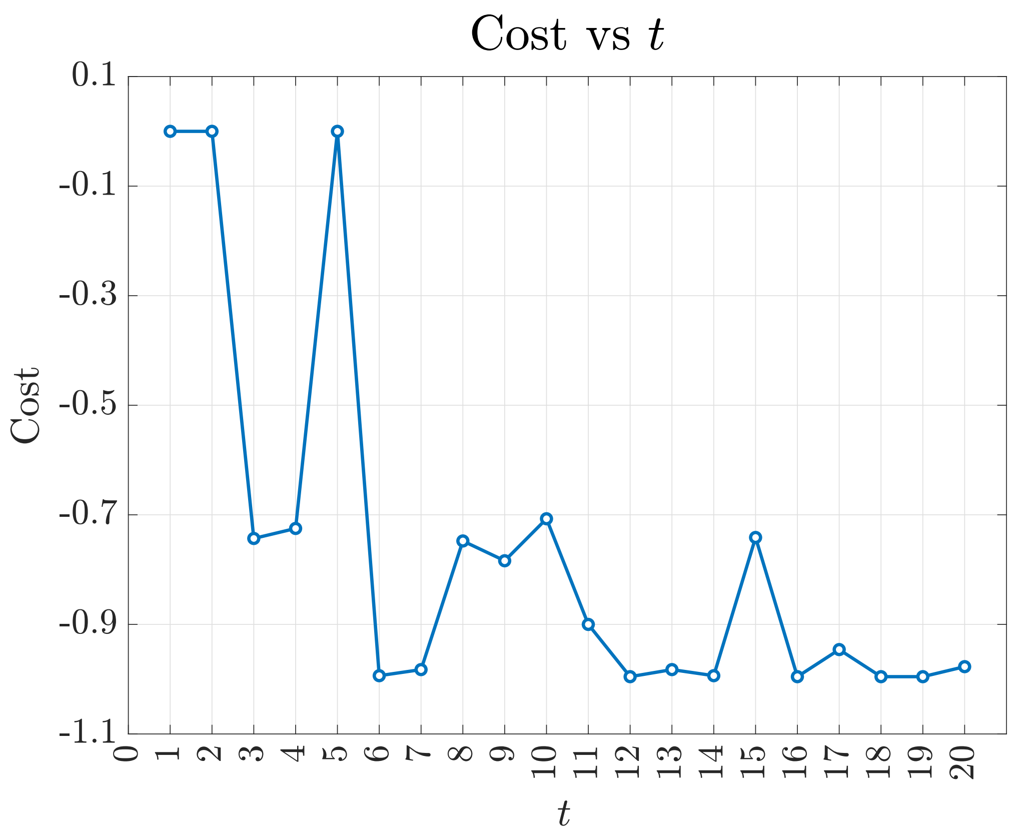


Figure 5.3: Cost vs t for the fixed pattern method

As expected, the cost decreases for higher values of t, but the trend is not monotonic, so this method does not seem to give very reliable results. This anomaly could be reasonably overcome by doing repeated tests.

The main problem of the application of a fixed pattern is that, in order to correctly work, it is required to start from a cost function with a lower number of negative values than positive ones, thus facilitating the negative values search with Grover's algorithm. In order to obtain this condition, it is necessary to estimate the limit of the function image and, eventually, to perform a translation to obtain an initial condition as close as possible to the desired one.

5.1.3 r taken randomly

Another considered methodology consists in choosing r randomly in a set that, proceeding with the execution of the algorithm, adapts its dimension as presented in Section 4.4.2.

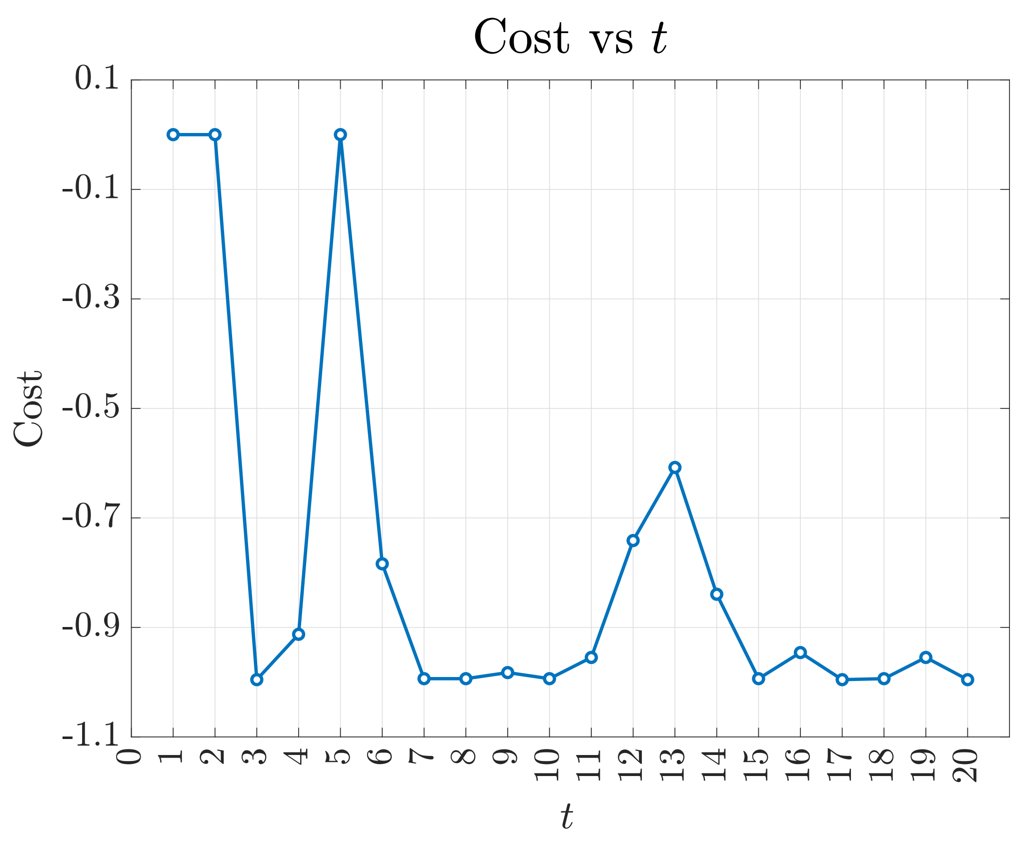


Figure 5.4: Cost vs t for the random method

Figure 5.4 reports the cost of the obtained minimum as a function of the threshold t. As expected, the cost decreases increasing the value of t.

This method has proven to require lower iteration counts than the pattern method and, on average, lower than the one required by the fixed method to obtain comparable results.

