UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Matematica “Tullio Levi-Civita”
Dipartimento di Fisica e Astronomia “Galileo Galilei”

Corso di Laurea Magistrale in Matematica

# A quantum approach to LP-induced unique sink orientation problems

Supervisor:

Prof. Simone Montangero

Co-supervisor:

Prof. Andris Ambainis

Student: Michele Cattelan

Matricola 1237577

Academic Year 2020/2021
23rd July 2021

# Acknowledgements

First and foremost, I would like to thank professor Andris Ambainis and professor Simone Montangero for supporting me in my research and in my mobility period during this pandemic. I feel lucky to have worked with them. I am also grateful to them for giving me the opportunity of writing my Master's thesis in the topic I am interesting in.

Special thanks to Evgeny Vihrov for joining our research and for the productive weekly meeting we had.

# Contents

# Introduction

Linear programming is the problem of finding the maximum or the minimum of a given linear function, called *objective function*, under linear constraints, both equalities and inequalities. A linear program (LP) can be stated as

$$\begin{aligned} \text{min (max)} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq \mathbf{0}, \end{aligned}$$

where **0** is the zero vector of $\mathbb{R}^n$, $c^T x$ is the objective function, with $c \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$, and where $Ax = b$ and $x \geq \mathbf{0}$ [1] are the *constraints*, with $A \in \mathbb{R}^{m \times n}$. From now on we concentrate on the maximization (max) case, the other one is analogous. Geometrically, the constraints can be represented as a polytope. Any vectors that belong to the constraints polytope are called *solutions*. The goal of linear programming is to find a solution $x^*$ that maximize the objective function. This solution is said to be *optimal*. Depending on the solutions, linear programs can be classified. If the constraints polytope is non empty, then the linear program is said to be *feasible*, otherwise it is called *infeasible*. A simple example of feasible linear program is the following

$$\begin{aligned} \text{max} \quad & x_1 \\ \text{s.t.} \quad & x_1 + x_2 = 1 \\ & x_1, x_2 \geq 0, \end{aligned}$$

where clearly the constraints polytope defined is non empty and the optimal solution is the vector $x^* = (1, 0)$. On the other hand, to be infeasible means

[1] Hereafter the comparison between two vector is element-wise.

that the constraints have no solutions, for example,

$$\begin{aligned} \text{max} \quad & x_1 \\ \text{s.t.} \quad & x_1 + x_2 = 1 \\ & x_1 + x_2 = -1 \\ & x_1, x_2 \geq 0. \end{aligned}$$

If in the feasible case the objective function is bounded over the constraints polytope, the program is called *bounded*, otherwise it is said to be *unbounded*. The feasible program above is bounded because it has an optimal solution and because clearly the objective function is bounded, $x_1 \leq 1$. A trivial example of an unbounded program is the maximization on the line

$$\begin{aligned} \text{max} \quad & x_1 \\ \text{s.t.} \quad & x_1 \geq 0. \end{aligned}$$

Indeed, we can notice that it is feasible but for every solution $\bar{x}$ of the program there always exists a solution $\tilde{x}$ such that $\tilde{x}_1 \geq \bar{x}_1$. Therefore a feasible program is unbounded when there is not an optimal solution. The widely known method for solving LP is the *Simplex Method*. Despite it is used in practice for solving linear programs, in the worst cases it is not an efficient algorithm. In fact, it has been proven that for special linear programs the computational time of the simplex method is exponential [9]. Many studies focused on finding a polynomial-time algorithm for solving linear programs and to date different efficient algorithms are available [14][10][8]. Although linear programming is known to be in the class of polynomial-time problems **P**, no *strongly polynomial-time* algorithm[2] is known. A possible direction for finding a strongly polynomial algorithm for linear programming is by means of solving a *unique sink orientation problem* on an hypercube. A unique sink orientation on an hypercube is a directed hypercube where all its faces, which are subgraphs that are hypercubes, have a unique sink. Therefore the hypercube, being a face of itself, has a unique global sink. It has been shown by Gärtner and Schurr [6] that for any linear programs in standard form with $n$ variables we can define an equivalent unique sink orientation on an $n$-hypercube, where the global unique sink of the cube represents the solution of the problem. The idea is to exploit the equivalence between USO and a certain class of strictly

[2] a polynomial-time algorithm in the RAM model.

convex programs, i.e. constraints optimization problems where the function is strictly convex and the constraints are linear equalities or inequalities. By perturbing any LP to a family of strictly convex programs parameterized by $\epsilon > 0$, one can obtain a sequence of USO that depends on $\epsilon$. The limit orientation obtained from letting $\epsilon$ tend to zero is a USO, which is called LP-induced. In *Figure* 1 it is shown an example of an LP and its LP-induced USO. By finding the sink of the LP-induced USO, indeed, it can determine whether the program is infeasible, unbounded or feasible. In the latter case it is also possible to compute the optimal solution. For solving a unique sink orientation problem we need a function, called oracle, because in such a problem we do not know the edge orientations a priori and the only way we can access to this information is via the oracle. The oracle takes as input a vertex and outputs a list of the directions of the outgoing edges of that vertex. Therefore the algorithm complexity is computed with the query complexity, i.e. the number of times we use the oracle. Until now, no efficient algorithms for finding the unique sink in such frameworks are known. The best known algorithm queries to the oracle $O\left((1,467\ldots)^n\right)$ times, where $n$ is the dimension of the hypercube taken into account.

$$\begin{array}{rc} \text{max} & x_1 \\ \text{s.t.} & x_1 + x_2 = 1 \\ & x_1, x_2 \geq 0 \end{array}$$

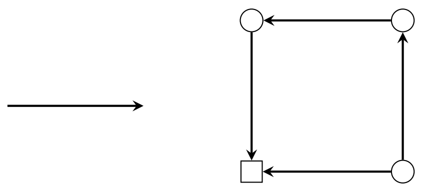


Figure 1: Example of a linear program and its unique sink orientation on hypercube equivalent problem. The $\square$ node is the unique sink of the $2$-hypercube. The arrow represents the procedure needed to define the orientation on the hypercube.

In this thesis we present a quantum approach to the unique sink orientation problem on hypercubes. Indeed, quantum computation outperforms the classical one in different mathematical problems. One of the most known is Grover's search algorithm [7]. This algorithm searches for a specific element in an unstructured database. Both for the Grover's and the classical search we need an oracle that detects the element we are looking for. The Grover's search provides a quadratic speedup on query complexity over the classical search methods. In fact, if we are looking for an item in a $N$ elements database, the classical search needs, in average, $O(N)$ queries to the oracle while the Grove's algorithm just $O(\sqrt{N})$. Moreover, Grover's search algorithm is important for our purpose because it outperforms the best classical algorithm for solving the unique sink orientation problem. Note that a trivial application of Grover's search to the general unique sink orientation problem yields a quantum algorithm with query complexity $O(\sqrt{2^n}) = O((1,141\ldots)^n)$, where $2^n$ are the vertices of the $n$-hypercube. Therefore Grover's algorithm just marginally beats the best known classical algorithm. Furthermore, our interest in the unique sink orientation problem arises from a recent result by Bacon [1]. The author showed that it is possible to find an efficient quantum algorithm for solving unique sink orientation problems if it is possible to define an efficient specific function based on the oracle. However, the author cannot provide such a function and, in general, finding it is an hard problem since we need to be able to compute efficiently the composite function $f^k$, where $f$ is the oracle, for any $k \in \{1, \ldots, 2^n - 1\}$. So the problem is still open.

In this thesis are presented different ways to attack USO problems with respect to that proposed by Bacon. Two quantum algorithms that exploit different geometrical and combinatorial properties of unique sink orientations on hypercubes are proposed. This particular properties can be used efficiently only on a quantum computer. In fact, in a nutshell, these algorithms take into account all the vertices of the hypercube (which are $2^n$ in the $n$-dimensional case) at once. In a classical computer implementing a list of $2^n$ elements requests an exponential number of resources, while in a quantum computer we need only $n$ qubits (the quantum counterpart of bits).

The structure of this thesis is the following: we first introduce the basic notions of linear programming and of quantum computation theory. Then, we present the geometrical and combinatorial properties of the unique sink orientation on hypercubes and we show how to generate a unique sink orientation from a linear program. In conclusion, we present and benchmark the two quantum algorithms that we developed.

# Chapter 1

# Preliminaries

In this chapter we show the basic notions of linear programming and quantum computing that are necessary for our discussion.

## 1.1 Linear programming

A *linear program* (LP) is a constrained optimization problem where the function to be maximize, or minimize, is linear and the constraints are linear equalities or inequalities. A general linear program can be stated as:

$$
\begin{aligned}
\text{max} \quad & c_1x_1 + \cdots + c_nx_n & & (1.1)\\
\text{subject to} \quad & a_{11}x_1 + \cdots + a_{1n} \quad \sim b_1 & & (1.2)\\
& \vdots \qquad\qquad\qquad\quad \vdots & & \\
& a_{m1}x_1 + \cdots + a_{mn} \sim b_m, & & (1.3)
\end{aligned}
$$

where $(c_i) = c \in \mathbb{R}^n$, $(b_j) = b \in \mathbb{R}^m$ and $(a_{ij}) = A \in \mathbb{R}^{m\times n}$. The symbol "$\sim$" stands for "$\leq$", "$\geq$" or "$=$". Notice that strict inequalities are not allowed. In the general linear program above every $x_i$, $i = 1, \dots, n$, is called *variable* of the program, the linear function $c^Tx$, in (1.1), is the *objective function* and the expression from (1.2) to (1.3) are called the *constraints* of the program. We take into account only maximization programs because the other ones are analogous. In fact, it is really straightforward to pass from a maximization program to a minimization one: if we have as an objective function $f(x)$ to be maximized, then, for having a minimization program, we take the opposite of the solution of the program which has as objective function the opposite function, $-f(x)$, and has the same constraints. Therefore “max $f(x)$" is equivalent to “$-$min $-f(x)$", under the same constraints. We refer to a general linear program in the following compact form

$$\begin{aligned} \text{min} \quad & c^T x \\ \text{s.t.} \quad & Ax \sim b. \end{aligned}$$

Every vectors $x \in R^n$ which satisfies the expressions $Ax \sim b$ is called *feasible solution* of the program. The correspondent value $c^T x$ is called *value* of the feasible solution $x$. The set of all the feasible solutions is called *feasible region*. A vector $x^* \in \mathbb{R}^n$ is said to be an *optimal solution* of the linear program whether $c^T x^* \geq c^T x$ for any vector $x$ that belongs to the feasible region. And finally, the *optimal value* is the value associated to an optimal solution.

What is important to stress in linear programming is the *Fundamental theorem of linear programming*, which gives a description of all the possibilities that can occur in the analysis of a linear program.

**Theorem 1.1.** *Given a linear program, only one of the following holds:*

*(i) the problem has at least an optimal solution.*

*(ii) the problem has no feasible solution and, then, it is said to be* infeasible*.*

*(iii) the problem is* unbounded*, that is, for any $\alpha \in \mathbb{R}$ there exists a feasible solution $x$ such that $c^T x \geq \alpha$ ( $c^T x \leq \alpha$ in the minimization case).*

This result says that it is impossible for the problem to have a infimum (or supremum in the maximization case) that is not also a minimum (or maximum respectively).

Let us show, now, that we are able to write any linear program in a more suitable way. In fact, without loss of generality we can study a program in a particular form, called *standard form*:

$$\begin{aligned} \text{min} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq \mathbf{0}, \end{aligned}$$

where with the vector notation $x \geq \mathbf{0}$ we are saying that $x_i \geq 0$ for any $i = 1, \ldots, n$.

First of all, we specify what it means to be equivalent for two linear programs and then explain why it is possible to find this form for every linear program. Therefore, given $P_1$ and $P_2$ two maximization linear programs we would say that they are *equivalent* if for any $\alpha \in \mathbb{R}$, $P_1$ has a feasible solution of value $\alpha$ if and only if $P_2$ has a feasible solution of value $\alpha$. Moreover, if $P_1$ and $P_2$ are equivalent it is easy to see that

(a) $P_1$ has a feasible solution if and only if $P_2$ has a feasible solution.

(b) for any $\alpha \in \mathbb{R}$, $P_1$ has an optimal solution value $\alpha$ if and only if $P_2$ has an optimal solution of value $\alpha$.

(c) $P_1$ is unbounded if and only $P_2$ is unbounded.

**Lemma.** *Every linear program is equivalent to a program in standard form.*

*Proof.* We say that a variable $x_i$ is *non negative* whether $x_i \geq 0$, otherwise it is said to be *free*.

Let us start with the procedure for deriving the standard form program from a generic program written like (1.1)-(1.3). First of all, every inequality constraint $a_i^T x \geq b_i$, where $a_i$ is the $i$-th row of $A$, is equivalent to the inequality $-a_i^T x_i \leq -b_i$. Thus, every constraint, except the non negative ones, in the program is either of the type $a_i^T x \leq b_i$ or $a_i^T x = b_i$ .

Now, if we take into account the constraint $a_i^T x \leq b_i$ and we add a new non negative auxiliary variable $s_i$, called *slack variable*, then we can substitute the above inequality constraint in the program for

$$a_i^T x + s_i = b_i, \quad s_i \geq 0. \tag{1.4}$$

Notice that if $x$ satisfies the constraint $a_i^T x \leq b_i$ to equality then it means that $s_i = b_i - a_i^T x$. On the other hand, if both $x$ and $s_i$ satisfy the constraints (1.4), then $x$ satisfies the above constraint $a_i^T x \leq b_i$. Therefore, since the objective function has not been modified then the problem obtained is equivalent to the starting one.

Eventually, let us consider *free* variables, i.e. the ones which have not a constrained sign. So if $x_j$ is a free variable it means that it belongs to $\mathbb{R}$ and, thus, from a real number property we are able to rewrite it as a subtraction of two different non negative numbers. Therefore, we can substitute the variable $x_j$ for the two variables $x'_j$ and $x''_j$ by adding

$$x_j = x'_j - x''_j, \quad x'_j, x''_j \geq 0 \tag{1.5}$$

to the constraints. It is easy to compute $x_j$ from $x'_j$ and $x''_j$, and vice versa.

Therefore we found a linear program in standard form equivalent to the starting one. □

We now introduce the dual theory of linear programming which would be crucial in *Chapter* 3 for understanding the equivalence between linear programming and unique sink orientation problems.

Given a LP:

$$P := \begin{cases} z^* = \max & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0, \end{cases}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ and $z^* := c^T x^*$ is the optimal value whether it exists. The goal of the duality theory is to find the best bound for the optimality, i.e. a real number $u \in \mathbb{R}$ such that $u \geq z^*$. Note that by giving an arbitrary vector $y \in \mathbb{R}^m$, every vector $x$, feasible for $P$, fulfills

$$y^T(Ax) = y^T b. \tag{1.6}$$

Moreover, by assuming that $Ay^T \geq c$, since $x$ is non negative we get the inequality $c^T x \leq (A^T y)^T x = (y^T A)x$. Therefore, by using equation (1.6) we obtain

$$c^T x \leq (A^T y)^T x = y^T(Ax) = y^T b = b^T y. \tag{1.7}$$

Finally, for any $y \in \mathbb{R}^m$ such that $A^T y \geq c$, every feasible $x$ for $P$ satisfies $c^T x \leq b^T y$. Thus, $z^* \leq b^T y$ and, therefore, for any suitable $y$ we have an upper bound of the optimal value $z^*$. Finding the smallest upper bound means solving

$$D := \begin{cases} d^* = \min & b^T y \\ \text{s.t.} & A^T y \geq c. \end{cases}$$

This last LP is called the *dual problem* of $P$, which is instead called *primal problem*.

**Theorem 1.2** (Duality weak theorem)**.** *If $x$ is a feasible solution of $P$ and $y$ is a feasible solution of $D$, then $c^T x \leq b^T y$.*

*Proof.* It follows form (1.7) since the first inequality comes from $A^T y \geq c$ and $x \geq 0$, while the second equality comes from $Ax = b$. □

This result yields to a characterization for the optimality of solutions.

**Corollary.** *Given $x^*$ feasible solution for $P$ and $y^*$ feasible solution for $D$, if $c^T x^* = b^T y^*$, then $x^*$ and $y^*$ are optimal solutions for their respective problems.*

*Proof.* If $x^*$ had not been optimal, then it would have existed another feasible solution $\bar{x}$ for the primal problem such that $c^T \bar{x} \geq c^T x^* = b^T y^*$, which would contradict the *Theorem* 1.2. Analogously, if $y^*$ had not been optimal, then there would have existed another feasible solution $\bar{y}$ of the dual problem such that $b^T \bar{y} < b^T y^* = c^T x^*$, which contradict the *Theorem* 1.2 as well. □

**Corollary.** *If either $P$ or $D$ is unbounded, then the other one is infeasible.*

*Proof.* Let us suppose that $D$ has a feasible solution $\bar{y}$. Then since *Theorem* 1.2 we have that $c^T x \leq b^T \bar{y}$ for each $x$ feasible for $P$, therefore the former cannot be unbounded. The other case is analogue. □

Finally, all of this considerations yield to the following theorem. We omit the proof for lightening the discussion.

**Theorem 1.3** (Strong duality theorem)**.** *If $P$ has an optimal solution $x^*$, then $D$ has an optimal solution $y^*$; moreover, $c^T x^* = b^T y^*$ holds.*

Moreover, from the duality theory we can also find out the following characterization for optimal solution.

**Theorem 1.4** (Complementary slackness theorem)**.** *Let $P$ be a LP in standard form and let $x^*$ and $y^*$ be feasible solution respectively of $P$ and of its dual $D$. Then $x^*$ and $y^*$ are optimal solution for their respective problems if and only if for any $j \in \{1, \dots, n\}$, $x_j^* = 0$ or $y^*$ satisfies to equality the correspondent dual constraint.*

*Proof.* The primal problem has the form

$$\begin{aligned} \text{max} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

while its dual

$$\begin{aligned} \text{min} \quad & b^T y \\ \text{s.t.} \quad & A^T y \geq c. \end{aligned}$$

Since the feasibility of $x^*$ and $y^*$ it follows that

$$c^T x^* \leq \left(A^T y^*\right)^T x^* = (y^*)^T A x^* = (y^*)^T b = b^T y^*.$$

Given Strong duality theorem, *Theorem* 1.3, both $x^*$ and $y^*$ are optimal for the respective problem if and only if $c^T x^* = b^T y^*$, that is, when in the expression above all the relations are equality. This happens if and only if $c^T x^* = (A^T y^*)^T x^*$ and, therefore, if and only if $\left(A^T y^* - c\right)^T x^* = 0$, or explicitly

$$\sum_{j=1}^{n} \left(A_j^T y^* - c_j\right) x_j^* = 0.$$

Since $x_j^* \geq 0$ and $A_j^T - c_j \geq 0$ for any $j$, this happens if and only if all terms of the sum are zero, i.e. if and only if for any $j \in \{1, \ldots, n\}$ at least one of $x_j^*$ and $A_j^T y^* - c_j$ is equal to zero. □

Eventually, we look for a general description of duality and a general statement for the complementary slackness theorem.

Therefore we present a method for computing the dual problem of a general linear program. We do not enter in the details, but the derivation is similar to the standard form case. For this method we take into account any variables and constraints separately, thus we refer to a general linear program as

$$\begin{aligned}
\text{max} \quad & c^T x & & \qquad (1.8)\\
\text{s.t.} \quad & a_i^T x \leq b_i, & i = 1, \dots, h & \qquad (1.9)\\
& a_i^T x \geq b_i, & i = h+1, \dots, k & \qquad (1.10)\\
& a_i^T x = b_i, & i = k+1, \dots, m & \qquad (1.11)\\
& x_j \quad \geq 0, & j = 1, \dots, p & \qquad (1.12)\\
& x_j \quad \leq 0, & j = p+1, \dots, q & \qquad (1.13)\\
& x_j \quad \in \mathbb{R}, & j = q+1, \dots, n & \qquad (1.14)
\end{aligned}$$

The dual problem have as many variables as many rows $A$ has and as many constraints as many variables the primal problem has. Moreover the following relations hold

| max $c^T x$ | min $b^T y$ | |
|---|---|---|
| $a_i^T x \geq b_i$ | $y_i \geq 0$ | $i = 1, \dots, h$ |
| $a_i^T x \geq b_i$ | $y_i \leq 0$ | $i = h+1, \dots, k$ |
| $a_i^T x = b_i$ | $y_i \in \mathbb{R}$ | $i = k+1, \dots, m$ |
| $x_j \geq 0$ | $A_j^T y \geq c_j$ | $j = 1, \dots, p$ |
| $x_j \leq 0$ | $A_j^T y \leq c_j$ | $j = p+1, \dots, q$ |
| $x_j \in \mathbb{R}$ | $A_j^T y = c_j$ | $j = q+1, \dots, n$ |

The above table can be read in both the ways, that is, we can use it for computing the dual of a maximization program (read from left to right) or of a minimization one (read from right to left).