Comparison

The comparison of the cost function response with respect to t, employing the three different methods for the choice of r, proved that the random method is the best choice, since it permits to obtain solutions at lower cost requiring lower thresholds, thus a lower number of iterations to perform efficiently the Grover Adaptive Search algorithm. Moreover, it permits to execute the internal Grover routine with a lower number of repetitions of the Grover operator, so shorter quantum circuits are required. To obtain the same final value of the cost function, 17 iterations of

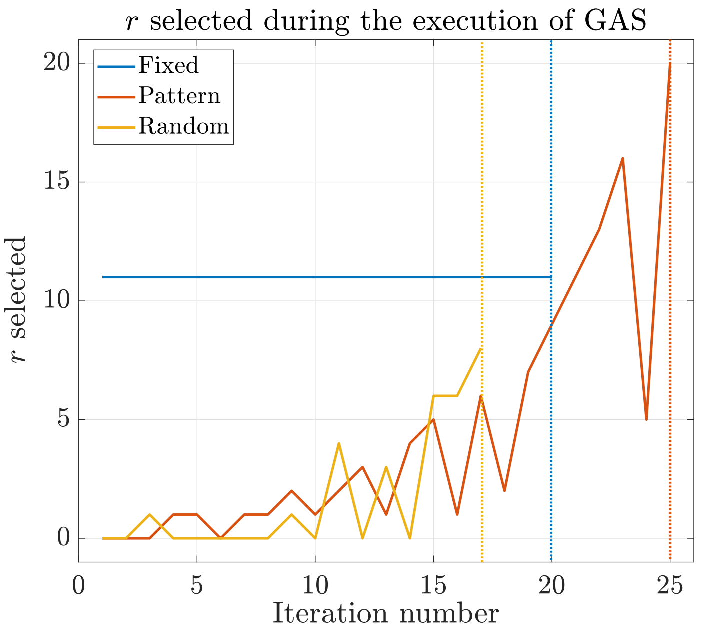


Figure 5.5: Evolution during the execution of r for different strategies

the sample-shift procedure are required with the random strategy and the Grover's algorithm is executed at most with 8 iteration of the Grover's operator, while 24 iterations with the pattern method are required and the maximum r in the pattern is 20, while the fixed r method requires 20 iterations with a fixed r of 11 (Table 5.1).

To visualize this result, Figure 5.5 reports the selected r at every execution of the Grover's algorithm during the Grover Adaptive Search, on the x axis there is

	r_{max}	total iterations
r random	8	17
r fixed	11	20
r fixed pattern	20	20

Table 5.1: Comparison between the maximum required r and total GAS iterations for different r selecting policies

the index of the considered iteration, while on the y axis there is the corresponding r. Dashed lines mark the moment in which the algorithm stops.

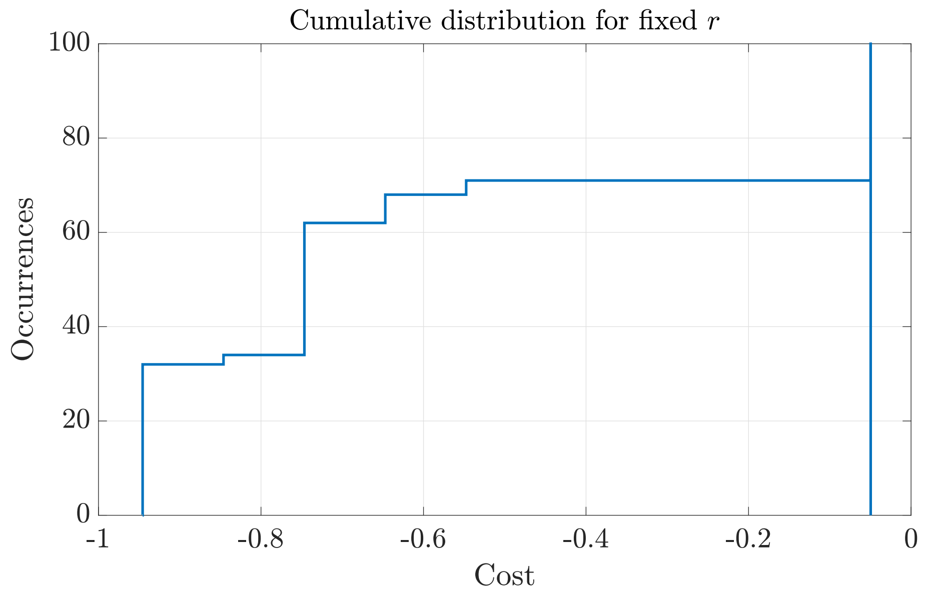


Figure 5.6: Cumulative distribution for the fixed r method

Figure 5.5 remarks how the random selection method has better performance in terms of complexity of the required quantum circuit and requires fewer iterations of the complete procedure. Figure 5.6, Figure 5.7 and Figure 5.8 report the cumulative distributions of the three analysed methods.

It is possible to notice that the random method permits to achieve a slightly higher probability of obtaining lower results then the fixed pattern. In addition to this, it needs fewer iterations and shorter quantum circuits. While the fixed r has proven to be the worst one even under this point of view.