Then the complementary slackness theorem become

**Theorem.** *Given a general linear program in the form (1.8)-(1.14) and the feasible solution $x^*$ and $y^*$ of the primal and the dual problem respectively, then $x^*$ and $y^*$ are optimal if and only if the following slackness conditions hold:*

- *For any $j \in \{1, \dots, n\}$, $x_j^* = 0$ or $y^*$ satisfy at equality the correspondent dual constraint.*
- *For any $i \in \{1, \dots, m\}$, $y_i^* = 0$ or $x^*$ satisfy at equality the correspondent primal constraint.*

We stress the fact that if a primal (dual) variable is free, then its correspondent dual (primal) constraint is an equality, therefore the complementarity conditions always hold.

## 1.2 Quantum computing theory

For understanding how quantum algorithms work, we give the basic notions of quantum computing and quantum information theory. In the classical computation the bit is the fundamental concept. Quantum computation is build on an analogous concept: the quantum bit, or *qubit*.

Qubits are two-level quantum systems described by a two dimensional Hilbert space where only two orthogonal and normalized states are taken into account: the fundamental state $|0\rangle$ and one excited state $|1\rangle$. Notation $'|\ \rangle'$ is called Dirac notation and it is a standard way to refer to quantum physical state[1]. A classical bit can be implemented on a classical hardware only in two different ways, the state $|0\rangle$ and the state $|1\rangle$. In the other hand, a qubit, by exploiting the property of superposition can be found in the state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ under the unitary constraint $|\alpha|^2 + |\beta|^2 = 1$. In *Figure* 1.1 we can see a concrete example of a qubit.

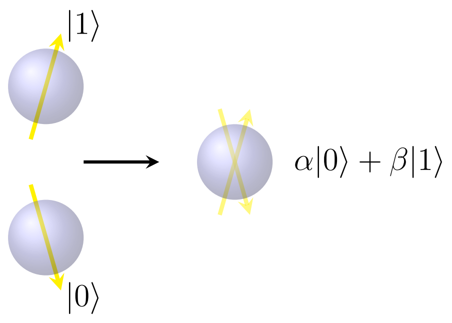


Figure 1.1: Example of a qubit: we can take into account an electron where only two energy levels are considered, the fundamental one (spin down) $|0\rangle$ and an exited one (spin up) $|1\rangle$. Starting from one of these we can create a superposed state, represented in the right part of the figure. We can think that the electron is in a superposed state whether it is both in the spin up and in the spin down state.

The states $|0\rangle$ and $|1\rangle$ are represented by vectors that form an orthonormal basis of the Hilbert space. This basis is also known as *computational basis*.

[1] For more details about this, look at the Appendix.

One picture useful in thinking about qubits is the following geometric representation. By starting from the normalization constraint$|\alpha|^2 + |\beta|^2 = 1$ we can see that

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + ie^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right),$$

where $\gamma, \theta$ and $\varphi$ are real numbers. Notice that $e^{i\gamma}$ is a global phase of a physical quantum system, therefore it has no physical meaning and can be neglected. Then we can write

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + ie^{i\varphi}\sin\frac{\theta}{2}|1\rangle.$$

The numbers $\theta$ and $\varphi$ define a point on the unit three dimensional sphere, as shown in *Figure* 1.2. This sphere is called the *Bloch sphere*.

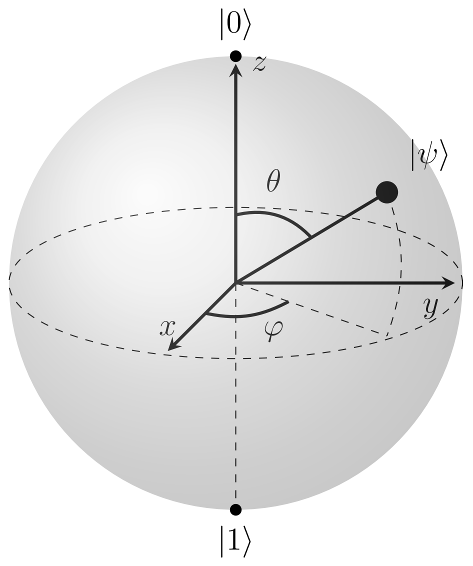


Figure 1.2: Bloch sphere where are highlighted the polar coordinates of the state $|\psi\rangle$. Notice that the “classical state" are in the poles.

Although the states of a qubit are infinitely many, the information we have access is the same as the classical bit, therefore, after measuring, we find the qubit or in $|0\rangle$ or in $|1\rangle$, because, in quantum mechanics, after measuring a system its state changes.

Since qubits are quantum systems we want to exploit their properties. Therefore, in a general quantum computation we need to prepare an input state, implement an evolution by means of unitary transformation and measure the output state of the evolution. Informally, if we think about a bit, we consider it and all the classical gates as deterministic objects and functions; on the other hand, we can see a qubit as a distribution of two different states, therefore the idea is to use the evolution of the system, i.e. unitary matrices, for mapping distributions in distributions.

By summarising all the notions we gave above about qubits, we can state that a system with two different energy levels is suitable for our purpose whether, by starting from the fundamental state $|0\rangle$, we can create every state allowed in the system by means of unitary transformations and we can measure the state in the computational basis $\{|0\rangle, |1\rangle\}$. Being able to apply this measurements means to be able to measure the observable[2]

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

which has eigenstates $|0\rangle$ and $|1\rangle$.

For storing and manipulating information we need to be able to obtain string of bits. For achieving this result in a quantum framework we can just consider a composite system of qubits. For recalling the formalism of composite systems let us consider two qubits. If these were classical bits we would have four different possibilities $00, 01, 10$ and $11$. In a two qubits system we have four different computational states, denoted $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$,[3] as well. But, now, we know that a quantum state could be in any possible superposition of this four states, therefore a general wavefunction of two qubits is

$$|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle.$$

Here of course the probability that we measure one of the four possible outcomes is $|a_{ij}|^2$, with $i, j \in \{0, 1\}$, thus these amplitudes must satisfy the normalization constraint $\sum_{i,j\in\{0,1\}} |a_{ij}|^2 = 1$ . Eventually, we recall that we can measure a subset of the qubits. For instance, if we measure the state $|\psi\rangle$

[2]Notice that this operator is also unitary, in fact it is an important quantum gate as well. Following this concept is explained.

[3]From now on we are going to denote the composite state $|i_1\rangle \cdots |i_k\rangle$ as $|i_1 \cdots i_k\rangle$.

on the first qubit and as an outcome we get $|0\rangle$, which occurs with probability $|a_{00}|^2 + |a_{01}|^2$, then the state $|\psi\rangle$ after the measurement become

$$|\tilde{\psi}\rangle = \frac{a_{00}|00\rangle + a_{01}|01\rangle}{\sqrt{|a_{00}|^2 + |a_{01}|^2}}.$$

Beyond the previous simple example, generally, we are interested in using more qubits for creating longer strings. Indeed quantum computers are based on strings of bits as well as classical ones.

We can think about quantum computers as finite collections of $n$ qubits. A general set of $k$ qubit in a quantum computer is called a *quantum register* of size $k$. While a general state in a classical computer with $n$ bits can be described by means of the string $i = (i_1 i_2 \cdots i_n)$, where $i_j \in \{0,1\}$, which represents the number

$$i = 2^{n-1}i_1 + 2^{n-2} + \cdots + 2^1 i_{n-1} + 2^0 i_n, \tag{1.15}$$

in the case of quantum computation the general state is a superposition of all the possible states described in (1.15). Thus the physical system we want to use is described by a $2^n$-dimensional Hilbert space obtained by the composition of $n$ $2$-dimensional Hilbert spaces, which are the qubits. Therefore, if we denote the state $|i_1 \cdots i_n\rangle$ as $|i\rangle$, where $i_1 \cdots i_n$ is the number written in base $2$ and $i$ is the number written in base $10$, we obtain that a general superposition of the computational states of $n$ qubits is

$$\sum_{i=0}^{2^n-1} c_i |i\rangle,$$

where, of course, the amplitudes $c_i \in \mathbb{C}$ are under a normalization constraint. Thus, we stress that if the $c_i$ are all non zero, then we have a superposition of $2^n$ states. Therefore, while classical computation needs different inputs for different runs, quantum computers can perform a computation for exponentially many inputs on a single run. This is the core of quantum computation power.

For having a consistence computation we want that the composite system, obtained by composing qubits, fulfills the same requirements that we stated for a single qubit. That is, we want to be able to generate any state allowed in the Hilbert space, which described our system, by means of unitary operators applied to the state $|\mathbf{0}\rangle := |0 \cdots 0\rangle$ and to measure the states in the computational basis, obtained by the tensor product of the computational basis of the single qubits. This last condition means that we need to be able to measure $\sigma_z$ on each single qubit.

As we said before, in the basic framework of a basic quantum algorithm we have to compute the evolution of a prepared state. Mathematically the evolution of a $n$ qubits register is represented by a $2^n \times 2^n$ matrix. This can always be decomposed in the product of matrices that represent a local evolution only on one or two qubits. Those matrices are the elementary operations of quantum computation and are called *quantum gates*. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a *quantum circuit* containing wires and quantum gates.

Like the classical case, even in a quantum circuit wires carry around the information and gates manipulate it.

Following we introduce two fundamental single-qubit quantum gates: the *Hadamard gate* and the *phase-shift gate*. These are important because by means of them we can prove that we can create any unitary transformation for a one qubit system.

The Hadamard gate is defined as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

This gate creates a uniform[4] superposition of the computational basis states

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

It is also worthy to note that $H^2 = \mathbb{I}$. Furthermore,

$$H^\dagger = H^{-1} = H,$$

where $H^\dagger$ is the conjugate transpose matrix of $H$. The first equality is due to the unitarity of the matrix and the second one since what we stressed about $H$.

[4]In the probabilities point of view.

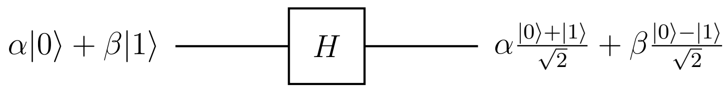

$\alpha|0\rangle + \beta|1\rangle$
$H$
$\alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$

Figure 1.3: Graphical representation of a quantum circuit where we have applied an Hadamard gate $H$.

The phase-shift gate is defined as:

$$R_z(\delta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix}.$$

It is a diagonal matrix that simply maps the state $|0\rangle$ in $|0\rangle$ and the state $|1\rangle$ in $e^{i\delta}|1\rangle$. So, on the computational basis state the action of this gate is trivial, because it changes only the global phase. However, on a general single-qubit state

$$R_z(\delta)|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix} \begin{pmatrix} \cos\frac{\theta}{2} \\ e^{i\varphi}\sin\frac{\theta}{2} \end{pmatrix} = \begin{pmatrix} \cos\frac{\theta}{2} \\ e^{i(\varphi+\delta)}\sin\frac{\theta}{2} \end{pmatrix}.$$

The action of this gate, then, it is not trivial because it modifies the relative phase.

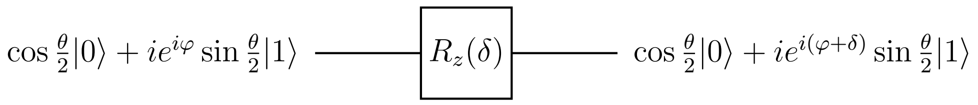

$\cos\frac{\theta}{2}|0\rangle + ie^{i\varphi}\sin\frac{\theta}{2}|1\rangle$
$R_z(\delta)$
$\cos\frac{\theta}{2}|0\rangle + ie^{i(\varphi+\delta)}\sin\frac{\theta}{2}|1\rangle$

Figure 1.4: Graphical representation of a quantum circuit where we have applied a phase-shift gate $R_z(\delta)$ on a general qubit written in the Bloch sphere framework.

It is important to realize that we can obtain any points on the Bloch sphere starting from the state $|0\rangle$ by applying only Hadamard and phase-shift gates:

$$R_z\left(\frac{\pi}{2}+\varphi\right) HR_z(\theta)H|0\rangle = e^{i\frac{\theta}{2}}\left(\cos\frac{\theta}{2}|0\rangle + ie^{i\varphi}\sin\frac{\theta}{2}\right).$$

Finally, we present the *Pauli gates*, which we used in our algorithm

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

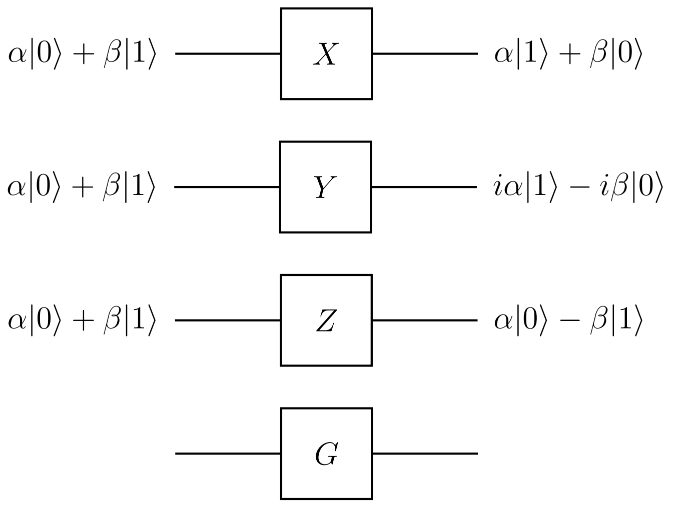

$\alpha|0\rangle + \beta|1\rangle$
$X$
$\alpha|1\rangle + \beta|0\rangle$
$\alpha|0\rangle + \beta|1\rangle$
$Y$
$i\alpha|1\rangle - i\beta|0\rangle$
$\alpha|0\rangle + \beta|1\rangle$
$Z$
$\alpha|0\rangle - \beta|1\rangle$
$G$

Figure 1.5: The representation of the Pauli gates in a quantum circuit. The last gate $G$ presents how to represent a general single-qubit gate action on a qubit.

First of all notice that they are symmetries and so $\sigma_j^2 = \mathbb{I}$, they anticommute $\sigma_j\sigma_h = -\sigma_h\sigma_j$ for $h \neq j$ and $[\sigma_h, \sigma_j] = 2i\epsilon_{hjk}\sigma_k$, where $[\cdot,\cdot]$ is the commutator and $\epsilon_{hjk}$ is the Levi-Civita tensor. The last property means that we are able to write each Pauli gate in function of the other ones. Eventually, we highlight that $\sigma_x$ is the quantum counterpart of the classical logic gate NOT and that $\sigma_z = R_z(\pi)$.

Above we mentioned the possibility of writing any evolution of a composite quantum system, i.e. a register of qubits, in a product of different single-qubit and two-qubits gates. Thus, let us present a two-qubits gates which is crucial for the universality result we will mention below.

Moreover, it is important because it creates entanglement between two qubits: the *controlled NOT*, or *CNOT*:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

As we can notice this matrix is non separable, i.e. there do not exist two matrices $A, B \in U_2(\mathbb{C})$ such that $A \otimes B =$ CNOT, therefore it can create entanglement. To stress the fact that the CNOT creates entanglement let us show how this gate makes the computational basis states evolve:

$$|00\rangle \to |00\rangle,\ |01\rangle \to |01\rangle,\ |10\rangle \to |11\rangle,\ |11\rangle \to |10\rangle.$$

Informally, for understanding what this gate does, we can interpret its action as a conditional command "if"[5]. Straightforwardly, by considering the system composed by two registers of size $1$, CNOT says to apply a gate $\sigma_x$ on the second register whether the first one is on the state $|1\rangle$. Hence, it is called "controlled" because we are checking the state of a qubit for deciding whether apply the $X$ gate or not.

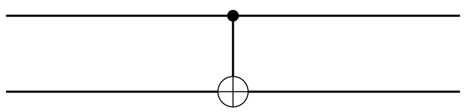
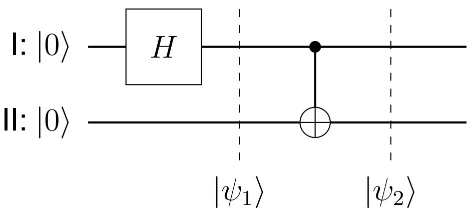


Figure 1.6: Graphical representation of a CONT gate in a $2$ qubits register in a quantum circuit. The black dot

To see clearly the creation of entanglement we can consider the following quantum circuit



The dashed lines mean that we are considering the information brought by the wires at that specific point.

The circuit has as input both in the first and in the second register the state $|0\rangle$. At $|\psi_1\rangle$ we have applied only an Hadamard gate on the first register, so the state results in

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle.$$

[5]In the computer science point of view.

This is a separable state, i.e. it is not entangled. But after applying the CNOT gate, controlled on the first register and applied on the second we obtain

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

This last state is non separable and, thus, entangled. There are different two-qubits gates but the CNOT occurs to be important due to this result of universality.

**Theorem.** *Any multiple-qubit logic gates can be composed by CNOT and single qubit gates.*

However, in a quantum circuit, multi-qubits gates can still be expressed by means of unitary operators.

Before giving a first example of quantum algorithm we present how to implement a general function in a quantum circuit. Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function[6] and let us consider a quantum register of size $n$. For implementing the function in a gate we find a unitary matrix that describe an evolution of the system, in particular we stress that this matrix must be invertible. A trivial way to implement a general function is to add an auxiliary register where to store the output of the function and to keep the input. In such a way we are sure about the unitarity of the gate.

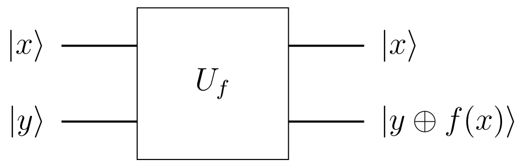


Figure 1.7: Example of a general unitary gate of a function $f$. In the case explained above the first register represented by $|x\rangle$ has size $n$, instead, the second one has size $1$ and it is used for storing the output of the function by means of $\oplus$, which is the sum on $\mathbb{Z}_2$, the ring of integer modulo $2$. It is easy to see that by applying again the same operator $U_f$ we obtain $|x\rangle|y\rangle$ as output, i.e. $U_f = U_f^{-1}$ and so it is invertible.

[6]For the sake of simplicity we have taken into account only a Boolean function but further we will see also the case of functions with multiple output

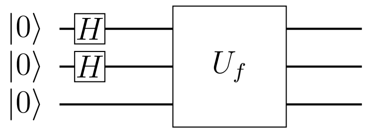
Every multi-qubits gate creates entanglement because in its decomposition in single-qubit gates and CNOT, the latter can appear. Therefore also the unitary gates defining starting by a function create entanglement. First of all, we spot that the gate CNOT itself can be seen as an implementation of a function. Indeed, let us consider the function

$$f : \{0,1\} \to \{0,1\},\ f : \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 1 \end{cases}$$

which is the identity. Now we take a register of $1$ qubit and we add an auxiliary register of $1$ qubit. Notice that the computational basis states of the composite system of the two registers are mapped in

$$\begin{aligned} &|0\rangle|0\rangle \mapsto |0\rangle|0 \oplus f(0)\rangle = |0\rangle|0\rangle,\ |0\rangle|1\rangle \mapsto |0\rangle|1 \oplus f(0)\rangle = |0\rangle|1\rangle \\ &|1\rangle|1\rangle \mapsto |1\rangle|1 \oplus f(1)\rangle = |1\rangle|0\rangle,\ |1\rangle|0\rangle \mapsto |1\rangle|0 \oplus f(1)\rangle = |1\rangle|1\rangle, \end{aligned}$$

which is exactly the action of a CNOT gate. Furthermore, other functions can generate entanglement, let us consider the following example. Let $f\{0,1\}^2 \to \{0,1\}$ with $f :=$AND, the classical logic gate that returns $1$ if and only if both of the inputs are $1$. Then, if we prepared a quantum circuit with a first register of size $2$ and an auxiliary register of size $1$, we would be able to implement this function in the following circuit



After applying the circuit we obtain

$$\frac{1}{2}(|00\rangle|0\rangle + |01\rangle|0\rangle + |10\rangle|0\rangle + |11\rangle|1\rangle),$$

which is an entangled state.

Finally, we have all the tools for starting to present quantum algorithms. We define a *quantum algorithm* as an algorithm which runs on any realistic model of quantum computation, in our case our model are quantum circuits themselves. Following, we present one of the most famous algorithm called *Grover's search algorithm*.

Grover's algorithm is a quantum algorithm used for searching a marked element in an unstructured database. Classically, for finding a specific element, which is also said to be marked, in an $N$ elements unstructured database, we check in average $N/2$ elements. For checking whether an element is the marked one we have to apply a function, called *oracle*, which returns $1$ if the element is marked and $0$ otherwise. This means that the number of checks corresponds to the number of evaluations of the oracle $f$. Then we define the *query complexity* of an algorithm as the number of time we query the oracle. Therefore, the classical query complexity of the algorithm for finding the marked element in an $N$ elements unstructured database is $N/2$.

We will see that Grover's algorithm has a query complexity of $O(\sqrt{N})$, hence it finds the marked elements with a quadratics advantage. We start presenting the algorithm with the case $N = 4$ and then we generalize it for arbitrary $N$.

For describing four different elements in a quantum circuit, we need only two qubits, in fact, the system with $2$ qubits has $2^2 = 4 = N$ different states. Therefore, we need a first register of size $2$ and, since we need to use an oracle, i.e. a function, we need an auxiliary register with a qubit. By denoting as $\bar{x}$ the marked element, the oracle works in this way

$$f : D \to \{0, 1\}, \; f(x) := \begin{cases} 1 & \text{if } x = \bar{x}, \\ 0 & \text{otherwise.} \end{cases}$$

The idea behind the algorithm is to exploit entanglement, highlight the marked element by changing its phase in the superposition by means of the oracle and apply a change of basis[7] for making the superposition become only the marked element state. Let us see it in details.

Let us suppose that the marked element is represented by the state $|01\rangle$. First of all, we consider all of the states which represent the elements in our database. So in the first register we have

$$H^{\otimes 2}|00\rangle = \left[\frac{1}{\sqrt{2}}|0\rangle + |1\rangle\right] \otimes \left[\frac{1}{\sqrt{2}}|0\rangle + |1\rangle\right] = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle),$$

where $H^{\otimes n} := \overbrace{H \otimes \cdots \otimes H}^{n \text{ times}}$. For what concerns the auxiliary register we initialize the state $|1\rangle$ and then we apply an Hadamard gate $H$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

[7]Unitary matrices map orthonormal basis in orthonormal basis.

This choice will be clear soon. By applying this two initial gates, we obtain

$$\underbrace{\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)}_{\text{Register that represents the elements}} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{\text{Auxiliary register}}.$$

And now we can see that by using the oracle $U_f$ we have that

$$U_f|x\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} = |x\rangle\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = \begin{cases} -|x\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{if } x = \bar{x} \\ |x\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{otherwise} \end{cases}$$

Essentially, with the application of the oracle we are changing the relative phase of the marked element in the superposition, in particular in our case after $U_f$ we obtain

$$\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

As we said before this does not suffice to measure the marked element with certainty. Indeed, we can easily notice that now in our superposition we have changed only the phases of our states, but the amplitudes modulus remained the same as well as the probability of measuring any state. Therefore, the idea is to make this phase difference become an amplitude one. In order to do this we apply a basis change which is represented by the matrix

$$D_{ij} = -\delta_{ij} + \frac{2}{2^n}, \tag{1.16}$$

where $\delta_{ij}$ is the Kronecker delta and $n$ is such that $N = 2^n$. This operator is called the *diffusion operator* and when $N = 4$ it appears

$$D_4 = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}.$$

This matrix is unitary and it can be implemented in a quantum circuit, of course. Since we want to generalize this algorithm eventually, it is interesting to see how we can decompose this matrix in elementary gates. First of all, we can compute the diagonal matrix

$$D = (H^{\otimes 2})^\dagger D' H^{\otimes 2} = H^{\otimes 2} D' H^{\otimes 2},$$

where

$$D' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

and where

$$\begin{aligned} H^{\otimes n} H^{\otimes n} &= (H \otimes \cdots \otimes H)(H \otimes \cdots \otimes H) \\ &= H^2 \otimes \cdots \otimes H^2 = \mathbb{I} \otimes \cdots \otimes \mathbb{I} \\ &= \mathbb{I}, \end{aligned}$$

therefore $(H^{\otimes n})^\dagger = (H^{\otimes n})^{-1} = H^{\otimes n}$. Moreover, up to an overall global phase, we can decompose $D'$ with

$$D' = \sigma_x^{\otimes 2}(\mathbb{I} \otimes H)\text{CNOT}(\mathbb{I} \otimes H)\sigma_x^{\otimes 2},$$

where $\sigma_x^{\otimes 2} := \sigma_x \otimes \sigma_x$. Eventually, we can compute the last step for finding the marked element

$$D\frac{1}{2}\begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Here we have taken into account only the first register and we have computed all the tensor products as Kronecker products. Now, after measuring the first register, we obtain the marked state $|01\rangle$.

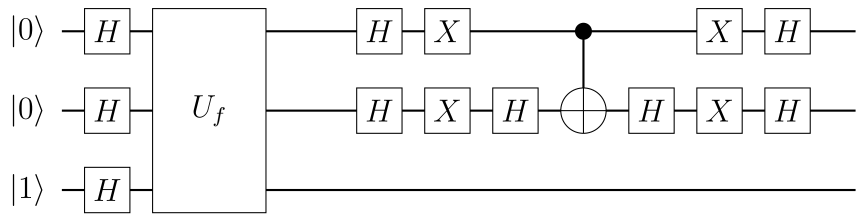


Figure 1.8: Quantum circuit model representing the Grover's algorithm.

Notice from *Figure* 1.8 that with only one query, i.e. one application of $U_f$, we are able to find the marked element, that is, the query complexity of the

Grover's search in a $4$ elements database is $1$.

Now, let us generalize this algorithm for arbitrary $n$. We describe the method for a general number $N = 2^n$ of items. As in the previous case, the registers are only two: the former of size $n$, where the states that represent the elements of the database are stored, and an auxiliary one for storing the result of the oracle. Thus by applying the Hadamard gates at both the register we obtain

$$(H^{\otimes n} \otimes H)|\mathbf{0}\rangle|1\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}},$$

where $|\mathbf{0}\rangle := |00\cdots 0\rangle$ is the fundamental state of the first register. Before carrying on our discussion we need to prove

$$H^{\otimes n}|\mathbf{0}\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle.$$

Notice that the Hadamard gate can be stated also as

$$H|x\rangle = \frac{1}{\sqrt{2}} \sum_{y=0,1} (-1)^{xy}|y\rangle,$$

where $x \in \{0,1\}$. Let us prove that

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle. \tag{1.17}$$

The case $n = 1$ is clear. For $n = 2$, generally

$$\begin{aligned}(H \otimes H)|x_1\rangle|x_2\rangle &= \left(\frac{1}{\sqrt{2}} \sum_{y_1=0,1} (-1)^{x_1y_1}|y_1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_2=0,1} (-1)^{x_2y_2}|y_2\rangle\right) \\ &= \frac{1}{2} \sum_{y_1,y_2=0,1} (-1)^{x_1y_1+x_2y_2}|y_1\rangle|y_2\rangle \\ &= \frac{1}{2} \sum_{y\in\{0,1\}^2} (-1)^{x\cdot y}|y\rangle,\end{aligned}$$

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. This argument can be applied for $n$ arbitrary

$$\begin{aligned}(H \otimes \cdots \otimes H)|x\rangle &= \frac{1}{\sqrt{2^n}} \sum_{y_1,\ldots,y_n=0,1} (-1)^{x_1y_1+\cdots+x_ny_n}|y_1\rangle \cdots |y_n\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle,\end{aligned}$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1 \ldots, y_n)$. But, now, by setting $|x\rangle = |\mathbf{0}\rangle$ we easily see that the term $(-1)^{x\cdot y} = 1$ and, therefore, we have proven equation (1.17).

Now we can continue with the generalization of the Grover's algorithm. At the last step we obtained the superposition of all the states which represent the items in the database. At this point we query the oracle and change the relative phase of the marked element.

Finally, we apply the diffusion operator, which can be computed with the expression in (1.16) or by recalling the gate decomposition

$$D = (H^{\otimes n})^\dagger(-\mathbb{I} + 2|\mathbf{0}\rangle\langle\mathbf{0}|)H^{\otimes n},$$

where $D' = -\mathbb{I} + 2|\mathbf{0}\rangle\langle\mathbf{0}|$ is a reflection about the vector $|\mathbf{0}\rangle$. Unfortunately, after one application of the oracle we are not able to measuring the marked element with certainty. For reaching a suitable probability to measure the right state we have to apply the operator $G = DU_f$ several times.

Now, the goal is to compute the query complexity of this algorithm. For understanding it we give a geometrical interpretation of the Grover's algorithm. For the sake of simplicity, we neglect the auxiliary register whose state is always factorized and never changes.

First of all, it is clear that the action of the oracle $U_f$ is

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

and it can be implemented by means of the reflection

$$U_f = \mathbb{I} - 2|\bar{x}\rangle\langle\bar{x}| =: R_{|\bar{x}\rangle}.$$

Therefore, geometrically, $U_f$ makes a reflection about the orthogonal subspace of $|\bar{x}\rangle$ in $\mathcal{H}$, which is the Hilbert space that describes the system. By considering a state $|\psi\rangle$ we can decompose it in parallel and orthogonal part

$$|\psi\rangle = \alpha|\bar{x}\rangle + \beta|\bar{x}^\perp\rangle,$$

where $\alpha, \beta \in \mathbb{C}$. Thus, after the evaluation of the oracle we get

$$U_f|\psi\rangle = -\alpha|\bar{x}\rangle + \beta|\bar{x}^{\perp}\rangle.$$

Eventually, we consider the other component of $G$, $D$. As we defined above

$$D = (H^{\otimes n})^{\dagger}(-\mathbb{I} + 2|\mathbf{0}\rangle\langle\mathbf{0}|)H^{\otimes n} = -\mathbb{I} + 2H^{\otimes n}|\mathbf{0}\rangle\langle\mathbf{0}|H^{\otimes n} = -\mathbb{I} + 2|S\rangle\langle S|,$$

where we have used that $(H^{\otimes n})^{\dagger} = H^{\otimes n}$ and $|S\rangle$ is equal to the state

$$|S\rangle = H^{\otimes n}|\mathbf{0}\rangle = \frac{1}{\sqrt{2}}\sum_{i=0}^{2^n-1}|i\rangle.$$

Therefore, we can see the operator $D$ as a reflection as well

$$D = -\mathbb{I} + 2|S\rangle\langle S| = -R_{|S\rangle}.$$

Finally, we can write the Grover's operator $G$ as a composition of two different reflections

$$G = -R_{|S\rangle}R_{|\bar{x}\rangle}.$$

Now, for understanding the geometrical action of $G$ let us focus on the plane spanned by $\{|\bar{x}\rangle, |S\rangle\}$. In this plane we need also to consider the vectors $|\bar{x}^{\perp}\rangle$ and $|S^{\perp}\rangle$, which are the orthogonal vectors respectively of the states $|\bar{x}\rangle$ and $|S\rangle$ in the plane spanned by $\{|\bar{x}\rangle, |S\rangle\}$. Notice that

$$G = -R_{|S\rangle}R_{|\bar{x}\rangle} = R_{|S^{\perp}\rangle}R_{|\bar{x}\rangle}, x$$

where $R_{|S^{\perp}\rangle}$ is the reflection about $|S\rangle$. In *Figure* 1.9 we show that if the angle between $|\bar{x}^{\perp}\rangle$ and $|S\rangle$ is $\theta$, then the angle between a generic state vector $|\psi\rangle$ and $G|\psi\rangle$ is $2\theta$.

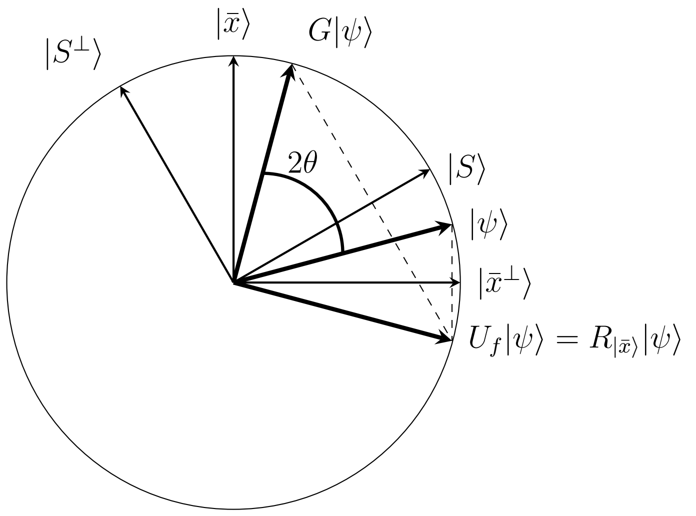


Figure 1.9: Graphical proof of the action of the operator $G$ on a generic vector $|\psi\rangle$ into the subspace spanned by $\{|\bar{x}\rangle, |S\rangle\}$. $\theta$ is the angle between $|\bar{x}^{\perp}\rangle$ and $|S\rangle$.

Therefore, by taking into account a general state $|\psi_0\rangle$, we can consider it as

$$|\psi_0\rangle := |S\rangle = \sin\theta|\bar{x}\rangle + \cos\theta|\bar{x}^{\perp}\rangle,$$

and after applying the operator $G$ $j$ times we obtain

$$|\psi_j\rangle = G^j|\psi_0\rangle = \sin\left((2j+1)\theta\right)|\bar{x}\rangle + \cos\left((2j+1)\theta\right)|\bar{x}^{\perp}\rangle.$$

Thus, for reaching almost the certainty we apply the operator in such a way

$$(2j+1)\theta \approx \frac{\pi}{2}.$$

The number of times we apply the algorithm for being as near as possible to the solution is

$$j = \left\lceil \frac{\pi}{4\theta} - \frac{1}{2} \right\rceil.$$

Finally, we are able to compute the query complexity of measuring the marked element with large $N$.

We recall that we are starting from $|\psi_0\rangle = |S\rangle$, therefore

$$\sin\theta = \langle\psi_0|\bar{x}\rangle = \frac{1}{\sqrt{N}}$$

and therefore for large $N$, we can state $\theta \approx 1/\sqrt{N}$ and

$$j = \left\lceil \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \right\rceil = O\left(\sqrt{N}\right)$$

At the end, notice that the Grover's algorithm for large $N$ does not find the marked element with certainty, but there is some probability of failure. After running the algorithm, we measure the state obtained in the computational basis and we query again the oracle for checking whether the state is in the right state or not. In the negative case we have to repeat the whole algorithm again.

Furthermore, from Grover's algorithm we can derive a method for increasing the probability of measuring a chosen state with certainty. This technique is called *amplitude amplification*. Let us taking into account a quantum algorithm that use no measurements $\mathcal{A}$, i.e. the algorithm is invertible, and acts on a quantum register of size $n$. We denote with $|\psi\rangle := \mathcal{A}|\mathbf{0}\rangle$ the final superposed state where the chosen state can be measured with probability $a$, $0 < a < 1$. The idea is to substitute in the Grover operator $G$ the Hadamard gates, which initialize the superposition $|S\rangle$, for the algorithm $\mathcal{A}$ which creates a specific superposition containing the chosen state. The operator we are looking for is

$$Q = -\mathcal{A}R_{|\mathbf{0}\rangle}\mathcal{A}^{-1}R_{|\bar{x}\rangle}, \tag{1.18}$$

which is, indeed, similar to the operator $G$, seen in the geometrical interpretation of Grover's search. In fact, $Q$ is a composition of two reflections: $R_{|\bar{x}\rangle}$, which is defined about the orthogonal space to $|\bar{x}\rangle$ and $R_{|\psi^{\perp}\rangle} := -\mathcal{A}R_{|\mathbf{0}\rangle}\mathcal{A}^{-1}$, which is defined about the state $|\psi\rangle$. The reflection $R_{|\mathbf{0}\rangle}$ is the reflection about the state $|\mathbf{0}\rangle$.

By recalling that we are always able to find an orthogonal decomposition of a general state, we can consider $|\psi\rangle = \sin(\theta_a)|\bar{x}\rangle + \cos(\theta_a)|\bar{x}^{\perp}\rangle$, where $a = \sin^2(\theta_a)$ and $\langle\bar{x}^{\perp}|\bar{x}\rangle = 0$, i.e. they are orthogonal. We can show that the subspace spanned by $\{|\bar{x}\rangle, |\bar{x}^{\perp}\rangle\}$, which we denote as $\mathcal{H}_{\psi}$, is stable under the action of $Q$.

Let us prove it. First of all, notice that the vector $|\bar{x}^{\perp}\rangle$ depends on $|\psi\rangle$ because it is the projection of $|\psi\rangle$ into the orthogonal hyperplane to $|\bar{x}\rangle$. Hence, since $0 < a < 1$ the subspace $\mathcal{H}_{\psi}$ has dimension $2$. We recall that the action of $Q$ on $\mathcal{H}_{\psi}$, as we noticed before, is given by the application of two reflections

$$R_{|\psi^{\perp}\rangle}R_{|\bar{x}\rangle}.$$

Consider, now, the orthogonal complement $\mathcal{H}_\psi^\perp$ of $\mathcal{H}_\psi$ in $\mathcal{H}$, which is the space that describes the whole system. Since the operator $\mathcal{A}R_{|\mathbf{0}\rangle}\mathcal{A}^{-1}$ acts like the identity on $\mathcal{H}_\psi^\perp$, because $|\psi\rangle \in \mathcal{H}_\psi$, then the operator $Q$ acts as $-R_{|\bar{x}\rangle}$ on $\mathcal{H}_\psi^\perp$. Thus $Q^2$ is the identity and the eigenvalues of $Q$ are only $+1$ and $-1$ in $\mathcal{H}_\psi^\perp$. It follows that for understanding the action of $Q$ on $\mathcal{H}$, it suffices to look at the action on the subspace $\mathcal{H}_\psi$.

Since the operator $Q$ is unitary and it is acting in a subspace spanned by two vectors $\mathcal{H}_\psi$ then there exists an orthonormal basis of two eigenvectors

$$|\psi_\pm\rangle = \frac{1}{\sqrt{2}}\left(|\bar{x}\rangle \pm i|\bar{x}^\perp\rangle\right),$$

associated with the eigenvalues

$$\lambda_\pm = e^{\pm i2\theta_a},$$

where, again, $\theta_a$ is such that $a = \sin^2\theta_a$ and $0 \leq \theta_a \leq \pi/2$.