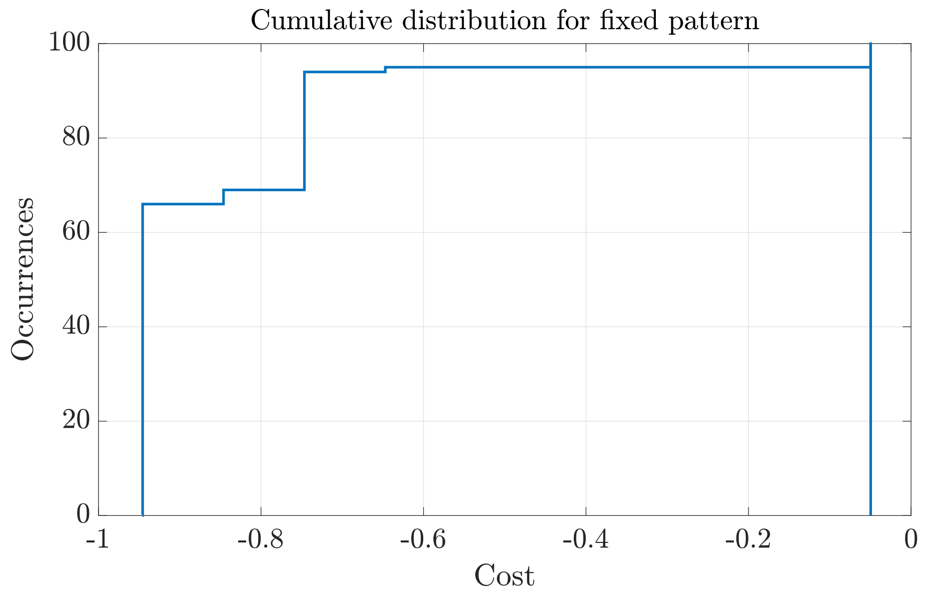


Figure 5.7: Cumulative distribution for the fixed pattern method

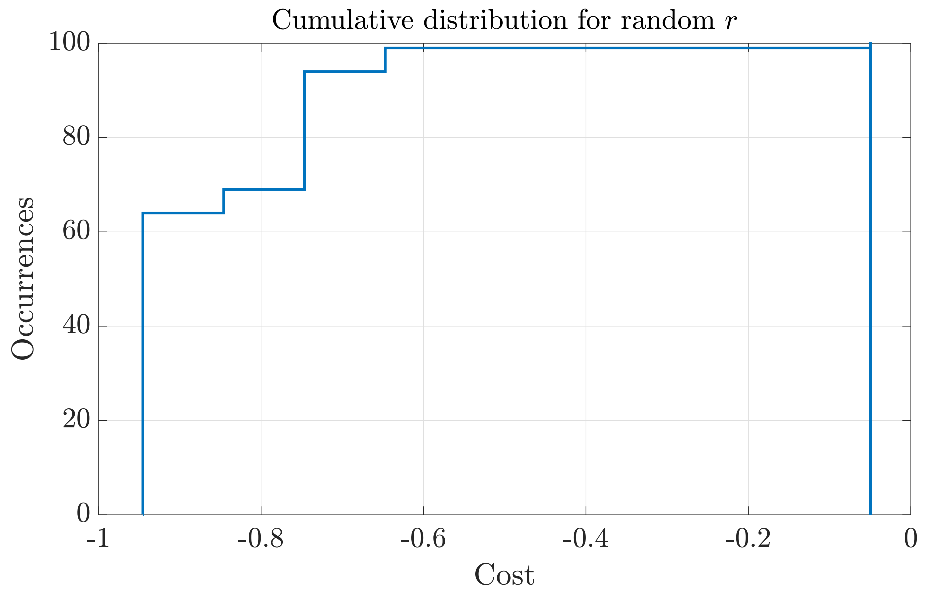


Figure 5.8: Cumulative distribution for the random r method

5.2 Linear regression results

To prove the effective functionality of the algorithm, a quantitative example is provided. The problem analysed before is used as example, in particular, $t = 7$ and the random r selection method are selected.

Figure 5.9 reports the plot of the dataset used to produce the model (blue points) and the corresponding line.

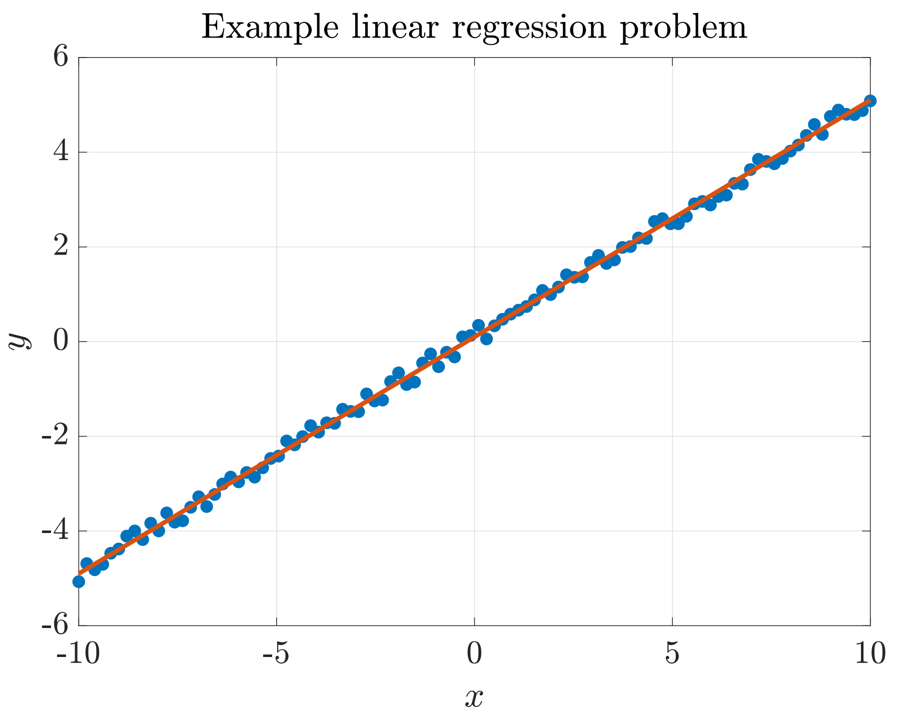


Figure 5.9: Linear regression problem representation

In order to avoid a too high number of qubits, a precision vector with six entries is chosen, so that the model contains twelve binary variables, so the quantum register with the keys of the quantum dictionary is composed of twelve qubits. In order to estimate the number of qubits for the value register of the quantum dictionary, the positive weights of the cost function are added. For the problem under test, six qubits are enough to handle the values of the function. In conclusion, considering also the ancillary qubit, nineteen qubits are required to encode the function.

The precision vector P is:

$$P = \left[-2^0, -2^{-1}, -2^{-2}, 2^{-2}, 2^{-1}, 2^0\right] \tag{5.1}$$

Figure 5.10 plots the cost function of the QUBO problem associated with linear regression.

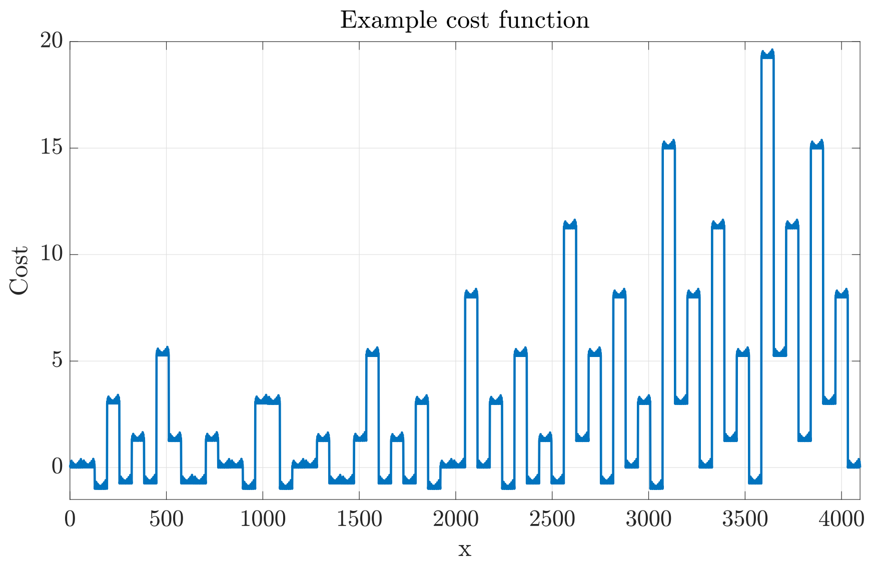


Figure 5.10: Cost function of the example problem

The solution vector $\hat{w}$ is:

$$\hat{w} = [1,0,0,0,1,1,0,1,0,0,1,0]^T \tag{5.2}$$

This $\hat{w}$ corresponds to a value equal to -0.9954, that is the absolute minimum of the cost function. The linear regression problem solution obtained with $\hat{w}$ is:

$$w = [0.5,0]^T \tag{5.3}$$

This a reasonable result, considering that it is not possible to obtain the real intercept (0.1) with the employed precision vector. In order to overcome this limitation, more qubits should be utilized. The offset error introduced by this is appreciable by Figure 5.11.