Now, we show how $Q$ acts on $|\psi\rangle$. Let us consider the following decomposition of $|\psi\rangle$:

$$|\psi\rangle = \mathcal{A}|0\rangle = \frac{-i}{\sqrt{2}}\left(e^{i\theta_a}|\psi_+\rangle - e^{-i\theta_a}|\psi_-\rangle\right).$$

It is now immediate to see the action of $Q$ after $j$ iterations:

$$\begin{aligned} Q^j &= \frac{-i}{\sqrt{2}}\left(e^{i(2j+1)\theta_a}|\psi_+\rangle - e^{-i(2j+1)\theta_a}|\psi_-\rangle\right) \\ &= \sin((2j+1)\theta_a)|\bar{x}\rangle + \cos((2j+1)\theta_a)|\bar{x}^\perp\rangle. \end{aligned}$$

Therefore if we compute $Q^m|\psi\rangle$ for some integer $m \geq 0$ then the final probability of measuring the state $|\bar{x}\rangle$ is $\sin^2((2m+1)\theta_a)$.

As the Grove's search case, we want to find the integer $m$ such that $\sin((2m+1)\theta_a) = 1$. We can distinguish two case. The first case is when the number

$$\tilde{m} = \frac{\pi}{4\theta} - \frac{1}{2}$$

is an integer. It is really straightforward to see that by applying $Q^{\tilde{m}}$ to $|\psi\rangle$ we measure the chosen state $|\bar{x}\rangle$ with likelihood exactly $1$. The other case is when the number $\tilde{m}$ is not an integer. In this case we take into account the number $\bar{m} = \lceil\tilde{m}\rceil$ and the angle $\bar{\theta_a} = \pi/(4\bar{m}+2)$, which is slightly smaller than $\theta_a$.

Of course any quantum algorithm, that succeeds with probability $\bar{a} = \sin^2(\bar{\theta_a})$, succeeds with certainty after $\bar{m}$ application of the amplitude amplification operator. For obtaining with certainty the chosen state in this case we have to define a new algorithm based on $\mathcal{A}$ and which succeeds with probability $\bar{a}$. Such algorithm is obtained by considering an auxiliary register of size $1$. In fact, by preparing the state

$$\left(\sqrt{1-\frac{\bar{a}}{a}}\right)|0\rangle + \left(\sqrt{\frac{\bar{a}}{a}}\right)|1\rangle$$

in the auxiliary register and by substituting the operator $R_{|\bar{x}\rangle}$ with an operator $U_f$, which is an oracle defined as in the Grover's search algorithm, we can apply the algorithm $\mathcal{A}$ and we obtain that the probability of measuring $|\bar{x}\rangle$ is exactly $\bar{a}$ and, therefore, we can apply $Q^{\bar{m}}$ to the new $|\psi\rangle$ for measuring the marked element with certainty. We do not enter in the details of the last case because in our discussion $\tilde{m}$ will be always an integer.

# Chapter 2

# Hypercubes and unique sink orientations

In this chapter we will give the definition and the properties of undirected hypercubes and then we will present the unique sink orientation problems and different results about unique sink orientation on hypercubes.

## 2.1 Hypercubes

$n$-*hypercubes* can be defined in two different equivalent ways.

In the former we define an $n$-hypercube, or simply a cube, as an undirected graph $\mathcal{C}_n$ (sometimes if the dimension is not specified, we denote it as $\mathcal{C}$). The vertex set[1] is $V(\mathcal{C}_n) = \mathbb{Z}_2^n$, where $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$ is the ring of integers modulo $2$, endowed by the operation

$$\cdot \oplus \cdot : \mathbb{Z}_2 \times \mathbb{Z}_2 \to \mathbb{Z}_2,\ (a,b) \mapsto a \oplus b = a + b \mod 2.$$

The operation $\oplus$ acts on the set $\mathbb{Z}_2^n$ element-wise. So,

$$v \oplus w = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \oplus \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 \oplus w_1 \\ \vdots \\ v_n \oplus w_n \end{pmatrix},$$

for any $v, w \in \mathbb{Z}_2^n$. The edge set is define by means of the operation $\oplus$:

$$E(\mathcal{C}_n) = \{\{u,v\} | u, v \in V(\mathcal{C}_n) : |u \oplus v| = 1\},$$

[1] For further details on graphs give a look at the Appendix.

where $|v| = \sum_{i=1}^{n} v_i$ with $v$ a generic vector in $\mathbb{Z}_2^n$.

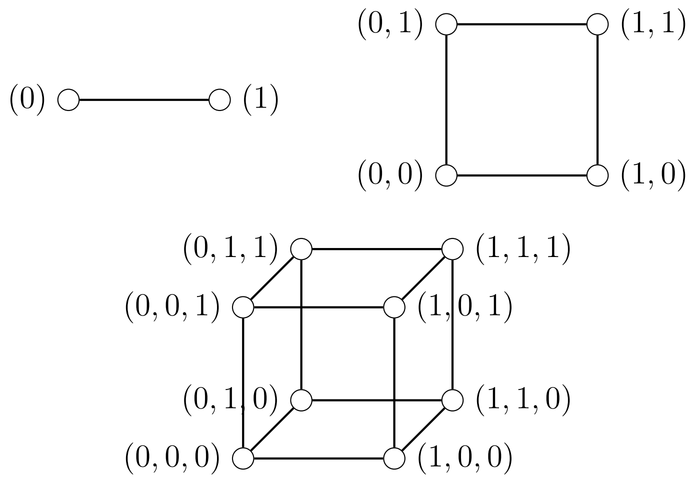


Figure 2.1: Three simple examples of hypercubes: $1$-hypercube, $2$-hypercube and $3$-hypercube in the “combinatorial view".

We call this definition the “combinatorial view".

In the latter definition we describe vertices as sets. An $n$-hypercube, $\mathcal{C}_n$, is an undirected graph whose vertex set is

$$V(\mathcal{C}_n) := 2^{[n]},$$

where [n]:={1,. . . ,n} and $2^{[n]}$ is its power set. The edge set is defined as following

$$E(\mathcal{C}_n) := \{\{u, v\} | u, v \in V(\mathcal{C}_n) : |u \oplus v| = 1\},$$

where, in this case, $\oplus$ is the symmetric difference and $| \cdot |$ is the cardinality of the set. We call this definition the “set view".

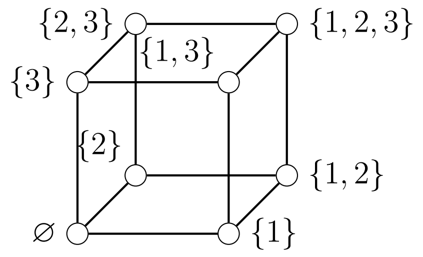


Figure 2.2: A $3$-hypercube in the “set view".

We use the same notation for defining both “views" because they are equivalent. Notice that the cardinalities of the two vertex sets are the same

$$|\mathbb{Z}_2^n| = |\mathbb{Z}_2|^n = 2^n = 2^{|\{1,\dots,n\}|} = 2^{|[n]|} = \left|2^{[n]}\right| .$$

Then we can define the following group isomorphism between $\left(2^{[n]}, \oplus\right)$ and $(\mathbb{Z}_2^n, \oplus)$. Indeed we define the map $f$

$$2^{[n]} \ni v = \{i_1, \dots, i_k\} \longmapsto (b_i)_{i=1,\dots,n} = b \in \mathbb{Z}_2^n, \text{ where } b_i = \begin{cases} 1 & \text{if } i \in v \\ 0 & \text{otherwise} \end{cases} .$$

It is straightforward to see that this map is a group isomorphism and that is why to consider the notation $\oplus$ for both the “views" is consistent. This induces also a graph isomorphism between the graphs of the two definitions. In fact, $\{u, v\} \in E(C_n)$ implies that $|u \oplus v| = 1$ for any $u, v \in 2^{[n]}$, therefore, by calling $u \oplus v = \{\lambda\}$ we have that

$$f(u) \oplus f(v) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \}\lambda \ .$$

Hence, since $f(u), f(v) \in \mathbb{Z}_2^n$ and $|f(u) \oplus f(v)| = 1$, we have that $\{f(u), f(v)\} \in E(\mathcal{C}_n)$ for any $u, v \in 2^{[n]}$. So $f$ is a graph isomorphism.

Now we introduce different definitions and properties that will be useful and crucial in the analysis of the unique sink orientation problems on hypercubes. For what concerns this chapter we will always use the “set view".

Given $A, B \subseteq [n]$, let $[A, B] := \{X | A \subseteq X \subseteq B\}$. This is said to be an *interval*. Given two sets $A \subseteq B \subset [n]$, we define the graph $C$, called *subcube*, where $V(C) := [A, B]$ and

$$E(C) := \{\{u, u \oplus \{a\}\} | u \in V(C), a \in B \setminus A\}.$$

Moreover, we define an *edge labelling*

$$\lambda : \{u, u \oplus \{a\}\} \longmapsto a,$$

which is a mapping defined over the edge set that associates each edge with a *label*, which is sometimes called a *direction*. Sometimes we denote a subcube as $C^{[A,B]}$ for stressing the fact that it is induced by $[A, B]$. We call the set of the directions on a subcube $C$ the *carrier* of $C$, carr$C := \{B \setminus A\}$. The cardinality of the carrier of a subcube is called the *dimension* of the subcube and it is denoted as $dim\, C$.

Notice that subcubes are subgraphs of the main $n$-hypercube and, moreover, we can show that they are $(dim\, C)$-hypercubes. Given a subcube $C$, indeed, we can find a graph isomorphism between $C$ and the $k$-hypercube, with $k = dim\, C$. Let us consider the edge labelling $\lambda$ of $C$. Its image is

$$\text{Im}(\lambda) := \{\lambda_1, \dots, \lambda_k\} = \text{carr}C.$$

By spotting that the interval

$$V(C) = [A, B] = \{A \oplus D | D \subseteq \text{carr}C\},$$

the map $g$ is defined as

$$Q = A \oplus D_Q \mapsto \{i | \lambda_i \in D_Q\} \subseteq [k],$$

for any $Q \in [A, B]$ and $D_Q \subseteq$ carr$C$. For proving that $g$ is a graph isomorphism, it suffices to note that $\{u, v\} \in E(C)$ if and only if $v = u \oplus \{a\}$ for some $a \in$ carr$C$. In fact, if $u = A \oplus D_u$ (the case in which $u$ contains $\{a\}$ is analogous), then $v = A \oplus (D_u \cup \{a\})$ and

$$|g(u) \oplus g(v)| = |\{i|\lambda_i \in D_u\} \oplus \{i|\lambda_i \in D_u \cup \{a\}\}| = |\{i|\lambda_i \in \{a\}\}| = 1.$$

Hence, $\{g(u), g(v)\}$ is in the edge set of the $k$-hypercube for any $u, v \in E(C)$ and $g$ is a graph isomorphism.

The subcubes of an hypercube are also called *faces* or, if we are referring to a subcube of dimension $k$, they are called $k$-faces. Notice that the $n$-face defined by the interval $[\varnothing, [n]]$ is the whole $n$-hypercube, that the $1$-faces are the edges and the $0$-faces are the vertices.

We denote the $(n-1)$-faces as *facets*. For every direction $\lambda$ we can define two different facets. Let $\mathcal{C}_n$ be an $n$-hypercube and let us take a label $\lambda \in [n]$. We can define two different facets: the upper $\lambda$-facet, which is the subcube induced by the interval $[\{\lambda\}, [n]]$, and the lower $\lambda$-facet, which is the subcube induced by the interval $[\varnothing, [n] \setminus \{\lambda\}]$. We can easily see that in both cases the subcubes has dimension $n-1$ and indeed are facets, because we are considering all the possible directions of the cube except from $\lambda$, in fact in both cases $\lambda \notin \text{carr}C$.

## 2.2 Unique sink orientations on hypercubes

In this section we introduce unique sink orientations (USOs) on hypercubes and their properties.

First of all, we define what an orientation is. Given an undirected graph $G = (V, E)$, an orientation is a map $\phi : E \to V$ such that $\phi(e) \in e$. This maps edges to their correspondent sink vertex. Hence if $\{v, w\} \in E$ and $\phi(\{v, w\}) = v$, then the orientation has the arc $w \to v$ and in this case $w$ is a source and $v$ is a sink. Therefore, a graph $G$ endowed with an orientation $\phi$ is a digraph. Given a vertex we can partitioned its incident edges into those for which the vertex is a sink (ingoing edges) and those for which the vertex is the source (outgoing edges).

Given an orientation $\phi$ on an hypercube $\mathcal{C}$ and a subcube $C$, we define the *output* map $s : V(C) \to 2^{\text{carr}C}$ of $\phi$ as the map that assigns to every vertex the labels of the outgoing edges from that vertex.

Finally, a USO of an $n$-hypercube $\mathcal{C}_n$ is an orientation $\phi$ such that every face has a unique sink with respect to $\phi$. Or equivalently, in term of the output map, an orientation $\phi$ is a USO if for any two distinct vertices $v \subseteq w \subseteq [n]$ there exists in the subcube $C^{[v,w]}$ a unique vertex $u \in [v, w]$ such that $s(u) = \varnothing$, where $V(C^{[v,w]}) = [v, w]$ and $s$ is the outmap of $\phi$ in $C^{[v,w]}$.

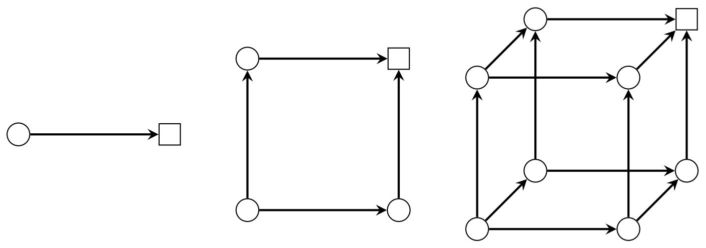


Figure 2.3: Example of $1$, $2$ and $3$-hypercube. The $\square$ node represents the global unique sink of the cube. We stress that every subcubes on the hypercube has a unique sink, in fact we can think at the $1$-hypercube as a subcube of the $2$-hypercube and the $2$-hypercubes as a subcube of the $3$-hypercube.

We denote a unique sink orientation $\phi$ on an $n$-hypercube as $\mathcal{C}_n^\phi$.

Now we present several properties of USO on hypercubes that we will use for defining our algorithms.

**Lemma 2.1.** *Given a unique sink outmap $s$ of a USO on an $n$-hypercube $\mathcal{C}_n^\phi$, and a set of directions $\Lambda \subseteq$ carr$\mathcal{C}_n^\phi$, the map defined by $s'(v) := \Lambda \oplus s(v)$ is the outmap of an USO.*

In other words, it means that whenever in $\mathcal{C}_n^\phi$ we reverse the orientation of all the edges with label $\lambda \in \Lambda$, then the new orientation obtained $\phi'$ is also a USO and its outmap is, indeed, $s'(v) := \Lambda \oplus s(v)$.

*Proof.* We first show that this is true for $\Lambda = \{\lambda\}$. We prove that the new outmap $s'$ corresponds to an orientation $\psi'$ that satisfies the USO definition. Let $C_u$ and $C_l$ the upper and lower $\lambda$-facets of $\mathcal{C}_n$ generated respectively by the intervals $[\{\lambda\}, [n]]$ and $[\varnothing, [n] \setminus \{\lambda\}]$. Consider any subcubes $C$ of $\mathcal{C}_n$. We have to prove that in each $C$ there is a unique sink for the orientation $\phi'$, defined as in the statement. If $C$ is entirely contained in a $\lambda$-facet then the orientation obtained by reversing all edges along the $\lambda$ direction does not change the orientation on any edges of $C$. In this case, since $\phi$ has a unique sink on $C$, so does $\phi'$ on $C$. On the other hand, if $C$ spans $\lambda$-facets, then we can partition $C$ into subcubes $\mathcal{D}_u := C \cap C_u$ and $\mathcal{D}_l := C \cap C_l$. Over the orientation $\phi$ suppose that these two subcubes have unique sink $o_u$ and $o_l$. Assume without loss of generality that the unique sink of $\phi$ over $C$ is $o_u$. Flipping all the edges along $\lambda$ means that $o_u$ is no longer a unique sink, since its $\lambda$ edge now points away from it. However, $o_l$ must now be a unique sink of $C$ since prior to flipping the $\lambda$ edge the only thing keeping $o_l$ from being a unique sink of $C$ was the $\lambda$ edge. All other vertices cannot be unique sinks of $C$ since they are not unique sinks in $\mathcal{D}_u$ and $\mathcal{D}_l$ and none of the edges in those subcubes are flipped. Hence, $\phi'$ is the outmap of a USO for $\Lambda = \{\lambda\}$.

For the general case of $\Lambda = \{\lambda_1, \dots, \lambda_k\}$, note that it follows from the application of the just proven case $k$ times since $s'(v) = \Lambda \oplus s(v) = \{\lambda_1\} \oplus \dots \oplus \{\lambda_k\} \oplus s(v)$. □

The above lemma implies the following results.

**Lemma 2.2.** *The outmap of a USO on an $n$-hypercube $C_n$ is a bijection.*

*Proof.* Given a USO $\phi$ on $C_n$, suppose that there exist two vertices $u \neq v$, which have the same image under the outmap s, $s(u) = s(v) = t$. Then consider the orientation $\phi'$ obtained by flipping all of the edges along the $t$ direction. This has outmap $s'(v) = t \oplus v$. Via *Lemma* 2.1, this new orientation is a USO. However, $s'(u) = s'(v) = \varnothing$, which is a contradiction. Hence the map is a bijection, because the cardinalities of the domain and of the codomain are the same. □

The following property is a characterization for USO and from it we can derive a property that we will use in the *Local Grover search algorithm*.

**Lemma 2.3.** *A mapping $s : V(\mathcal{C}) \to 2^{\text{carr}\mathcal{C}}$ is an outmap of a unique sink orientation of the cube $\mathcal{C}$ if and only if*

$$(s(u) \oplus s(v)) \cap (u \oplus v) \neq \varnothing \tag{2.1}$$

*for all $u, v \in V(\mathcal{C})$ with $u \neq v$.*

*Proof.* First assume $s$ satisfies the property (2.1) and fix a subcube $C^{[I,J]}$ of $\mathcal{C}$. We have to show that $C^{[I,J]}$ has a unique sink. Consider the restricted map $\tilde{s} : V(C^{[I,J]}) \to 2^{\text{carr}C^{[I,J]}}$,

$$\tilde{s}(u) = s(u) \cap \text{carr}C^{[I,J]},$$

that is the outmap of the orientation on $[I, J]$ induced by $s$. If $\tilde{s}$ is not injective there are $u, v \in [I, J]$, $u \neq v$, with $\tilde{s}(u) = \tilde{s}(v)$. For these vertices we get

$$(s(u) \oplus s(v)) \cap \text{carr}C^{[I,J]} = \varnothing$$

and since $u \oplus v \subseteq \text{carr}C^{[I,J]}$ this contradicts the assumption (2.1). Therefore $\tilde{s}$ is injective and thus bijective. In particular there exist only a vertex $o \in [I, J]$ such that $\tilde{s}(o) = \varnothing$ and, hence, it is the unique sink.

Conversely, assume $s$ has not the (2.1) property and let $u, v$ witness its failure. Then, since $(u \oplus v) \cap (s(u) \oplus s(v)) = \varnothing$, in the subcube induced by $[u \cap v, u \cup v]$ both $u$ and $v$ have the same outmap $\Lambda = s(u) \cap (u \oplus v) = s(v) \cap (u \oplus v)$. Thus $\Lambda \oplus s$ has two sinks in $[u \cap v, u \cup v]$, i.e. it is not a USO. By *Lemma* 2.1 $s$ is not a USO either.

□

Now we can present the idea which is the core of the *Local Grover search*.

Consider a general subcube $C := C^{[X,Y]}$ and let $A \subseteq \text{carr}C$. By removing all the edges with labels in $\text{carr}C \setminus A$, we obtain a graph whose connected components are exactly the faces of $C$ with carrier $A$. For every $v$ with $X \subseteq v \subseteq Y \setminus A$ there is a unique such face $F_v$ with carrier $A$ containing $v$. Now let $\phi$ be an orientation of $C$ such that every face with carrier $A$ has a unique sink. Then we define a mapping

$$\begin{aligned} s_{\phi/A} : [X, Y \setminus A] &\to 2^{\text{carr}C \setminus A} \\ v \qquad &\mapsto s(o_v), \end{aligned}$$

where $o_v$ is the unique sink of the face $F_v$ and $s$ is the outmap of the orientation $\phi$. In other words, $s_{\phi/A}$ maps a vertex $v$ in the set of labels of the edges outgoing from the sink of $F_v$. We call $s_{\phi/A}$ the *$A$-inherited outmap of $\phi$*. Note that a priori we do not know whether the inherited orientation is an outmap of any orientations on the subcube $C^{[X,Y \setminus A]}$. The following theorem proves that $s_{\phi/A}$ is, indeed, an outmap of a USO and we denote this property as the *heritage property*.