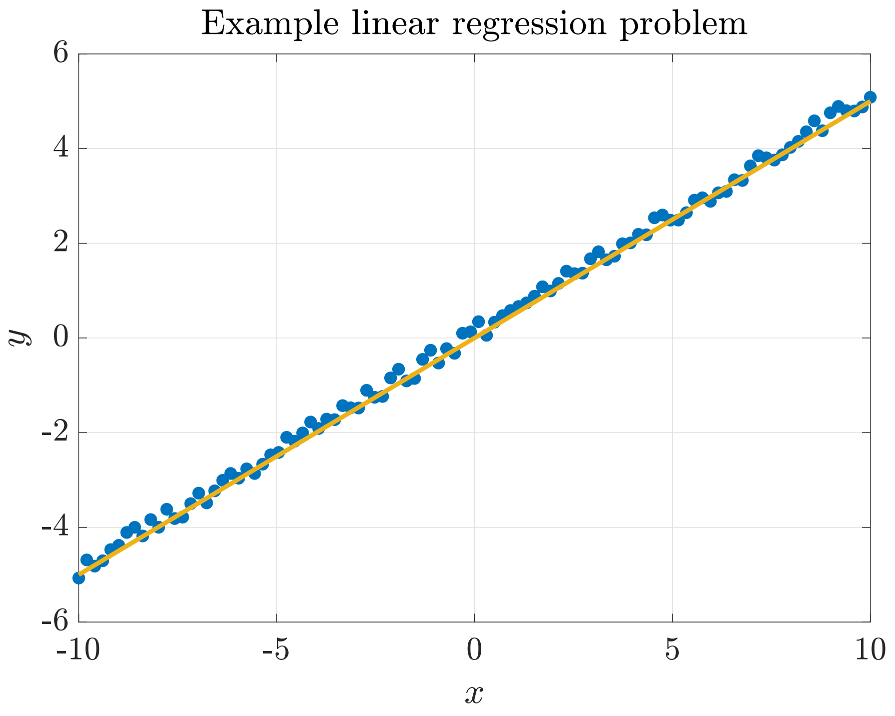


Figure 5.11: Linear regression problem representation and solution

Chapter 6

Comparisons

This chapter contains comparisons between the results provided by Grover Adaptive Search, simulated annealing and quantum annealing. Firstly, the results obtained with the three algorithms are presented singularly, then they are compared.

The empirical cumulative distribution of the obtained cost function value over different repetitions of the algorithm is used as a figure of merit to compare the performance.

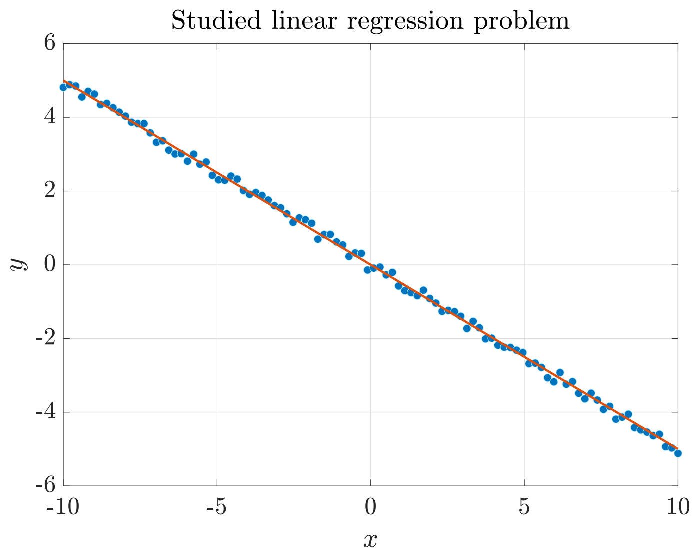


Figure 6.1: Reference linear regression problem submitted to different algorithms

Algorithms have been tested and compared on different linear regression problems. Here, for the sake of readability, just the results related to the problem reported in Figure 6.1 are presented. The dataset for the analysed problem is obtained by applying white noise on samples of a straight line with slope equal to -0.5 and offset equal to 0.

To do so an interface that, given a dataset, produces the quadratic unconstrained optimization problem to find the interpolating line has been produced, this outputs a Qubovert [31] model, that is then passed to the solver under test. The QUBO model associated to the presented problem has 12 binary variables, that is the maximum dimension solvable with the simulation of the GAS algorithm.

6.1 Simulated annealing

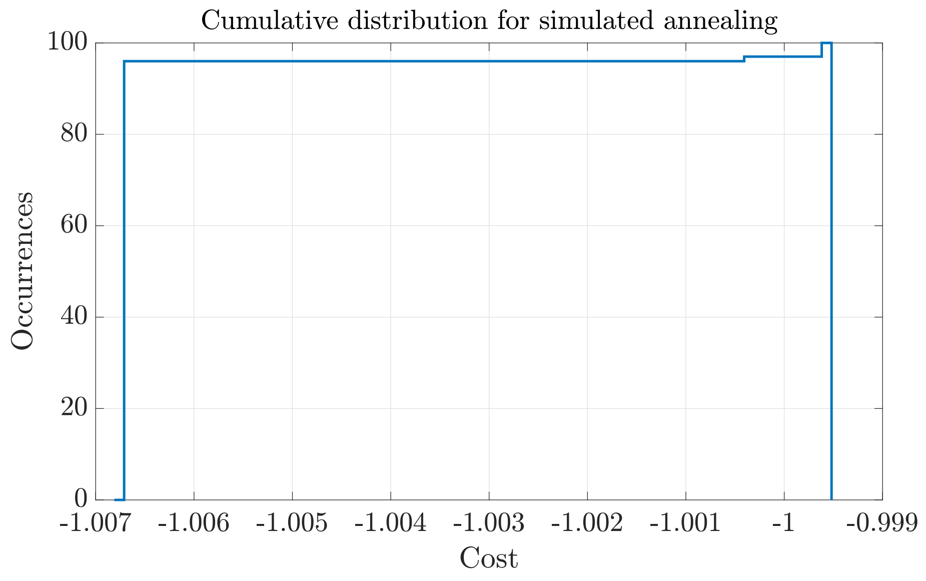


Figure 6.2: Cumulative distribution of cost for simulated annealing

The simulated annealing solver available in the Qubovert library is employed to solve the considered QUBO problem, it is possible to select the annealing time, obviously, a longer annealing time gives better results.

Figure 6.2 reports a plot of the cumulative distribution obtained solving the problem with this method, fixing the annealing time parameter of the solver to 1000, as it is possible to notice, it is a quasi-ideal distribution, indeed the probability of finding a solution with a cost near to the absolute minimum of the function (success probability) is 96%.

The main problem about this technique resides in the fact that to obtain good results, high annealing times are required and complexity and computational time grow fast [33]. Moreover, problems with many variables require to explore a big solution domain that grows exponentially. In addition to this, the mechanism to escape local minima, i.e. thermal perturbation, is not very effective in presence of high and narrow peaks.

Cumulative distributions of two executions with different annealing time are reported in Figure 6.3. The one with longer annealing time has a visible better behaviour.

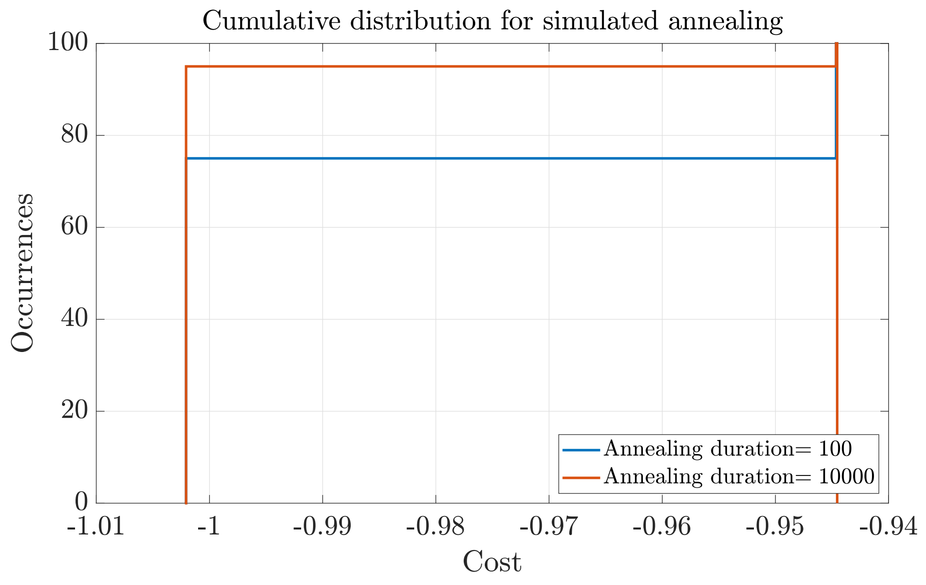


Figure 6.3: Cumulative distribution of cost for simulated annealing with different annealing durations

From Figure 6.3, it is possible to notice that the probability of obtaining the optimal solution increases by 20%, increasing the annealing duration from 100 to 10000.

6.2 Quantum annealing

The same problem was solved by a D-Wave's quantum annealer [27]. This is accessible via cloud by using the **Ocean SDK** [34], which permits to submit a QUBO problem to a quantum annealer, encapsulating the conversion to Ising model and relative mapping to the quantum annealer hardware.

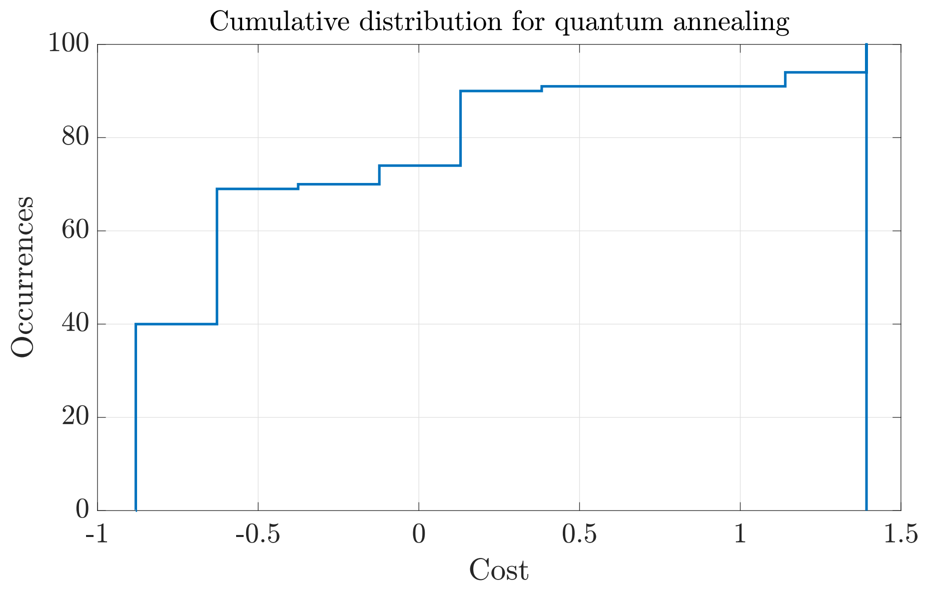


Figure 6.4: Cumulative distribution of cost for quantum annealing

Figure 6.4 reports the cumulative distribution obtained by using quantum annealing approach. It has a lower success probability than the simulated annealing, but some considerations are necessary.

First, quantum annealing runs on real quantum hardware, which introduces non-idealities, as fluctuations and errors. Moreover, the cost function of the considered problem (Figure 6.5) corresponds to an energy profile that is not particularly compliant to quantum annealing. In fact, the energy profile is composed by about flat regions (Figure 6.6) and sudden jumps, while the most suitable one for quantum annealing exploration presents high and narrow peaks, as reported in the Master Thesis [23].

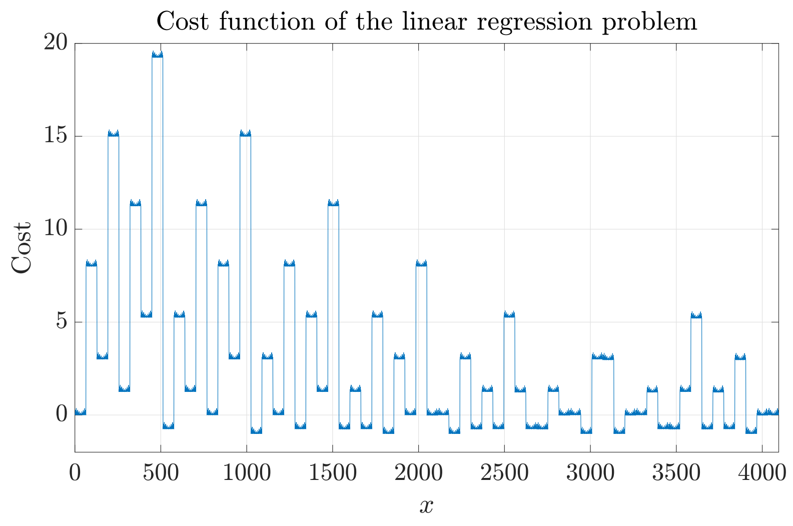


Figure 6.5: Cost function of the submitted problem

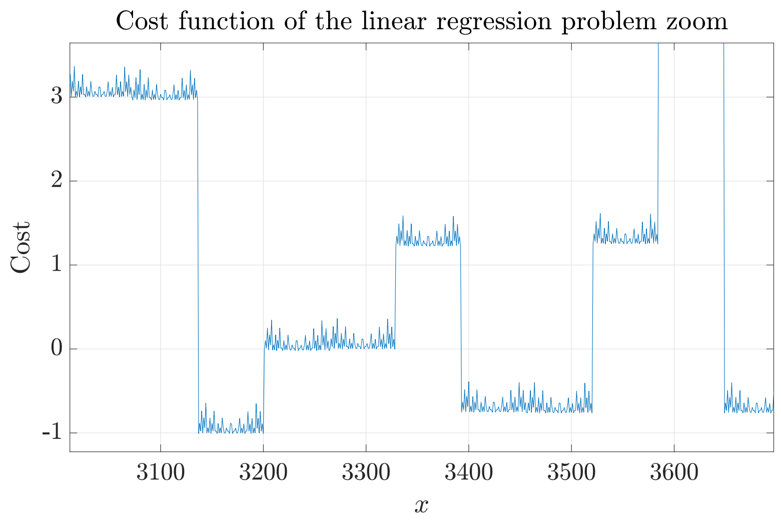


Figure 6.6: Zoom of the cost function of the submitted problem

Quantum annealing undoubtedly introduces an advantage in solving QUBO problems [35] with respect to simulated annealing, indeed it permits to find faster a solution, but, facing with the complexity of parameters' tuning and related problems deriving from badly-tuned parameters [36], introduce a penality too high.