**Lemma 2.4.** *Let $\phi$ be a USO of a cube $\mathcal{C}$, and let $A \subseteq \text{carr}C$. Then $s_{\phi/A}$ is the outmap of a USO on a $(dim\, C - |A|)$-hypercube.*

*Proof.* This can be derived from characterization of *Lemma* 2.1. Indeed, let us suppose that $V(C) := [X, Y]$, then for every vertex $v \in [X, Y \setminus A]$ we have that $v = o_v \setminus A$. Then, for any $u, w \in V(\mathcal{C}') := [X, Y \setminus A]$, where $\mathcal{C}'$ is the $(dim\, C - |A|)$-hypercube

$$(u \oplus w) \cap (s_{\phi/A}(u) \oplus s_{\phi/A}(w)) \underset{(a)}{=} (o_u \oplus o_w) \cap (s(o_u) \oplus s(o_w)) \neq \varnothing$$

where $o_u$ and $o_w$ are respectively the sinks of the faces $F_u$ and $F_w$. Notice that $(a)$ holds because $A \nsubseteq (s(o_u) \oplus s(o_w))$ and so we can add in the other term of the intersection whatever elements of $A$. The last “$\neq$" holds because $s$ is a USO and so $S_{\phi/A}$ is a USO either. $\square$

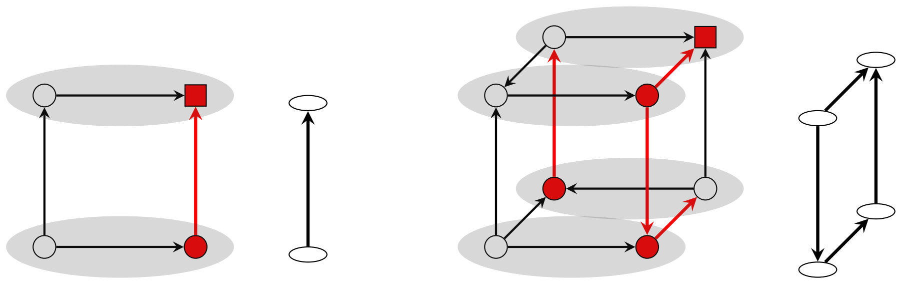


Figure 2.4: Example of cubes with their inherited orientation, the nodes of the inherited cube are highlighted with gray. The red node represent the sinks of the faces induced by the subset $A = \{1\}$. The correspondent inherited orientation is obtained by considering the faces $F_v$ as a single node with outgoing edges the ones of the sink of the face (such edges are highlighted with red).

# Chapter 3

# LP-induced unique sink orientations

In this chapter we present how to exploit the properties of hypercubes and, specially, unique sink orientation on hypercubes for solving linear programs. Hence, we show how to obtain an LP-induced orientation from a general LP in standard form. Then, we study its edge orientations for proving that it is, indeed, a USO. This procedure and the result below were proved by Gärtner and Schurr [6]. Let us consider a general linear program in the standard form

$$\begin{aligned} \text{max} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq \mathbf{0}, \end{aligned}$$

Now we change our point of view on LP to compute the edge orientations. For $x \in \mathbb{R}^n$ and $J \subseteq [n] := \{1, \dots, n\}$, $x_J$ is the $|J|$-dimensional vector obtained from $x$ by collecting all the coordinates with subscript in $J$, which as we noticed in *Chapter* 2 represents a node in the cube. For $A \in \mathbb{R}^{m \times n}$, $A_j \in \mathbb{R}^{m \times |J|}$ is the matrix that collects all the columns of $A$ with subscript in $J$.

With **0** being the zero vector of the appropriate dimension, we also use

$$\mathbb{R}^J := \{x \in \mathbb{R}^n | x_{[n] \setminus J} = \mathbf{0}\}.$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function with continuous partial derivatives. For $I \subseteq J \subseteq [n]$ we define a *convex program* as

$$\begin{array}{lll} \text{CP}(I,J) & \min & f(x) \\ & \text{s.t.} & Ax = b \\ & & x \in \mathbb{R}^J \\ & & x_{J \setminus I} \geq 0, \end{array}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. By the well known Karush-Khun-Tucker conditions we obtain the following theorem:

**Theorem 3.1.** $x^* \in \mathbb{R}^n$ *is an optimal solution to the problem CP(I,J) if and only if*

*(i)* $Ax^* = b,\ x^* \in \mathbb{R}^J,\ x^*_{J\setminus I} \geq 0$

*(ii) there exist* $\lambda \in \mathbb{R}^m$ *such that for all* $j \in J$,

$$\nabla f(x^*)_j - \lambda^T A_j \geq 0$$

*with equality if* $j \in I$ *or* $x^*_j > 0$.

Here $\nabla f(x^*)$ is the gradient of the function $f$ evaluated at $x^*$ which by convection is an $n$-dimensional row vector.

*Proof.* Assume $(i)$ and $(ii)$ hold, then we get:

$$\begin{aligned} \left(\nabla f(x^*) - \lambda^T A\right) x^* &= 0, \\ \left(\nabla f(x^*) - \lambda^T A\right) x &\geq 0, \end{aligned}$$

for some suitable $\lambda \in \mathbb{R}^m$ and for all the feasible solution of CP(I,J). By subtracting the first equation to the second, the term $\lambda^T A$ cancels, and we get

$$\nabla f(x^*)(x - x^*) \geq 0. \tag{3.1}$$

This is a well-known optimality condition for $x^*$. In fact, if $x^*$ satisfies inequality (3.1) for any $x$ feasible, then, since $f$ is convex, we have

$$f(x) \geq f(x^*) + \nabla f(x^*)(x - x^*) \geq f(x^*).$$

On the other hand, if $x^*$ is optimal, by contradiction, suppose that there exists a feasible solution $x$ such that $\nabla f(x^*)(x-x^*) < 0$. Consider the point $z(t) = tx + (1-t)x^*$ with $t \in [0,1]$. Clearly $z(t)$ is feasible for any $t$. Notice that

$$\lim_{t\to 0} \frac{f(z(t)) - f(x^*)}{t} = \nabla f(x^*)(x-x^*) < 0.$$

Therefore, for $t$ small enough we have that $f(z(t)) < f(x^*)$, which contradicts our assumption of optimality.

In such a way we proved that since $x^*$ satisfies (3.1), it is optimal.

Conversely, let us assume that $x^*$ is optimal, thus feasible, which gives $(i)$. Moreover, since in this case (3.1) holds for all feasible solutions $x$, the vector $x^*$ is an optimal solution of the following LP as well

$$\begin{array}{rl} \text{min} & \nabla f(x^*)x \\ \text{s.t.} & Ax = b \\ & x \in \mathbb{R}^J \\ & x_{J\setminus I} \geq \mathbf{0}. \end{array}$$

Let consider the vector $\nabla f(x^*)$ as the one which defines an objective linear function and let $y^*$ be an optimal solution of the dual program

$$\begin{array}{rll} \text{max} & b^T y & \\ \text{s.t.} & y^T A_j = \nabla f(x^*)_j, & j \in I \\ & y^T A_j \leq \nabla f(x^*)_j, & j \in J \setminus I. \end{array}$$

By the complementary slackness condition, *Theorem* 1.4, $(y^*)^T A_j = \nabla f(x^*)_j$ for all $j \in J \setminus I$ with $x^*_j > 0$, implying that $\lambda := y^*$ fulfills $(ii)$. □

For describing the characterization that defines the connection between LP and USO we need to consider a specific problem. Let us fix a strictly convex function $f$ and consider for $I \subseteq J \subseteq [n]$ we consider the convex program

$$\begin{array}{rrl} \text{SCP}(I,J) & \min & f(x) \\ & \text{s.t.} & x \in \mathbb{R}^J \\ & & x_{J\setminus I} \geq \mathbf{0}. \end{array}$$

This is just the CP$(I, J)$ from above, restricted to the strictly convex case, and without any equality constraints. Since $f$ is strictly convex and SCP$(I, J)$ is feasible, the program SCP$(I, J)$ has a unique solution $x^*(I, J)$, for all pairs $I \subseteq J$. By applying *Theorem* 3.1 we see that $x^* := x^*(I, J)$ is optimal if and only if

$$x^*_{[n]\setminus J} = \mathbf{0}, \tag{3.2}$$

$$x^*_{J\setminus I} \geq \mathbf{0}, \tag{3.3}$$

by condition $(i)$, along with dual feasibility,

$$\nabla f(x^*)_I = \mathbf{0}, \tag{3.4}$$

$$\nabla f(x^*)_{J\setminus I} \geq \mathbf{0}, \tag{3.5}$$

and complementarity

$$\nabla f(x^*)_j x^*_j = 0, \quad j \in J \setminus I, \tag{3.6}$$

by condition $(ii)$.

Let us focus on the case $I = J$, for some considerations that will be suitable for reaching our goal . By reading the equation above notice that

$$\begin{aligned} x^*_{[n]\setminus J} &= \mathbf{0}, \\ \nabla f(x^*)_J &= \mathbf{0}. \end{aligned} \tag{3.7}$$

Now, in the “set view" of hypercubes, we choose any vertex $J \in [n]$ and check its orientation by means of the vector $x^*(J, J)$:

**Lemma 3.1.** *For $J \in [n]$, $j \in J$ and $I := J \setminus \{j\}$, the following two statements are equivalent*

*(i)* $x^*(J, J)_j > 0$.

*(ii)* $\nabla f(x^*(I, I))_j < 0$.

Notice that in this framework $I$ and $J$ represent two adjacent vertices of the cube.

*Proof.* If $x^*(J, J)_j > 0$, then $x^*(J, J)$ is feasible and, thus, optimal for the more restricted program SCP(I,J), i.e. the feasible region of SCP$(I, J)$ is contained in the SCP$(J, J)$ one, therefore an optimal solution in the latter is optimal in the former as well. On the other hand, $x^*(J,J)_j > 0$ shows that $x^*(J,J) \neq x^*(I,I)$, because of the constraint $x^*(I,I)_j = 0$. This means we have $x^*(I,I) \neq x^*(I,J)$, which is the only possible reason being that (3.5) fails for $x^* = x^*(I,I)$. This shows $\nabla f(x^*(I,I))_j < 0$.

Conversely, $\nabla f(x^*(I,I))_j < 0$ implies $x^*(I,I) \neq x^*(I,J)$, again for the conditions in *Theorem* 3.1, so $x^*(I,J)_j > 0$. Complementarity yields

$$\nabla f(x^*(I,J))_j = 0,$$

so $x^*(I,J)$ is an optimal solution for the restricted program SCP(I,J), by (3.7). This yields $x^*(J,J)_j = x^*(I,J)_j > 0$ □

**Theorem 3.2.** *For $J \subseteq [n]$, $j \in J$ and $I := J \setminus \{j\}$, the edge orientation*

$$I \to J : \Longleftrightarrow \; x^*(J,J)_j > 0 \; (\nabla f(x^*(I,I))_j < 0)$$

*define a USO on an $n$-hypercube.*

For lightening the notation in this *Chapter* the orientation function is denoted by an arrow "$\to$".

*Proof.* For understating that the above edge orientation is indeed a USO we have to check that every face has a unique sink. We recall that faces can be identified as intervals

$$[I,J] := \{F \subseteq [n] | I \subseteq F \subseteq J\}.$$

We claim that

$$S := I \cup \{j \in J | x^*(I,J)_j > 0\} \tag{3.8}$$

is the desired sink of the face $[I,J]$, $I \subseteq J$. First we observe that by the definition of $S$, $x^* = x^*(I,J)$ satisfies

$$x^*_{[n]\setminus S} = \mathbf{0} \tag{3.9}$$

$$\nabla f(x^*)_S = \mathbf{0}, \tag{3.10}$$

by (3.4) and complementary (3.6). It follows that $x^*(I,J) = x^*(S,S)$ by (3.7). Therefore,

$$x^*(S,S)_j > 0, \quad j \in S \setminus I \tag{3.11}$$

$$\nabla f(x^*(S,S))_j \geq 0, \quad j \in J \setminus S, \tag{3.12}$$

by (3.5). According to the definition of the orientation, $S$ is a sink of $[I, J]$.

Conversely, if $S$ is a sink of $[I, J]$, then the previous inequalities hold, so $x^*(S, S)$ is feasible for SCP$(I, J)$, since (3.9) and (3.11) imply (3.2) and (3.3), and it is feasible in the dual because of (3.10) and (3.12) imply (3.4) and (3.5). Complementarity (3.6) comes from (3.9) and (3.10). Thus $x^*(S, S) = x^*(I, J)$, where (3.11) forces $S$ to coincide with the set defined in (3.8). □

We know that we are able to find a equivalent reduction of SCP to USO for programs with quadratic objective function like

$$f(x) = x^T Q x + u^T x + w$$

for some symmetric positive definite matrix $Q \in \mathbb{R}^{n \times n}$.

So, given a general LP in a standard form with $n$ variables, we define for any $\epsilon > 0$ a quadratic function $f_\epsilon$ by

$$\begin{aligned} f_\epsilon(x) &:= x^T(A^T A + \epsilon^2 \mathbb{I})x - 2b^T A x - 2\epsilon c^T x \\ &= \|Ax - b\|^2 - 2\epsilon c^T x + \epsilon^2 \|x\|^2 - b^T b. \end{aligned}$$

Since $(A^T A + \epsilon^2 \mathbb{I})$ is positive definite for all $\epsilon$, $f_\epsilon$ is a strictly convex function. Let us define the program SCP$_\epsilon(I, J)$ as before with $f_\epsilon$ in place of $f$. We are interested in the behavior for $\epsilon \to 0$. We expect that in the limit, the program lexicographically minimize the tuple

$$(\|Ax - b\|^2, -2c^T x, \|x\|^2).$$

In the feasible and bounded case, the solution $x^*_\epsilon(\varnothing, [n])$ of SCP$_\epsilon(\varnothing, [n])$ should therefore converge to the optimal LP solution of the minimum norm.

In order to understand the USO induced by $f_\epsilon$ we have to know the value of $x^*_\epsilon(J, J)$, since *Theorem* 3.2. From the equation we spotted for the case $I = J$ we can notice that for finding this vector we have to solve the linear equation system

$$\frac{\nabla f(x)_J^T}{2} = (A^T A + \epsilon^2 \mathbb{I}) x_J - A_J^T b - \epsilon c_J = \mathbf{0} \tag{3.13}$$

with $x \in \mathbb{R}^J$, which arise form (3.6).

The following result shows why it does make sense to take into account the limiting of the orientation obtained by the program SCP$_\epsilon(\varnothing, [n])$.

**Lemma 3.2.** *Let $\overset{\epsilon}{\to}$ be the USO on an $n$-hypercube induced by $f_\epsilon$, according to Theorem (3.2). Then there exists a USO $\to$ such that $\to = \overset{\epsilon}{\to}$ for $\epsilon$ small enough.*

*Proof.* Using Cramer's rule for computing the solution $x^*_\epsilon(J,J)_j$ of the system defined in (3.13), we can see that the entries in $x^*(J,J)$ are all rational functions in $\epsilon$. By *Theorem* 3.2, $\overset{\epsilon}{\to}$ is defined by the sign of finitely many of these rational functions.

Now, for any nonzero rational function $r(\epsilon)$ there exist a small $\delta$ such that neither the numerator nor the denominator has any zeros in $(0,\delta)$. In this interval the sign of $r(\epsilon)$ is fixed. The lemma is proved. □

We stress the fact that the entries of the vector $x^*_\epsilon(J,J)$ are all rational function in $\epsilon$ and that in a suitable interval $(0,\delta)$, when $\delta$ is small enough, both the numerator and the denominator have no zeros in it. Then the responsible for this orientation is only the smallest $\epsilon$-power term. In fact, this determine the sign for $\epsilon \to 0$. Therefore, our goal becomes to see at the coefficients of the power series of $x^*_\epsilon(J,J)$. Moreover, since the limiting USO does not depend on $\epsilon$ we want to find a way for computing it avoiding $\epsilon$.

For this purpose we should define different strictly convex and quadratic auxiliary programs. The following programs are needed to compute directly the coefficients of the power series. Notice that is clear that all the programs are feasible and have only one optimal solution, except from the last one which need a proof of its feasibility.
For $J \subseteq [n]$, set

$$\begin{aligned} \bar{b}(J) = \text{argmin} \quad & (b-y)^T(b-y) \\ \text{s.t.} \quad & Ax = y \\ & x \in \mathbb{R}^J, \end{aligned}$$

$$\begin{aligned} x(J) = \text{argmin} \quad & x^T x \\ \text{s.t.} \quad & Ax = \bar{b}(J) \\ & x \in \mathbb{R}^J, \end{aligned}$$

$$\begin{aligned} \bar{c}(J) = \text{argmin} \quad & (c-x)^T(c-x) \\ \text{s.t.} \quad & A_J^T y = x_J, \end{aligned}$$

$$\begin{aligned} y(J) = \text{argmin} \quad & y^T y \\ \text{s.t.} \quad & A_J^T y = \bar{c}(J)_J, \end{aligned}$$

$$\begin{aligned} c(J) = \text{argmin} \quad & (c-x)^T(c-x) \\ \text{s.t.} \quad & Ax = \mathbf{0} \\ & x \in \mathbb{R}^J, \end{aligned}$$

$$\begin{aligned} b(J) = \text{argmin} \quad & (b-y)^T(b-y) \\ \text{s.t.} \quad & A_J^T y = \mathbf{0}, \end{aligned}$$

$$\begin{aligned} t(J) = \text{argmin} \quad & x^T x \\ \text{s.t.} \quad & Ax = y(J) \\ & x \in \mathbb{R}^J. \end{aligned}$$

If there are no constraints for the variables $x$ and $y$, it means that they are free and, thus, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$.

It appears counter-intuitive the fact that we have to solve quadratic programs. For understanding the reason why it is suitable, we analyze the computational cost of these problems. In fact, by assuming that also the last program is feasible and, thus, has a unique solution, we can see that convex programs with quadratic objective function can be solved in polynomial time by using the optimality conditions stated in *Theorem* 3.1 with $J = [n]$ and $I = \varnothing$. Indeed from the proof of the theorem, we obtain that for finding an optimal solution of a general convex problem we satisfy the relation (3.1) for any $x$ such that $Ax = b$. Notice that all of the programs we are taking into account have only equality constraints, thus we can represent them by a general set of constraints $Ax = b$. This means that we can rewrite any feasible solutions as $x = x^* + v$, where $v \in \ker(A)$, the kernel of $A$. Thus, by substituting $x = x^* + v$ in (3.1), we obtain the condition

$$\nabla f(x^*)v \geq 0, \text{ for any } v \in \ker(A).$$

If a linear function is non negative on a subspace, then it must be zero on that subspace, therefore we have the condition

$$\nabla f(x^*)v = 0, \text{ for any } v \in \ker(A),$$

or, in other words $\nabla f(x^*) \in \text{ker}(A)^\perp$, the orthogonal subspace of ker$(A)$ in $\mathbb{R}^n$.

But, now, by using the fact that ker$(A)^\perp = C(A^T)$, the vector space spanned by the columns of $A^T$, the optimality condition can be expressed as $\nabla f(x^*) \in C(A^T)$, i.e. there exists a vector $\nu \in \mathbb{R}^m$ such that

$$\nabla f(x^*) + A^T \nu = 0.$$

Therefore for finding the optimal solution we solve

$$Ax^* = b, \quad \nabla f(x^*) + A^T \nu = 0, \tag{3.14}$$

where $x^*$ is the optimal solution. Therefore, if the program to be solved is quadratic

$$\begin{array}{ll} \text{min} & f(x) = \frac{1}{2} x^T P x + q^T x \\ \text{s.t.} & Ax = b, \end{array}$$

where $P$ is a quadratic matrix of order $n$, $A \in \mathbb{R}^{p \times n}$ and $q \in \mathbb{R}^n$, then the relations (3.14) that we need to solve, are

$$Ax^* = b, \quad Px^* + q + A^T \nu = 0,$$

which we can write as

$$\begin{pmatrix} P & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x^* \\ \nu^* \end{pmatrix} = \begin{pmatrix} -q \\ b \end{pmatrix}.$$

We can notice that this system has $n+m$ variables with $n+m$ equations. Therefore, when the matrix is non singular we have a unique optimal solution $(x^*, \nu^*)$. If the matrix is singular but there exist at least a solution of the system, then every solution $(x^*, \nu^*)$ is an optimal solution of the convex program. Eventually, if the matrix is singular and has no solution then the problem is unbounded or infeasible. Indeed, let us suppose this is not infeasible. It means that there exist $v \in \mathbb{R}^n$ and $w \in \mathbb{R}^m$ such that

$$Pv + A^T w = 0, \quad Av = 0 \quad \text{and} \quad -q^T v + b^T w > 0. \tag{3.15}$$

Therefore by taking a feasible solution $\hat{x}$, we can study the whole line of feasible points $x = \hat{x} + tv$. By expanding up to the second order

$$\begin{aligned} f(\hat{x}+tv) &= f(\hat{x}) + t\left(v^T P\hat{x} + q^T v\right) + \frac{t^2}{2} v^T P v \\ &\underset{(a)}{=} f(\hat{x}) + t\left(-\hat{x}^T A^T w + q^T v\right) - \frac{t^2}{2} v^T A^T w \\ &= f(\hat{x}) + t(q^T v - b^T w), \end{aligned}$$

where in $(a)$ we have used the expressions (3.15) and the fact that $P$ is quadratic, thus symmetric. Hence the function decreases as $t \to \infty$.

Therefore, it is suitable for us to consider quadratic problems because for solving them we have to solve a linear system, which is a well-known polynomial issue.

Let us take a step back and give a look at the auxiliary quadratic problems we defined before. To get an intuition on what these values are, let us consider for $\gamma \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$ the linear program

$$\begin{array}{lll} \text{LP(J)} & \max & \gamma^T x \\ & \text{s.t.} & Ax = \beta \\ & & x \in \mathbb{R}^J, \end{array}$$

along with its dual

$$\begin{array}{lll} \text{LP}^{\Delta}\text{(J)} & \min & \beta^T y \\ & \text{s.t.} & A_J^T y = \gamma_J. \end{array}$$

The vector $\beta = \bar{b}(J)$ is the vector closest to $b$ such that LP(J) is feasible and $\bar{x}(J)$ is the feasible solution with minimum norm. In the dual, $\gamma = \bar{c}(J)$ is the vector closest to $c$ such that LP$^{\Delta}$(J) is feasible and $y(J)$ is the feasible solution with minimum norm. $c(J)$ is the projection of $c$ onto ker$A_J$, while $b(J)$ is the projection of $b$ onto the ker$A_J^T$. In the following lemma we take into account $t(J)$ and we explain how it enters the picture and that it has a unique solution.

**Lemma 3.3.** *For all $J \subseteq [n]$, the following holds.*

*(i) $2t(J)$ is the shortest vector that fulfills condition $(ii)$ in* Theorem *3.1 for the program defining $y(J)$.*

*(ii)* *If* $b(J) \neq \mathbf{0}$*, then* $b^T b(J) > 0$*, and if* $c(J) \neq \mathbf{0}$*, then* $c^T c(J) > 0$*.*

*(iii)* $b = b(J) + \bar{b}(J)$ *and* $c = c(J) + \bar{c}(J)$*.*

*(iv)* $x(J)$ *has the following alternative definition:*

$$\begin{aligned} x(J) = \textit{argmin} \quad & x^T x \\ \textit{s.t.} \quad & A_J^T A x = A_J^T b \\ & x \in \mathbb{R}^J. \end{aligned}$$

*(v)* $t(J)$ *has the following alternative definition:*

$$\begin{aligned} t(J) = \textit{argmin} \quad & x^T x \\ \textit{s.t.} \quad & A_J^T A x = \bar{c}(J)_J \\ & x \in \mathbb{R}^J. \end{aligned}$$

*Proof.* $(i)$ *Theorem* 3.1 state that $2t(J) = \lambda^T A_J^T$, for some vectors $\lambda \in \mathbb{R}^{|J|}$ and $\lambda = 2t(J)_J$ is the shortest such vector by definition of $t(J)$.

$(ii)$ If $b(J) \neq 0$, the optimality of $b(J)$ under a strictly convex function yields

$$(b - b(J))^T (b - b(J)) < (b - \mathbf{0})^T (b - \mathbf{0}) = b^T b.$$

The inequality $2b^T b(J) > b(J)^T b(J) \geq 0$ follows. The argument for $c(J)$ is the same.

$(iii)$ We only give the argument for $b$, the one for $c$ is analogous. First of all, notice that the program defining $\bar{b}(J)$ restricted to $(x_J|y)$, can be written as the program $CP(I', J')$, by choosing properly $J'$ and $I'$:

$$\begin{aligned} \text{min} \quad & (b - y)^T (b - y) \\ \text{s.t.} \quad & (A| - \mathbb{I})(x|y) = \mathbf{0} \\ & x \in \mathbb{R}^{J'} \\ & x_{J' \setminus I'} \geq \mathbf{0}. \end{aligned} \tag{3.16}$$

The notation $(\cdot|\cdot)$ means the juxtaposition of two matrices or two vectors. Now, the vector coefficients of $(x|y)$ are $n+m$, therefore $J' \subseteq [n+m]$. For obtaining exactly the program $\bar{b}(J)$ we have to leave out the constraints $x_{J' \setminus I'} \geq \mathbf{0}$ by imposing $J' = I'$ and to make the entries of $y$ free we choose $J' := J \cup [n+$

$1, n+m]$. With such $J'$ and $I'$ we have obtained $\bar{b}(J)$ written in the $CP(I', J')$ way. Now, by using *Theorem* 3.1 we can show that

$$A_J x^* = \bar{b}(J),$$
$$(\mathbf{0}^T | 2(\bar{b}(J) - b)) = \lambda^T (A_J | - \mathbb{I}).$$

The former comes from the $(i)$ optimality condition, in fact, by taking

$$(A| - \mathbb{I})(x^* | y^*) = \mathbf{0}$$

and $(x^*|y^*) \in \mathbb{R}^{J'}$, we can easily notice that, with $y^* = \bar{b}(J)$,

$$Ax^* = \bar{b}(J),\ x^* \in \mathbb{R}^J \Rightarrow A_J x^* = \bar{b}(J).$$

The latter comes form the $(ii)$ condition: we can compute directly

$$\nabla f\left((x^*|y^*)\right) = \nabla\left[(b - y^*)^T (b - y^*)\right] = -2(b - y^*) = 2(\bar{b}(J) - b)$$

and the condition holds for every $j \in J'$, so we obtain exactly the expression above. Now, by setting $\nu^* = \lambda/2$ and $\mu = -2x^*$ we have that $\nu^* = b - \bar{b}(J)$, $A_J^T \nu^* = \mathbf{0}$ and $2(\nu^* - b) = -2\bar{b}(J) = \mu^T A_J^T$. This means that $\nu^*$ and $\mu$ satisfy *Theorem* 3.1 conditions for the program defined by $b(J)$. Then $b(J) = \nu^* = b - \bar{b}(J)$ follows.

$(iv)$ In proving $A_J^T \nu^* = \mathbf{0}$ in $(iii)$ we have shown that $A_J^T \bar{b}(J) = A_J^T b$. Since any feasible linear system $Mx = q$ is equivalent to $M^T M x = M^T q$, we know that the system $A_J x_J = \bar{b}(J)$ that yields $x(J)_J$ can be replaced by $A_J^T A_J x_J = A_J^T \bar{b}(J) = A_J^T b$.

$(v)$ Knowing from $(i)$ that $A_J x_J = y(J)$ is feasible, we can replace it by $A_J^T A_J x_J = A_J^T y(J) = \bar{c}(J)_J$.

□

Here is the series expansion.

**Theorem 3.3.** *For $J \subseteq [n]$, define*

$$g_{-1}(J) = c(J),$$
$$g_0(J) = x(J),$$
$$g_1(J) = t(J),$$

*and for $i \geq 2$,*

$$\begin{aligned} g_i(J) \quad = \quad \textit{argmin} & \quad x^T x \\ \textit{s.t.} & \quad A_J^T A x = -g_{i-2}(J)_J \\ & \quad x \in \mathbb{R}^J \end{aligned} \tag{3.17}$$

*Then for any $k \geq 1$,*

$$x_\epsilon^*(J,J) = \sum_{i=-1}^{k} \epsilon^i g_i(J) + O(\epsilon^{k+1}),$$

*where the big-$O$ notation refers to the asymptotic behavior for $\epsilon \to 0$.*

With *Lemma* 3.3 $(iv)$ and $(v)$, any $g_i(J)$, $i \geq 0$, is by *Theorem* 3.1 of the form $2g_i(J)_J^T = \lambda^T A_J^T A_J$ for some $\lambda$. This guarantees that the programs defining the $g_i(J)$ with $i \geq 2$ are feasible, so all the $g_i(J)$ are indeed well-defined.

*Proof.* Let us write $x_\epsilon(J,J)$ in the form

$$x_\epsilon(J,J) = \sum_{i=-1}^{k} \epsilon^i g_i(J) - r_\epsilon(J), \tag{3.18}$$

with $r_\epsilon \in \mathbb{R}^J$ the remainder term. The fact that $x_\epsilon(J,J)_J$ solves (3.13) implies that $r_\epsilon(J)$ must be the unique solution to the system

$$\left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) x_J = \epsilon^{k+1} g_{k-1}(J)_J + \epsilon^{k+2} g_k(J)_J.$$

To see this plug (3.18) into (3.13)

$$\begin{aligned} 0 &= \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) x_\epsilon(J,J)_J - A_J^T b - \epsilon c_J \\ &= \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) \left( \sum_{i=-1}^{k} \epsilon^i g_i(J)_J - r_\epsilon(J)_J \right) - A_J^T b - \epsilon c_J \end{aligned}$$

and, eventually, we can read the part of $x_\epsilon(J,J)_J$ as

$$\sum_{i=-1}^{k} \epsilon^i g_i(J)_J - r_\epsilon(J)_J = \epsilon^{-1} c(J)_J + x(J)_J + \epsilon t(J)_J + \sum_{i=2}^{k} \epsilon^i g_i(J)_J - r_\epsilon(J)_J.$$

Then by using $A_J c(J)_J = 0$ and $A_J^T A_J x(J)_J = A_J^T b$ (*Lemma* 3.3 $(v)$) we can compute

$$\begin{aligned}\left(A_J^T A_J + \epsilon^2 \mathbb{I}\right)\epsilon^{-1} c(J)_J &= 0 + \epsilon c(J)_J\\ \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) x(J)_J &= A_J^T b + \epsilon^2 x(J)_J\\ \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right)\epsilon t(J)_J &= \epsilon \bar{c}(J) + \epsilon^3 t(J)_J.\end{aligned}$$

By using $c = c(J) + \bar{c}(J)$ (*Lemma* 3.3 $(iii)$) we see that some terms is cancelled. Finally

$$\begin{aligned}&\epsilon^2 x(J)_J + \epsilon^3 t(J)_J + \sum_{i=2}^{k} \epsilon^i A_J^T A_J g_i(J)_J + \epsilon^{i+2} g_i(J)_J - \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) r_\epsilon(J)_J =\\ &\underset{(a)}{=} \sum_{i=2}^{k} -\epsilon^i g_{i-2}(J)_J + \sum_{i=2}^{k+2} \epsilon^i g_{i-2}(J)_J - \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) r_\epsilon(J)_J =\\ &= \epsilon^{k+1} g_{k-1}(J)_J + \epsilon^{k+2} g_k(J)_J - \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right) r_\epsilon(J)_J = 0,\end{aligned}$$

where in $(a)$ we have exploited the fact that $g_k(J)$ is the optimal solution of (3.17), in particular it is a feasible solution. In other words we can state that $r_\epsilon(J)_J$ is in the form

$$\epsilon^{k+1}\left(\left(A_J^T A_J + \epsilon^2 \mathbb{I}\right)^{-1}\left(g_{k-1}(J)_J + \epsilon g_k(J)_J\right)\right).$$

If we can show that for all $i \geq 0$,

$$s_\epsilon := \left(A_J^T A_J + \epsilon^2 \mathbb{I}\right)^{-1} g_i(J)_J$$

converges as $\epsilon \to 0$, we have shown $r_\epsilon(J) = O\left(\epsilon^{k+1}\right)$. By the remark preceding this proof, we can write this system as

$$\left(Q + \epsilon^2 \mathbb{I}\right) s_\epsilon = Q\lambda$$

for some vector $\lambda$ and symmetric matrix Q. Choose a diagonalizing transformation $P$ such that $Q = P^{-1} D P$, where $D$ is a diagonal matrix with diagonal entries $a_1, \ldots, a_l, 0, \ldots, 0$, the first $l$ of them non zero.Then the matrix equation can be rewritten as

$$P^{-1}(D + \epsilon^2 \mathbb{I}) P s_\epsilon = P^{-1} D P \lambda,$$

which in turn is equivalent to $(D + \epsilon^2\mathbb{I})s'_\epsilon = D\lambda'$, with $s'_\epsilon = Ps_\epsilon$, $\lambda' = P\lambda$. We then get

$$s'_\epsilon = \left(\frac{a_1\lambda'_1}{a_1 + \epsilon^2}, \ldots, \frac{a_l\lambda'_l}{a_l + \epsilon^2}, 0, \ldots, 0\right),$$

meaning that $s'_\epsilon$ and therefore $s_\epsilon = P^{-1}s'_\epsilon$ converges.

□

This theorem states that we are able to read off the edge orientations in the LP-induced USO from the first non zero coefficient in the power series expansion. More precisely,

**Corollary.** *Let $J \in [n]$, $j \in J$ and set $I := J \setminus \{j\}$. Furthermore, define*

$$i(J, j) := \min\{i \geq -1 | \boldsymbol{g}_i(J)_j \neq 0\}.$$

*Then $i(J, j) = \infty$ or $i(J, j) \leq 2|J| - 1$, and the LP-induced USO $\rightarrow$ derived in the above Lemma induces the edge orientation*

$$I \rightarrow J \iff \boldsymbol{g}_{i(J,j)_j} > 0, \tag{3.19}$$

*where we set $\boldsymbol{g}_\infty(J)_j := 0$.*

Since our power series expansion also induces an expansion of $\nabla f(x^*_\epsilon(J, J))$, we can compute the orientation of all the edges incident to a given vertex $J$ in the LP-induced USO by solving at most $2|J| + 2$ unconstrained quadratic programs, and hopefully, much less in most cases. This occurs to be our vertex evaluation oracle. As we spotted before for compute these solutions it suffices to solve linear systems.

*Proof.* We have

$$I \overset{\epsilon}{\rightarrow} J \iff x^*_\epsilon(J, J)_j > 0.$$

Then, according to *Theorem* 3.3 the first non zero value $g_i(J)_j$ determines the sign that defines the orientation $I \rightarrow J$ in the limiting USO. For the bound on $i(J, j)$ in the finite case, recall that the numerator contains a monomial $\epsilon^i$ with $i \leq 2|J| - 1$, this is Cramer's rule applied to the system (3.13). It follows that

$$|x^*_\epsilon(J, J)_j| = \Omega\left(\epsilon^{2|J|-1}\right)$$

for $\epsilon \rightarrow 0$, and the previous theorem implies that $i(J, j) \leq 2|J|-1$. On the other hand, $i(J, j) = \infty$ implies $x^*_\epsilon(J, J)_j = 0$, so (3.19) gives the right orientation also in this case.

□

In conclusion, after computing all the orientation and the sink of the LP-induced USO we prove that the vertex $S \subseteq [n]$, which is the unique sink of the USO, tells us the solution of the LP considered.

Before seeing how the sink characterize the solution, we highlight the following result:

From *Theorem* 3.3 we know that

$$x^*_\epsilon(S,S) = \frac{c(S)}{\epsilon} + x(S) + \epsilon t(S) + O(\epsilon^2), \tag{3.20}$$

which implies

$$\frac{\nabla f(x^*_\epsilon(S,S))^T}{2} = A^T(\bar{b}(S) - b) + \epsilon(A^T y(S) - \bar{c}(S)) + O(\epsilon^2) \tag{3.21}$$

using (3.13), $Ac(S) = \mathbf{0}$, $Ax(S) = \bar{b}(S)$, $At(S) = y(S)$ and $c = c(S) + \bar{c}(S)$ (*Lemma* $(iii)$).

For sufficiently small $\epsilon$, S is also the sink in $\xrightarrow{\epsilon}$, so $x^*_\epsilon(S,S) = x^*_\epsilon(\varnothing,[n])$. Using the optimality criteria (3.3), (3.5) and (3.6), we deduce

$$x^*_\epsilon(S,S) \geq \mathbf{0}, \tag{3.22}$$
$$\nabla f(x^*_\epsilon(S,S)) \geq \mathbf{0}, \tag{3.23}$$
$$\nabla f(x^*_\epsilon(S,S))_j x^*_\epsilon(S,S)_j = 0, \quad j \in [n] \tag{3.24}$$

And, finally, we can derive

**Theorem 3.4.** *Consider the linear program*

$$\begin{aligned} \textit{max} \quad & \bar{c}(S)^T x \\ \textit{s.t.} \quad & Ax = \bar{b}(S) \\ & x \geq \boldsymbol{0}, \end{aligned} \tag{3.25}$$

*along with its dual*

$$\begin{aligned} \textit{min} \quad & \bar{b}(S)^T y \\ \textit{s.t.} \quad & A^T y = \bar{c}(S). \end{aligned} \tag{3.26}$$

*For sufficiently small $\epsilon > 0$, the following statements hold.*

*(i)* $x(S) + c(S)/\epsilon$ *is optimal for (3.25).*

*(ii)* $y(S) - b(S)/\epsilon$ *is optimal for (3.26).*

*Proof.* By putting together (3.20) and (3.22) show that $x(S) + c(S)/\epsilon \geq 0$, and feasibility for $(i)$ follows from the definition of $c(S)$ and $x(S)$. Analogously, by combining (3.21) and (3.23) we deduce that $y(S) - b(S)/\epsilon$ is feasible for $(ii)$; for this we recall $b(S) = b - \bar{b}(S)$ by *Lemma* 3.3.

To prove optimality, we argue as follows. From (3.24), we see that if $x(S)_j + c(S)_j/\epsilon > 0$, then $A_j^T(\bar{b}(S) - b) = -A_j^T b(S) = \mathbf{0}$ and $A_j^T y(S) - \bar{c}(S)_j = \mathbf{0}$, since otherwise, the lower order terms of (3.20) and (3.21) cannot contribute enough to reach complementarity in coordinate $j$ for small $\epsilon$.

The latter observation implies the complementary slackness condition for the pair of feasible solutions in $(i)$ and $(ii)$, and this shows that both are optimal in their respective programs. □

Finally we can prove the relation between the sink of the USO and the solution of the LP that induces it. In the following let us denote as *unbounded ray* for an LP an halfline whose tail is feasible and on which the objective function is unbounded.