6.3 Grover Adaptive Search

Grover Adaptive Search algorithm gives promising results. Figure 6.7 reports the cumulative distribution of the GAS algorithm with $t = 10$ and a random selection method for r.

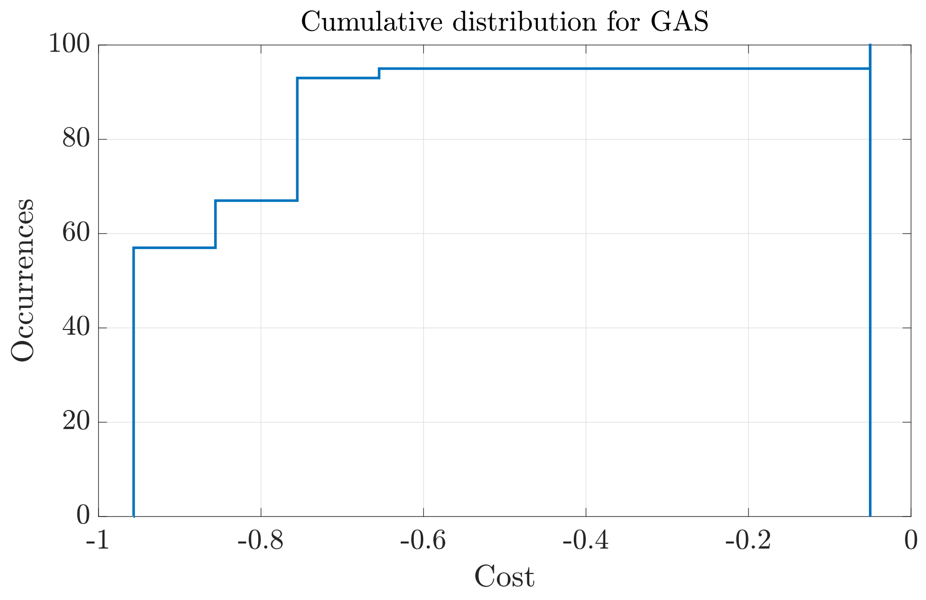


Figure 6.7: Cumulative distribution of cost for GAS

As it is possible to notice, there is a 100% probability of obtaining a negative value of the cost function, indeed, as mentioned in Chapter 4, this algorithm excludes positive values of the cost function, even if the aleatory nature of it naturally includes the fact that ideally every negative number could be accepted as a valid solution, tuning properly the parameters, in particular t, permits to avoid problems related to early acceptance of suboptimal results.

Furthermore, the Grover Adaptive Search algorithm permits to solve PCBO problems, that are a generalization of QUBO problems.

Figure 6.8 reports cumulative distributions of Grover Adaptive Search for different values of t. It is interesting to notice how little increments of this parameter, so few more repetitions of the Grover algorithm, can lead to an increment of some tens of percentage points in the probability of obtaining a value nearer to the absolute minimum.

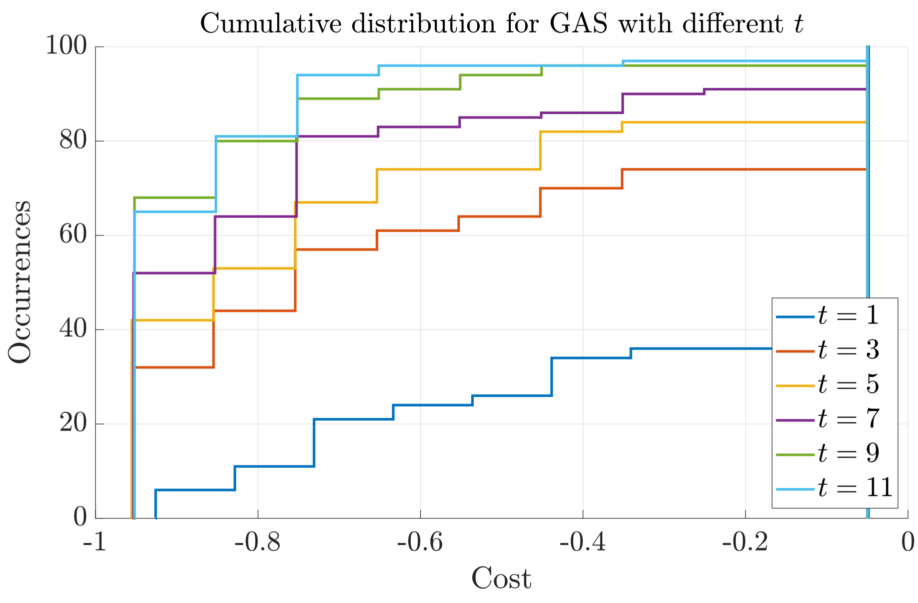


Figure 6.8: Cumulative distribution of cost for GAS at different values of t

Execution on real quantum hardware should be performed to effectively compare results with quantum annealing, but this is not possible due to the limited amount of qubits available on the accessible one [32]. Since the presented results are obtained with ideal simulations, they do not suffer from problems related to noise and quantum errors.

6.4 Conclusions

In conclusion, the Grover Adaptive Search algorithm is an optimization methodology attractive under different points of view.

Figure 6.9 highlights how, with the caveat of ideality of simulations, it presents higher probabilities of producing a value near to the absolute minimum than quantum annealing, at least for the tested class of problems. Simulated annealing gives always better results, but due to computational reasons, GAS can be preferred even to classical methods since the solution associated with the values found by GAS is still a valid solution to the linear regression problem under test. Moreover, since