**Theorem.** *Let $S$ be the sink of the USO induced by the LP*

$$\begin{aligned} \textit{max} \quad & c^T x \\ \textit{s.t.} \quad & Ax = b \\ & x \geq \mathbf{0}. \end{aligned}$$

*Then,*

*(i) If $\bar{b}(S) \neq b$, the LP is infeasible. Equivalently, the LP*

$$\begin{aligned} \textit{max} \quad & \bar{c}(S)^T x \\ \textit{s.t.} \quad & Ax = b \\ & x \geq \mathbf{0}. \end{aligned}$$

*is infeasible, and this is witnessed by the fact that*

$$\{y(S) - \frac{b(S)}{\epsilon} | \epsilon > 0\}$$

*is an unbounded ray of the dual problem*

$$\begin{aligned} \textit{min} \quad & b^T y \\ \textit{s.t.} \quad & A^T y = \bar{c}(S). \end{aligned}$$

*(ii) If $\bar{b}(S) = b$ and $\bar{c}(S) \neq c$, the LP is feasible but unbounded, and this is due to the fact that*

$$\{x(S) + \frac{c(S)}{2\epsilon} | \epsilon > 0\}$$

*is an unbounded ray of the LP.*

*(iii) If $\bar{b}(S) = b$ and $\bar{c}(S) = c$, then $x(S)$ and $y(S)$ is a pair of primal and dual optimal solutions o the LP.*

*Proof.* By weak duality, the existence of a dual unbounded ray implies infeasibility of the primal problem, so in order to show $(i)$ and $(ii)$, it remains to prove that the given rays are unbounded. But it follows form $c^T c(S) > 0$ and $b^T b(S) > 0$, see *Lemma* 3.3 $(ii)$. Property $(iii)$ is a corollary of the previous theorem, under $b(S) = b - \bar{b}(S) =$ **0** and $c(S) = c - c(S) =$ **0**. □

# Chapter 4

# Searching quantum algorithms

Finally we are going to present the results of our research. In the following two sections are showed two different quantum algorithms for solving unique sink orientation problems. By exploiting the technique and the ideas explained in the previous chapters we are able to build quantum algorithms that outperforms the best known classical algorithms for solving the USO problem. For the sake of completeness we consider the best classical result. The best known classical algorithm that solves this problem queries the oracle $O((1.467\ldots)^n)$ times, where $n$ is the dimension of the hypercube. It is straightforward to see that the Grover's search algorithm for solving a USO problem on an $n$-hypercube needs to query the oracle “only" $O(\sqrt{2^n}) = O((1.141\ldots)^n)$ times, which is already a better result than the best known classical algorithm.

## 4.1 Local Grover's search

This is a deterministic quantum algorithm, i.e. the probability of succeed is $1$, based on the method of amplitude amplification, showed in Chapter 1.

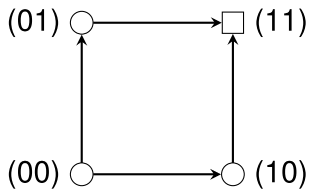
Let us recall it briefly: if we have a quantum algorithm $\mathcal{A}$ that uses no measurements and we know the probability $a > 0$ of measuring a specific state then there exists a quantum algorithm that finds the state with certainty using a number of applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$ which is $\Theta(\frac{1}{\sqrt{a}})$ in the worst case. Therefore, the basic idea is to find an $\mathcal{A}$ such that the desired state can be measured with a suitable probability.

The algorithm solves USOs on $4$-hypercubes, so let us consider a general $\mathcal{C}_4^{\phi}$. First of all, we take into account all of the nodes at once. Hence, we create the superposition of all the states that represent the nodes of $\mathcal{C}_4^\phi$. For this purpose Hadamard gates are a suitable choice, because we known that the probability of measuring any nodes in the superposition created by these gates is uniform. In fact, by considering the superposition of $2^n$ states,

$$H^{\otimes n}|\mathbf{0}\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle,$$

we can notice that the likelihood of measuring the state that represent the sink, that we denote as sink state, is exactly $1/2^n$. With the knowledge of the probability we can use amplitude amplification.

The local Grover search applies twice the amplitude amplification. The first time it is used to solve a general $\mathcal{C}_2^\phi$. Let us take the below figure as an example.



For solving this USO problem we need a quantum register of $2$ qubits for representing the nodes. We follow the amplitude amplification procedure. First of all, we compute the superposition of all the vertices:

$$|\psi\rangle = H^{\otimes 2}|00\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right),$$

Notice that $\mathcal{A} = H^{\otimes 2}$ is an invertible algorithm and the probability of measuring the sink is $a = 1/4$. First of all, we define the operator $Q := -R_{|\psi\rangle}R_{|\bar{x}\rangle}$, where

$$R_{|\bar{x}\rangle} = \mathbb{I}_2 - 2|11\rangle\langle 11|$$

is the reflection about the hyperplane perpendicular to the sink state and

$$R_{|\psi\rangle} = \mathcal{A}R_{|\mathbf{0}\rangle}\mathcal{A}^{-1} = \mathbb{I}_3 - 2|\psi\rangle\langle\psi|,$$

is the reflection about the hyperplane perpendicular to initial state, which is the superposition of all the nodes. We denote as $\mathbb{I}_n$ the identity unitary transformation on $n$ qubits and $R_{|\mathbf{0}\rangle} = \mathbb{I}_n - 2|\mathbf{0}\rangle\langle\mathbf{0}|$. Notice that for implementing the operator $R_{|\bar{x}\rangle}$ we have to know the sink state $|11\rangle$ a priori. Of course this would make this algorithm useless. We can fix this problem by means of an oracle. Indeed, by adding an auxiliary register of size $n$, we define

$$U_\phi|x\rangle|\mathbf{0}\rangle := |x\rangle|\mathbf{0} \oplus s(x)\rangle = |x\rangle|s(x)\rangle,$$

and, eventually we can apply a reflection $R_{|\mathbf{0}\rangle}$ on the auxiliary register for changing the relative phase of the sink state. Indeed, by definition the only vertex $x$ such that $s(x) = \mathbf{0}$ is the unique sink. For the operator $U_\phi$ we do not need to know the sink state because we use only the outmap $s$. For instance, in the case of LP-induced USO it can be computed by solving linear systems. As an example we show this procedure for the $2$ USO we are analyzing. We start from the superposition

$$\frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)|00\rangle,$$

we apply the oracle $U_\phi$

$$\frac{1}{2}\left(|00\rangle|11\rangle + |01\rangle|01\rangle + |10\rangle|10\rangle + |11\rangle|00\rangle\right)$$

and, at the end, the reflection $R_{|\mathbf{0}\rangle}$ on the auxiliary qubits

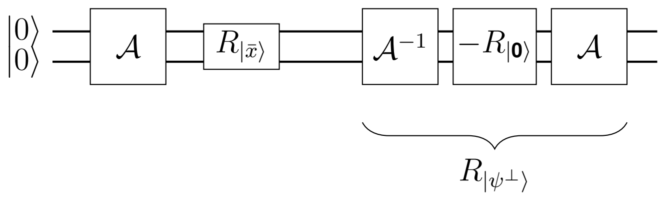
$$\frac{1}{2}\left(|00\rangle|11\rangle + |01\rangle|01\rangle + |10\rangle|10\rangle - |11\rangle|00\rangle\right).$$

Notice that we obtain the same action of the reflection $R_{|\bar{x}\rangle}$ without any knowledge about the sink state.

For the sake of simplicity and for maintaining a consistent notation with the one used for amplitude amplification, we continue to use $R_{|\bar{x}\rangle}$ for flipping the phase of the sink state.

Eventually, we apply the operator $Q^m$, with $m = \pi/4\theta - 1/2 = 1$, where $\theta$ is such that $\sin^2(\theta) = a$.

Therefore by applying the circuit



$(C)$

we measure the sink state $|11\rangle$ with certainty. The reflection $R_{|\psi^\perp\rangle} = -R_{|\psi\rangle}$ is about the state $|\psi\rangle$.

Now, we apply for the second time the amplitude amplification. Before proceeding, we recall the heritage property of USO hypercubes, which is crucial for understanding this step. Given the outmap $s$ of the orientation $\phi$ and $A \subseteq [n]$ a subset of directions, if we solve every USO problem on all the faces generated by the directions $A$, then for finding the general sink it suffices to find the unique sink in a lower dimensional USO problem. Let us show how we can exploit this property. We start from the superposition of all the vertices:

$$|\psi\rangle = \frac{1}{4}\sum_{i=0}^{15} |i\rangle = \frac{1}{2}\sum_{i'=0}^{3} |i'\rangle \otimes \frac{1}{2}\sum_{i=0}^{3} |i\rangle. \tag{4.1}$$

Each state $|i\rangle := |i_1 \cdots i_4\rangle$ represents a direction on the hypercube. Therefore, by dividing the qubits as in equation (4.1), we are choosing the subset of directions $A$. In this case $A = \{3, 4\}$ and the qubits taken in account are $|i_3\rangle$ and $|i_4\rangle$.

The idea is to solve locally the USO problem on the faces generated by $A$. For succeeding in this it is sufficient to apply the same procedure showed before for the $2$-hypercube. In fact, if we focus on the second register we can see that we can use amplitude amplification. In fact circuit (C) is an invertible algorithm and the probability of measuring the local sink state for each face is $a = 1/4$. Moreover, if we rewrite the expression (4.1) as

$$\frac{1}{2}\left[|00\rangle \otimes \frac{1}{2}\sum_{i=0}^{3} |i\rangle + |01\rangle \otimes \frac{1}{2}\sum_{i=0}^{3} |i\rangle + |10\rangle \otimes \frac{1}{2}\sum_{i=0}^{3} |i\rangle + |11\rangle \otimes \frac{1}{2}\sum_{i=0}^{3} |i\rangle\right],$$

notice that the first two qubits identifies the local USO problem we have to solve. In *Figure* 4.1 there is an example of $\mathcal{C}_4^\phi$ where the faces generated by the set of direction $A$ are spotted. Therefore, by applying the amplitude amplification operator $Q$ on the two last qubits we are solving the USO problem on a superposition of $2$-hypercubes. The result is a superposition of the sinks of the faces spanned by $A$:

$$\frac{1}{2}\left[|00\rangle|\text{sink}_{00}\rangle + |01\rangle|\text{sink}_{01}\rangle + |10\rangle|\text{sink}_{10}\rangle + |11\rangle|\text{sink}_{11}\rangle\right], \tag{4.2}$$

where $|\text{sink}_{ij}\rangle$ are the local sink states.

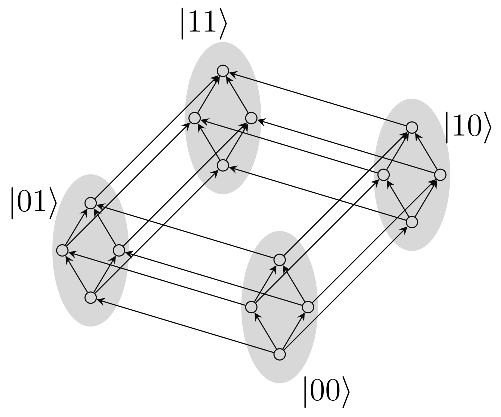
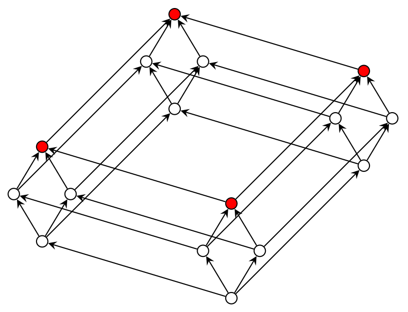


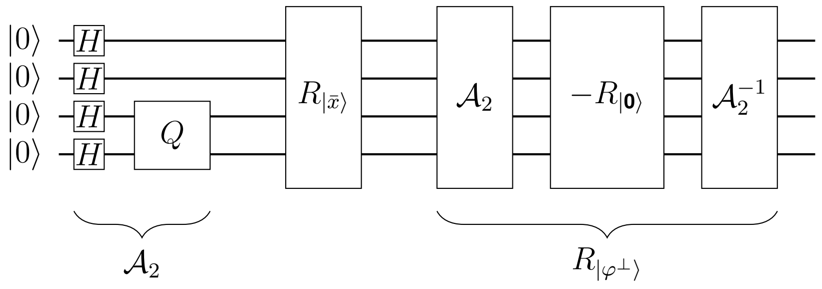
Figure 4.1: Example of a $4$-hypercube USO where the faces highlighted are generated by the superposition of the last two qubits. We stress that the first two qubits represent the different faces, that is, geometrically we are considering $2$-hypercubes that belong to different subspaces of the quantum system Hilbert space .

By referring to the Figure 4.1 this step could be represented graphically by



where state (4.2) is exactly the superposition of the red vertices.

Now we are ready to proceed with the amplitude amplification step. By the definition of USO on cubes we are sure that in the sinks superposition there is the global sink state which represents the unique sink of $\mathcal{C}_4^{\phi}$. Moreover, we know exactly that after a measurement we find the sink state as an outcome with probability $a = 1/4$. Since we know $a$ and since we have not used any measurements for solving the $2$-hypercubes, we can apply amplitude amplification again. Therefore, we define a new $Q_2 := -R_{|\varphi\rangle}R_{|\bar{x}\rangle}$, where $|\varphi\rangle := \mathcal{A}_2|\mathbf{0}\rangle$ and $R_{|\bar{x}\rangle}$ is the reflection about the hyperplane orthogonal to the sink state. $\mathcal{A}_2$ is the algorithm that has as output (4.2). As the previous step, after $m = 1$ applications the method leads to the result. Then the circuit that represents the procedure presented is:



At the end we measure the unique sink of the whole $\mathcal{C}_4^\phi$ with certainty.

Now, we show the query complexity of the algorithm. For obtaining it we count the number of times in which the oracle operator $U_{\phi_i}$ is applied. As we stressed before we query the oracle at every application of the reflection $R_{|\bar{x}\rangle}$. The first part $\mathcal{A}_2$ is equivalent to the circuit ($C$), thus we can easily see that we query the oracle only once. For the second part we can take into account the two reflection separately. Straightforwardly in $R_{|\bar{x}\rangle}$ we query the oracle one time. Instead, in the second reflection $R_{|\psi^\perp\rangle}$ there is another application of the algorithm $\mathcal{A}_2$ and one application of its inverse, $\mathcal{A}_2^{-1}$. Thus, for what we have said above about the query complexity of the algorithm $\mathcal{A}_2$, the reflection $R_{|\psi^\perp\rangle}$ queries two times the oracle. Therefore by denoting $T(n)$ the query complexity of the $n$-hypercube USO, we can sate

$$T(4) = \underbrace{T(2)}_{\mathcal{A}_2} + 1 + \underbrace{2T(2)}_{R_{|\psi^\perp\rangle}} = 3T(2) + 1 = 3 \cdot 1 + 1 = 4. \tag{4.3}$$

We can see that the local Grover search outperforms the best known classical algorithm that finds the unique sink in a USO. Indeed, the best classical algorithm queries to the oracle $O((1.467\ldots)^n)$ which is about $4.6314\ldots$ which is greater that $4$. Unfortunately, this is not the best algorithm for finding the unique sink in a USO. In fact, as we said in *Chapter* 1, the query complexity of Grover's search algorithm is

$$\frac{\pi}{4}\sqrt{N} - \frac{1}{2},$$

which is $2.641\dots$ and its ceiling is $3$. Therefore, it is the best known algorithm for finding the unique sink in $4$-hypercubes.

In conclusion, we showed two different way to find the unique sink in the $4$-hypercube with less queries than the best known classical algorithm. Both of these quantum algorithms are interesting: the latter because is the best algorithm we know for solving USO problem; the former because is obtained by exploiting the heritage property of the USO.

## 4.2 Controlled sink sieve

This quantum algorithm exploits the properties of the outmap $s$ of a USO on a cube. In the presentation of this algorithm we use the "combinatorial view" of hypercubes.

As we have seen in *Chapter* 3, given a LP with $n$ variables we can define a USO on an $n$-hypercube. For understanding such an orientation we take into account any nodes at a time and to compute the outgoing edges from that vertex by solving several linear systems. Notice that, at the end, we obtain a vector which entries represents the edge orientations. In fact, whether an entry of such vector is strictly greater then zero, then the edge is an ingoing one, otherwise it is an outgoing one. Hence, by taking a general vertex $w$ and its vector that corresponds to the orientation, $u = (u_i)_{i=1,\dots,n}$, we can define a map

$$w \longmapsto v = (v_i)_{i=1,\dots,n}, \text{ where } v_i = \begin{cases} 0 & \text{if } u_i > 0 \\ 1 & \text{otherwise.} \end{cases}$$

What we obtain at the end is a vector whose entries are $1$ in the possible outgoing directions and $0$ otherwise. Notice that this is the outmap definition in the "combinatorial view".

With the above definition we can use the map $s$ as an oracle which takes as inputs the vertices and outputs the edge orientations as vectors which entries are only $1$ or $0$.

The basic idea of the *controlled sink sieve* is tho exploits the bijectivity of the outmap. Given a USO on a cube $\mathcal{C}^\phi$ and its sink $o \in V(\mathcal{C})$, we have that it is the only vertex such that

$$s(o) = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Therefore, we can implement a trivial quantum algorithm. Initialize two quantum registers of $n$-size. In the first one we represent the vertices of the cube and in the second one we store the output of the outmap $s$, which is a vector of $n$ entries. So, we apply an Hadamard gate at each qubit of the first register

$$\left(H^{\otimes n} \otimes \mathbb{I}_n\right) |\mathbf{0}\rangle|\mathbf{0}\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle\right) |\mathbf{0}\rangle \tag{4.4}$$

and, then, we apply the outmap $s$ implemented by means of the unitary operator $U_s$, which acts like $U_s|v\rangle|\mathbf{0}\rangle = |v\rangle|\mathbf{0} \oplus s(v)\rangle = |v\rangle|s(v)\rangle$. After applying $U_s$ to the state (4.4), we obtain

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle|s(i)\rangle = \frac{1}{\sqrt{2^n}} \left[\sum_{i \neq o} |i\rangle|s(i)\rangle + |o\rangle|\mathbf{0}\rangle\right]. \tag{4.5}$$

Now, if we measure the first register we find the unique sink $o$ with probability $1/2^n$. Notice that if we measure the second register we obtain the vector **0** with probability $1/2^n$ as well. And, furthermore, if we have measure $o$ in the first register or **0** in the second one, then we are sure to measure **0** or $o$ respectively in the other register, because after the first measurement the state collapses. Therefore the problem of finding **0** in the second register is equivalent to the USO problem and this is due to the definition of $s$ for a USO.

Our current goal now is to create a quantum algorithm for measuring **0** in the second register. For this purpose, we act on the second register maintaining unchanged the link between a vertex and its outmap output. In fact, the easiest way for having in the second register the vector **0** is to measure each qubit and then apply a $\sigma_x$ gate for changing the state if the measure gave $|1\rangle$. Notice that without any action on the first register the above procedure fails, for example in a $1$-hypercube

$$\begin{tikzcd}(0)\ \bigcirc \arrow[r] & \square\ (1)\end{tikzcd}$$

by starting from the superposition as (4.5)

$$|0\rangle|1\rangle + |1\rangle|0\rangle \xrightarrow{\text{measure } |1\rangle} |0\rangle|1\rangle \xrightarrow{\text{apply } \mathbb{I}\otimes\sigma_x} |0\rangle|0\rangle.$$

In the first step we have measured the second register with outcome $|1\rangle$ and, therefore, the state collapses in the state $|0\rangle|1\rangle$. In the second step we apply the gate $\sigma_x$ to the second register. At the end, by measuring the first register, we do not find the unique sink of the USO above. So, we define an algorithm that takes into account the connection between the two registers.

Notice that it is possible to change a specific bit on the second register by changing the correspondent bit on the representation of the vertex. In detail, if we are in the node $v \in V(\mathcal{C})$ and we know that in $s(v)$ the $j$-th entry is $1$, then we can flip the $j$-th entry of the node $v$. Afterwards, we are in the vertex $v \oplus e_j$ and the $j$-th entry of its outmap vector is $0$, because we moved from the head to the tail of the arc $(v, v \oplus e_j) \in A(\mathcal{C}^\phi)$. The problem that arises now is that for following the above procedure we are suppose to store in the second register the new vertex outmap output, i.e. we need to be able to reset the second register. Fortunately, the operator $U_s$, which is unitary and, thus, invertible, with inverse $U_s$ itself. In fact, for any nodes $v \in V(\mathcal{C})$, we have

$$U_s^2|v\rangle|\mathbf{0}\rangle = U_s|v\rangle|s(v)\rangle = |v\rangle|s(v) \oplus s(v)\rangle = |v\rangle|\mathbf{0}\rangle.$$

So, we are able to reset the second register by applying a second time the operator $U_s$.