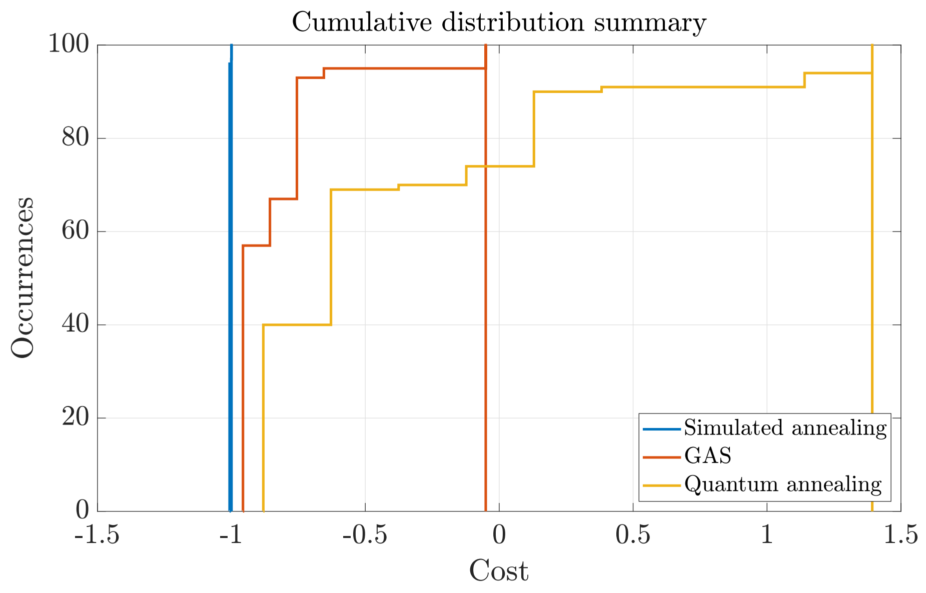


Figure 6.9: Cumulative distributions of cost for different algorithms

GAS follows the quadratic speed-up introduced by Grover Search [5], it is expected that the execution of this hybrid quantum-classical algorithm can reduce the total execution time with respect to simulated annealing, thus partially compensating the gap in terms of probability of obtaining a value close to the absolute minimum.

Chapter 7

Conclusions and future perspectives

In this chapter, final conclusions are reported and some future perspectives are suggested.

7.1 Future perspectives

The Grover Adaptive Search algorithm can be further improved by tuning differently the algorithm parameters, evaluating other stop conditions and acting on the cost function.

7.1.1 r and t optimization

As shown in Chapter 5, t and r are two crucial parameters of the implemented algorithm, so exploring other strategies with respect to the ones proposed could provide some improvements.

One possible alternative could be to identify statistically the best parameters couple for each class of problems. To do this, it is required to test different combinations of parameters with each class of problems and to compare the obtained results. The training procedure could be also executed by a neural network, which classifies the QUBO problem to be solved and consequently sets the most suitable parameters for solving it.

Moreover, other strategies to select r could be tested. For example, it could be interesting to test an alternative strategy to the random one, where the r parameter (initialized to 1) is increased until a negative value is sampled. This could be a good strategy considering that, g_r has a sinusoidal, i.e. periodic, behaviour with respect to r (Equation (2.56) and Figure 2.21 highlight this) so it is assured that the maximum value of g_r is achieved by exploring linearly the r domain.

7.1.2 Stop policy

The proposed stop policy could be improved, requiring a lower threshold, if it is considered to execute a further Grover's search, changing the Grover's oracle, after a certain number of positive values. In particular, the oracle could look for positive samples, exploiting the Grover's properties, which amplifies probabilities of non-solutions when more than half of the item set is labelled by the oracle. In this way, if few negative values of the cost function are available, they are detected.

Another strategy could be to use an oracle for detecting values lower or equal to zero. If the negative sample detected in the last iteration is measured, there is a high probability that this is the only item satisfying the oracle condition, so it is the optimal solution.

A more drastic approach to find if the algorithm has found the minimum could be to evaluate the probability distribution of the output state or to use the quantum counting algorithm [12] periodically to know how many zero values are present in the cost function.

7.1.3 Cost function pre-processing

As underlined in Section 2.4.6, the statistical characteristics of the searching domain, so the distribution of the values of the cost function, are crucial for a successful execution of the Grover algorithm. An effective improvement can be introduced by the definition of an efficient strategy to find an offset to be applied to the cost function, this permits to change its statistic and to start the algorithm with some negative values to mark. The same preprocessing could be used to better estimate the number of qubits of the values register of the quantum dictionary (Section 2.4.4) to avoid overflow.

A brute force approach for reaching these purposes is clearly unfeasible, so it is required to exploit strategies which sample some points of the function or evaluate the cost function coefficients.

7.2 Conclusion

This thesis proves how the Grover Adaptive Search algorithm could be a promising quantum approach for solving QUBO problems.

However, optimization is currently limited with respect to quantum annealing or other solvers, because of the limitations of quantum-gate-array hardware (in terms of total number of qubits) and of classical simulators (in terms of amount of memory required for simulation).

Figure 7.1, from the Yole report on quantum technology [6], reports a comparison between the roadmaps of different industries in the sector. It is possible to notice how the gate-array based quantum hardware is believed to scale faster the number of qubits than quantum annealers. Therefore, it is important to explore a quantum-gate-array based approach for optimization, like the Grover Adaptive Search algorithm, in the expectation that this general-purpose quantum computing paradigm could be the most effective quantum approach for solving complex optimization problems, with the hope that it will have in future positive resonances on Society and Industry.

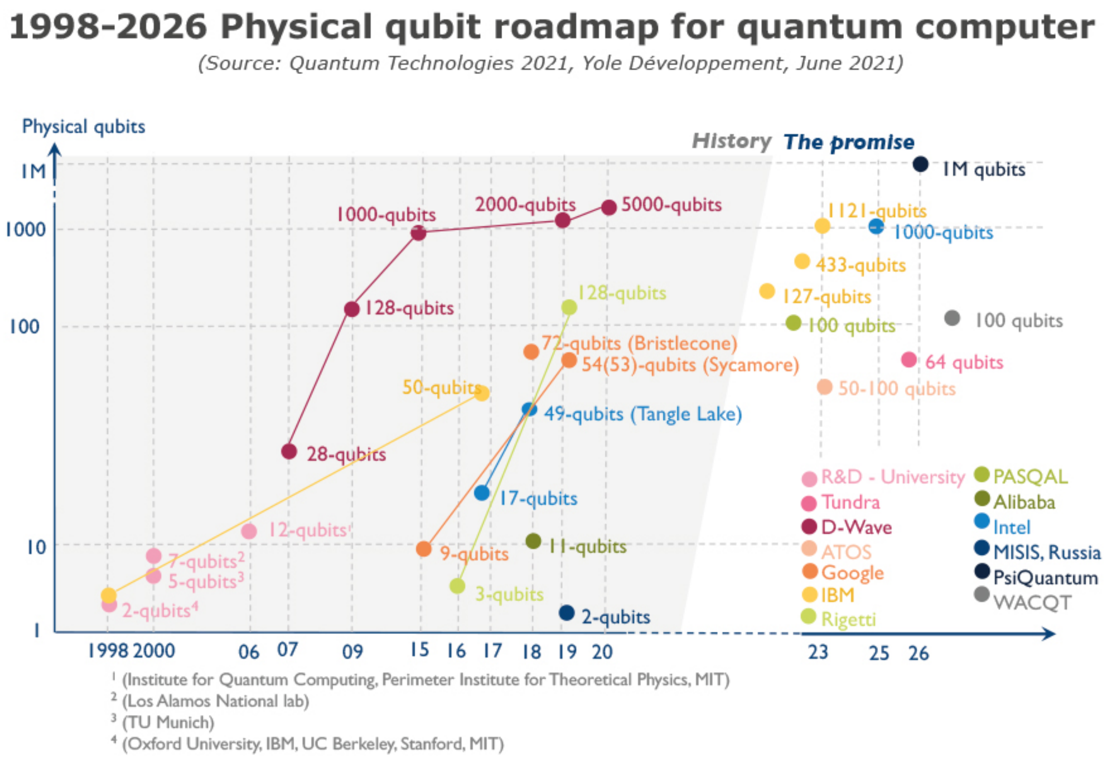


Figure 7.1: Quantum related companies roadmap from the Yole report on quantum technology [6]

Bibliography

[1] J. Groß. *Linear Regression*. Lecture Notes in Statistics. Springer Berlin Heidelberg, 2012. ISBN: 9783642558641. URL: https://books.google.it/books?id=enwQBwAAQBAJ.

[2] Imran Naseem, Roberto Togneri, and Mohammed Bennamoun. “Linear Regression for Face Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.11 (2010), pp. 2106–2112. DOI: 10.1109/TPAMI.2010.128.

[3] Robin J. Leatherbarrow. “Using linear and non-linear regression to fit biochemical data”. In: *Trends in Biochemical Sciences* 15.12 (1990), pp. 455–458. ISSN: 0968-0004. DOI: https://doi.org/10.1016/0968-0004(90)90295-M.

[4] Prasanna Date and Thomas Potok. “Adiabatic quantum linear regression”. In: *Scientific Reports* 11.1 (Nov. 2021). DOI: 10.1038/s41598-021-01445-6.

[5] Lov K. Grover. “Quantum Mechanics Helps in Searching for a Needle in a Haystack”. In: *Phys. Rev. Lett.* 79 (2 July 1997), pp. 325–328. DOI: 10.1103/PhysRevLett.79.325.

[6] Eric Mounier. *Quantum Technologies: Market and Technology Report 2020*. 2020. URL: https://tinyurl.com/2p9ecw3d.

[7] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. DOI: 10.1007/bf02650179.

[8] S. Leontica, F. Tennie, and T. Farrow. “Simulating molecules on a cloud-based 5-qubit IBM-Q universal quantum computer”. In: (June 2021). DOI: 10.1038/s42005-021-00616-1.

[9] Ekert. “Quantum cryptography based on Bell’s theorem”. In: *Physical review letters* 67 6 (1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661.

[10] Yangyang Li et al. "Quantum Optimization and Quantum Learning: A Survey". In: *IEEE Access* 8 (2020), pp. 23568–23593. DOI: 10.1109/ACCESS.2020.2970105.

[11] Noson S. Yanofsky and Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008. DOI: 10/ghbtw5.

[12] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[13] W.K. Wootters and W.H. Zurek. "A Single Quantum Cannot Be Cloned." In: *Nature* 299 (1982), pp. 802–803. DOI: 10.1038/299802a0.

[14] Manfredi Avitabile. "Proposal for a multi-technology, template-based quantum circuits compilation toolchain." Politecnico di Torino, July 2021. URL: http://webthesis.biblio.polito.it/id/eprint/19223.

[15] Giovanni Amedeo Cirillo. *Oracle design*. Online tutorial. URL: https://github.com/giovanniamedeocirillo/qiskit-textbook/blob/master/ch-algorithms/advanced/oracle-design.ipynb.

[16] Austin Gilliam et al. "Foundational Patterns for Efficient Quantum Computing". In: (2021). arXiv: 1907.11513 [quant-ph].

[17] W. P. Baritompa, D. W. Bulger, and G. R. Wood. "Grover's Quantum Algorithm Applied to Global Optimization". In: (Jan. 2005). DOI: 10.1137/040605072.

[18] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003. URL: https://tinyurl.com/4efyh8hn.

[19] Fred Glover, Jin-Kao Hao, and Gary Kochenberger. "Polynomial unconstrained binary optimisation - Part 1". In: *International Journal of Metaheuristics* 1 (July 2011), pp. 232–256. DOI: 10.1504/IJMHEUR.2011.041196.

[20] Fred Glover, Gary Kochenberger, and Yu Du. "A tutorial on formulating and using QUBO models". In: *arXiv preprint arXiv:1811.11538* (2018).

[21] Prasanna Date, Davis Arthur, and Lauren Pusey-Nazzaro. "QUBO formulations for training machine learning models". In: *Scientific Reports* 11.1 (May 2021). DOI: 10.1038/s41598-021-89461-4.

[22] Davis Arthur and Prasanna Date. "Balanced k-means clustering on an adiabatic quantum computer". In: *Quantum Information Processing* 20.9 (Sept. 2021). DOI: 10.1007/s11128-021-03240-8.

[23] Deborah Volpe. "Software and Hardware Design of Digital Quantum Annealing Emulators." Politecnico di Torino, Oct. 2021. URL: http://webthesis.biblio.polito.it/id/eprint/20457.

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (May 1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[25] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". In: *Phys. Rev. E* 58 (5 Nov. 1998), pp. 5355–5363. DOI: 10.1103/PhysRevE.58.5355.

[26] Jeffrey Chang. *The Ising model.* URL: https://stanford.edu/~jeffjar/statmech/intro4.html.

[27] *D'wave cloud resource.* URL: https://www.dwavesys.com/solutions-and-products/cloud-platform/.

[28] David W. Bulger, William Baritompa, and Graham R. Wood. "Implementing Pure Adaptive Search with Grover's Quantum Algorithm". In: *Journal of Optimization Theory and Applications* 116 (2003), pp. 517–529. DOI: 10.1023/A:1023061218864.

[29] Austin Gilliam, Stefan Woerner, and Constantin Gonciulea. "Grover Adaptive Search for Constrained Polynomial Binary Optimization". In: (Apr. 2021). DOI: 10.22331/q-2021-04-08-428.

[30] Amian S. Steiger, Thomas Häner, and Matthias Troyer. "ProjectQ: an open source software framework for quantum computing". In: (Jan. 2018). DOI: 10.22331/q-2018-01-31-49.

[31] Joseph T. Iosue. *Qubovert.* Python package. URL: https://github.com/jtiosue/qubovert.

[32] *IBM Quantum*. 2021. URL: https://quantum-computing.ibm.com/.

[33] Hansen Per Brinch. “Simulated Annealing”. In: *Electrical Engineering and Computer Science - Technical Reports* (1992). URL: https://surface.syr.edu/eecs_techreports/170.

[34] D’wave. *Ocean SDK*. Python package. URL: https://docs.ocean.dwavesys.com/en/stable/index.html.

[35] Sei Suzuki. “A comparison of classical and quantum annealing dynamics”. In: *Journal of Physics: Conference Series* 143 (Jan. 2009), p. 012002. DOI: 10.1088/1742-6596/143/1/012002.

[36] D’wave. *Programming the D-Wave QPU: Parameters for Beginners*. Whitepaper. URL: https://www.dwavesys.com/media/qvbjrzgg/guide-2.pdf.