Now, the procedure is the following: create the superposition as in (4.5) and measure the $j$-th qubit of the second register; if it is has as an outcome $|1\rangle$ then apply the oracle $U_s$ for resetting the second register, flip the $j$-th qubit in the first register and, eventually, apply again the operator $U_s$ for creating a superposition similar to (4.5).

We show this algorithm for the $1$-hypercube example above. We start with the superposition of all the vertices in the first register and their correspondent outmap evaluation on the second register

$$|0\rangle|1\rangle + |1\rangle|0\rangle.$$

Then we measure the qubit of the second register. If we measure $|0\rangle$ we have finished, because we have found the vector **0** on the second register. Otherwise we follow the procedure

$$|0\rangle|1\rangle \xrightarrow{\text{apply } U_s} |0\rangle|0\rangle \xrightarrow{\text{apply } \sigma_x \otimes \mathbb{I}} |1\rangle|0\rangle \xrightarrow{\text{apply } U_s \text{ again}} |1\rangle|0\rangle.$$

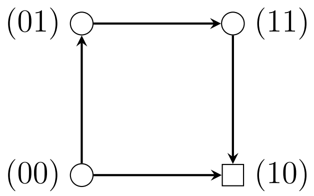
Eventually, we measure the second register and we find the vector **0**.

We generalize the algorithm for larger $n$.

```
Algorithm: Controlled sink sieve
1) We initialize the two quantum registers $|\mathbf{0}\rangle|\mathbf{0}\rangle$.
2) We compute the superposition as we have seen in (4.5).
3) We choose a qubit to be measured. Let us take the $j$-th one. If the
measurement outcome is $|0\rangle$ then pass to the next qubit, otherwise
      . Apply again $U_s$ for resetting the second register.
      . Apply a $\sigma_x$ operator to the $j$-th qubit of the first register.
      . Restore the superposition by applying again $U_s$.
Continue until all the qubits of the second register are measured.
4) Eventually, measure all the second register. If the result of the
measurement is the state $|\mathbf{0}\rangle$, then the first register is the sink of the
USO. Otherwise restart from $(1)$.
```

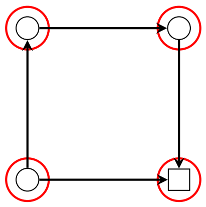
As we can see from the fourth step of the above algorithm might fail. In fact, by moving from a vertex to another we could change more than one entries of the outmap, not only the one considered. Let us show an example.

Let $\mathcal{C}_2^{\phi}$ be the following USO



At each step we will show the superposition of the nodes and we will mark which nodes are involved in the superposition in the graph. At the beginning we initialize the superposition like in (4.5)

$$\frac{1}{2}(|00\rangle|11\rangle+|01\rangle|10\rangle+|10\rangle|00\rangle+|11\rangle|01\rangle)$$

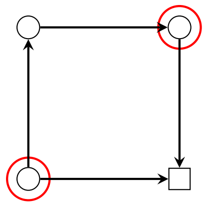


At this step all the vertices are still in the superposition because we have used no measurements yet. Now let us suppose that we want to measure the second qubit of the second register. For understanding what happens after the measurement let us consider the above superposition as

$$\frac{1}{2}[(|00\rangle|1\rangle + |11\rangle|0\rangle) \otimes |1\rangle + (|01\rangle|1\rangle + |10\rangle|0\rangle) \otimes |0\rangle].$$

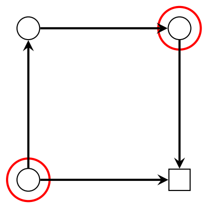
Therefore, if we measure it we have probability $1/2$ of having $|1\rangle$ and probability $1/2$ of having $|0\rangle$. Let us suppose to have measured the state $|1\rangle$. The state collapses after the measurement and we obtain

$$\frac{1}{\sqrt{2}}(|00\rangle|1\rangle + |11\rangle|0\rangle) \otimes |1\rangle$$



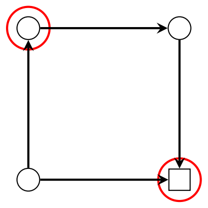
Now we reset the second register by applying $U_s$,

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)|00\rangle$$



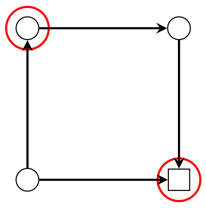
we flip the second qubit by using the $\sigma_x$ gate

$$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)|00\rangle$$



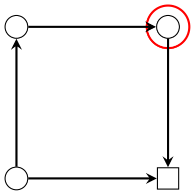
and, eventually, we restore the superposition with the operator $U_s$.

$$\frac{1}{\sqrt{2}}(|01\rangle|10\rangle + |10\rangle|00\rangle)$$



Finally, we repeat the same procedure for the first qubit of the second register and let us suppose that once again we have as an outcome the state $|1\rangle$. Then we obtain the following

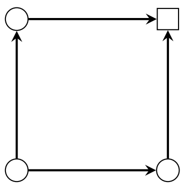
$$|11\rangle|01\rangle$$



This example is one of the possible failures that the algorithm can output. Notice that after measuring $|1\rangle$ the second time the state collapses into the state $|01\rangle|10\rangle$. As we stressed before, after moving from the tail to the head of the arc, both the qubits in the second register flip. The result state is $|11\rangle|01\rangle$.

From the example above we can understand that the choice of the order in which the qubits are measured is important. In fact, if we choose the inverse order in the above example, i.e. first we measure the first qubit and eventually the second one, the algorithm outputs the unique sink with probability $1$, although the possible outcomes of the measurements.

Unfortunately, we cannot insert in this thesis any discussion about the computational complexity of this algorithm. Indeed, the probability of success of this algorithm is strictly connected with the USO considered. For instance if we take into account the USO on the $2$-hypercube,



the controlled sink sieve in this case outputs the unique sink with likelihood $1$ for every choice of order and for any possible outcomes of the measurements.

Therefore, simulations are ongoing but we have not enough information for analyzing the probability of measuring the unique sink in a general USO on an $n$-hypercube.

Nevertheless, we present some considerations and some open problems for this algorithm:

- We can notice that after a run of the algorithm, its query complexity is at most $2n$, because every time we measure the state $|1\rangle$, we apply the oracle $U_s$ twice. Therefore, since the algorithm is probabilistic, if we consider $p$ as the probability of measuring the sink in the worst case scenario, then for finding the sink with almost certainty we run the algorithm for $O(1/p)$ times. So the algorithm would query the oracle $O(2n/p)$ times, in the worst case. If $p$ depends on $n$ we have a function that can be compared with the best known algorithm for solving a USO problem, which is the Grover's search algorithm. Therefore, the controlled sink sieve would be the best algorithm whether for large $n$

$$c \cdot \frac{2n}{p(n)} < \frac{\pi}{4}\sqrt{2^n},$$

  holds, where $c$ is a constant and the right hand term is the query complexity of the Grover's algorithm.

- Further analysis can be done also for what concerns the geometry of the USO. Starting from the fact that the success of the algorithm depends on the USO, one can wonder whether there exist particular classes of USO for which this algorithm is efficient. Furthermore, we can wonder under which conditions the algorithm fails.

# Conclusion

In this thesis, we presented two different quantum algorithms for solving the unique sink orientation problem on an hypercube. In both cases, for accessing to the information we needed an oracle mathematically defined as a surjective function $s$. The algorithmic complexity of these algorithms is determined via the number of times we call $s$ for solving the task.

The first algorithm, the *local Grover's search*, developed for the unique sink orientation on a $4$-hypercube, can find the unique sink by querying the oracle $4$ times. This result outperforms the best classical result that needs at most $7$ queries. Nevertheless, it is not the best known quantum algorithm: the Grover's search can find the sink (seen as a marked vertex in an undirected $4$-hypercube) with only $3$ queries. The algorithm is based on a quantum method called amplitude amplification. Amplitude amplification increases the probability of measuring a specific state in the superposition $|\psi\rangle$ created by applying an invertible quantum algorithm $\mathcal{A}$ to an initial state $|v\rangle$. This method for increasing the likelihood uses the operator $Q := R_{|\psi\rangle} R_{\phi}$, where $R_{\psi}$ is the reflection about the state $|\psi\rangle$ and $R_{\phi}$ is the reflection that has as unique eigenvector associated to the eigenvalue $-1$ the state that we are looking for.

The importance of the algorithm we proposed lies in the exploitation of the heritage property. This property states that we can divide the USO problem on an $n$-hypercube in smaller USO problems on cube of smaller dimension. Essentially the basic idea is to choose a suitable subset $A$ of directions and to solve all the USO spanned by $A$. After solving all of these problems if we look at the outgoing directions of all the sinks obtained from the smaller USO spanned by $A$ we obtain a new USO problem on an $(n - |A|)$-hypercube. The best classical algorithm is based on this idea, but it does not compute all of the sinks of the smaller problems because solving them classically requires checking all the nodes. On the other hand, our algorithm by exploiting superposition can solve all the problem spanned by $A$ just with one step.

The second algorithm we introduced, the *controlled sink sieve*, is based on the properties of the outmap $s$. The idea to exploit the quantum advantage is to take into account the whole directed cube. In this case we need only $2n$ qubits for implementing the vertices and a vector where the possible outgoing directions are stored. We get an advantage because in the classical case such implementation needs $2^{2n}$ strings of bits. The basic idea of the algorithm is to check the possible outgoing directions stored in the second register by measuring the qubits in a random order. When as an outcome we have the state $|0\rangle$ we continue by measuring another qubit. Otherwise, when we obtain the state $|1\rangle$, we reset the whole second register, flipping the correspondent qubit in the first register and then restore the outgoing edges information in the second register. This procedure is repeated until in the second register all the qubits are in the state $|0\rangle$.

The probability of succeeding and the computational cost need to be studied more in detail because this algorithm has different behavior for each unique sink orientation configuration. Extensive numerical simulation are ongoing but cannot be finished in time for the submission. Moreover, it is necessary an improved analysis for understanding whether this algorithm is suitable for only acyclic unique sink orientation, i.e. graphs with no cycle, instead of general ones.

Finally, although we are not able to show a more efficient algorithm yet, we think that it is worthy to continue in this direction. Indeed, it has been shown that Grover's search is optimal for a search on an unstructured database [16]. Therefore there cannot exist any algorithms that are asymptotically better then the Grover's algorithm. Nevertheless, because of their definition, USO on hypercube can be considered as structured database. So, a significant speed up could be achieved only by exploiting the USO properties.

Despite the USO problem is a possible key for finding a strongly-polynomial algorithm for the LP, the only quantum works for this topic are the result obtained by Bacon and an algorithm that recognize only whether an orientation of a cube is a USO or not [3]. Therefore to the last of our knowledge our work is the only one which presents specific quantum algorithms for solving USO problem on hypercubes.

# Appendix

## Basic notion of quantum mechanics

The quantum theory is based on the Hilbert space formalism. An Hilbert space is a vector space $X$ endowed with the metric $d$ such that the couple $(X, d)$ is a complete space. A quantum system is, indeed, a complex Hilbert space $\mathcal{H}$. We describe quantum state by means of unitary vectors of $\mathcal{H}$ and they from an orthonormal basis of the space.

In this thesis all the quantum systems taken into account are composed by a finite number of states, therefore we consider the following result:

**Theorem.** *Two Hilbert spaces are isomorphic if and only if they have the same dimension.*

In particular, any $N$-dimensional complex Hilbert space is isomorphic to $\mathbb{C}^N$, the complex vector space endowed with the standard scalar product. Therefore we represent the states by means of unitary complex vectors.

For instance, if we consider an half-spin particle, as an electron, the Hilbert space that describes the spin systems is formalised as:

$$\mathcal{H} = \text{Span}\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} = \text{Span}\{|0\rangle, |1\rangle\},$$

endowed with the standard complex scalar product:

$$\langle v, w \rangle = v^* w = \sum_{i=1}^{2} \bar{v}_i w_i = \langle v | w \rangle.$$

In the right side of the two equations above we introduced a new notation for the objects we are taking into account. This is called the bra-ket notation, which will be useful for making lighter the notation later.

Let us now formalize all this concepts by stating the postulates of quantum mechanics.

**Postulate 1.** *The state of a physical quantum system is completely described by a unity vector $|\psi\rangle$, which is called the state vector, or the wave function, and resides in the Hilbert space $\mathcal{H}$ associated with the system.*

The notation $|\psi\rangle$ is part of the bra-ket one introduced above. As you can see when we wrote the scalar product $\langle|\psi\rangle, |\psi\rangle\rangle$ the notation would be very heavy, therefore as explained above, we are going to identify $|\psi\rangle^* = \langle\psi|$ so that the scalar product can be easily seen as $\langle\psi|\psi\rangle$. Actually, we have different reasons for justifying the use of Dirac notation, but these exceed the purposes of this Appendix.

One of the advantages of using an Hilbert framework is the existence of basis. Therefore for describing a quantum system we can choose a basis of vectors $\mathcal{H} = \text{Span}\{|\phi_i\rangle\}_i$ and for each $|\psi\rangle \in \mathcal{H}$ we can rewrite it as

$$|\psi\rangle = \sum_i \alpha_i|\phi_i\rangle, \text{ where } \alpha_i = \langle\phi_i|\psi\rangle.$$

The coefficients of the above sum are called *amplitudes* and they play a main role on the description of physical system.

With this property we can highlight an important principle of quantum mechanics: the *superposition principle*. The idea is the following, if we take two independent possible state which describe our system $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{H}$, we could consider the new state

$$|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_2\rangle,$$

which is an allowed state whether $|\alpha|^2 + |\beta|^2 = 1$, because the new state $|\psi\rangle$ is a linear combination of the previous states and thus belongs in the vector space generated by them and moreover it has unitary norm, therefore it belongs in $\mathcal{H}$. Informally, one can think about a superposition of different states in this way: if we take into account again $|\psi\rangle$, described above, one could say that the system described by $|\psi\rangle$ is simultaneously in the state $|\psi_1\rangle$ and $|\psi_2\rangle$.

Now we have defined mathematically the basic objects we are interested in. The next step is to understand how to describe the dynamics of the system.

If we consider an Hamiltonian description of the energy of the system, i.e. if there exists a self-adjoint operator $\hat{H} : \mathcal{H} \to \mathbb{R}$, we state that

**Postulate 2.** *The evolution of a closed quantum system in time of the wave function $|\psi\rangle$ is governed by the Schrödinger equations:*

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle,$$

*where $i$ is the imaginary unit and $\hbar = h/2\pi$, with the Planck constant $h$ and $\hat{H}$ is a self-adjoint operator which describes the energy of the system.*

We must stress that Schrödinger equation is linear differential equation in time. Therefore, we are able to describe the complete dynamics of the system only by giving an initial state $|\psi_0\rangle$.

Moreover, by the linear framework we describe until now we also highlighted that by the superposition principle of the solutions of a linear differential equation whether $|\psi_1(t)\rangle$ and $|\psi_2(t)\rangle$ are two different solutions them also $|\psi(t)\rangle = \alpha|\psi_1(t)\rangle + \beta|\psi_2(t)\rangle$ is a solution. That means that $U(t)$ is linear.

Finally, Notice that if the Hamiltonian operator is time-independent then a solution of Schrödinger equation is

$$|\psi(t)\rangle = e^{-\frac{i}{\hbar}Ht}|\psi_0\rangle = U(t)|\psi_0\rangle,$$

where $|\psi_0\rangle$ is a give initial state. By this solution we spot that the operator $U(t)$ is the exponential map of a self-adjoint operator, therefore it is a unitary operator.

We pass now to the description of the quantum counterparts of the dynamical variables as *position*, *momentum* and so on. We define these physical objects as observables.

An *observable* in quantum mechanics is a physical dynamical quantity of the system that we can measure. We consider the *spectrum* of an observable $O$ as the set of values that arise from a measurement of $O$ to the system. All the measurements are real number and therefore the spectrum of an observable is a subset of $\mathbb{R}$.

**Postulate 3.** *We associate with any observables $O$ a self-adjoint operator $\hat{A}$ on the Hilbert space $\mathcal{H}$. The only possible outcome of a measurement of the* *observable $O$ is one of the eigenvalues of the operator $\hat{A}$, which has a non degenerate spectrum*[1]*.*

*By considering the eigenvalue equation for $\hat{A}$:*

$$\hat{A}|i\rangle = a_i|i\rangle,$$

*where $a_i \in \mathbb{R}$ is an eigenvalue and $\{|i\rangle\}$ form an orthonormal basis of eigenvectors of the operator $\hat{A}$, and we consider the Parseval identity we can rewrite any state vector $|\psi(t)\rangle$ as*

$$|\psi(t)\rangle = \sum_i \psi_i(t)|i\rangle,$$

*then the probability that by measuring the observable $O$ we get as an outcome $a_i$ is:*

$$P_i(t) = P(a = a_i|t) = |\langle i|\psi(t)\rangle|^2 = |\psi_i(t)|^2. \quad (4.6)$$

By these first postulates we can stress different consequences that can help us for understanding the purpose behind the choice of this formalization. We said that the outcomes of a measurement are real values and, then, eigenvalues of the operator $\hat{A}$ which describes the observable $O$. This appears suitable by taking into account the fact that $\hat{A}$ is self-adjoint, therefore its eigenvalues are real numbers. Moreover, they are an orthonormal basis of the whole Hilbert space $\mathcal{H}$ and, since $|\psi(t)\rangle$ is a unit vector, we have that

$$\sum_i P_i(t) = \sum_i |\psi_i(t)|^2 = 1,$$

where $P_i(t)$ and $\psi_i(t)$ are define in (4.6). Therefore the probabilities of getting any outcome of $O$ are normalized, i.e. the sum is equal to $1$. This is exactly the reason why we require states to have unit norm. Moreover, now it is easy to see that if a state $|\psi_0\rangle$ returns as an outcome $a_i$ with likelihood $1$ it means that

$$|\langle\psi_0|i\rangle|^2 = 1$$

but, now, we can notice that $|\langle\ |\ \rangle|$ is the inner product of $\mathcal{H}$, therefore to get $1$ is possible if and only if $|\psi_0\rangle = |i\rangle$, that is $|\psi_0\rangle$ is the eigenvector associated to

[1] In our discussion we are going to take into account only this kind of operator. Generally, this postulate is stated also for operator with degenerate eigenspaces.

the eigenvalue $a_i$. For this reason we would call the state $|i\rangle$ *eigenstate*.

Considering eigenstates is also important because it can let us understand deeper the nature of the superposition of the states. Let $|\psi_1\rangle$ and $|\psi_2\rangle$ be eigenstates respectively associates to the eigenvalues $a_1$ and $a_2$. The superposition principle tells us that the state

$$|\psi\rangle = \lambda_1|\psi_1\rangle + \lambda_2|\psi_2\rangle,$$

where $\lambda_1, \lambda_2 \in \mathbb{C}$ are such that $|\lambda_1|^2 + |\lambda_2|^2 = 1$, so that $|\psi\rangle$ has unit norm, is an allowed state of the quantum system described by $\mathcal{H}$. Therefore, if we measure the state $|\psi\rangle$, with an observable which has both $|\psi_1\rangle$ and $|\psi_2\rangle$ as eigenvectors, we measure $a_1$ with probability $|\lambda_1|^2$ and $a_2$ with probability $|\lambda_2|^2$. However, we stress that $|\psi\rangle$ is not equivalent to a statistical mixture of the state $|\psi_1\rangle$ and $|\psi_2\rangle$, taken with probabilities $|\lambda_1|^2$ and $|\lambda_2|^2$ respectively. Indeed let us consider $N$ identical physical system in the $|\psi\rangle$ state. If $|\psi\rangle$ is equivalent to a statistical mixture then it would be an ensemble of $|\lambda_1|^2 N$ systems in the state $|\psi_1\rangle$ and $|\lambda_2|^2 N$ systems in the $|\psi_2\rangle$ state. Though we are led to a different result if we compute the probability $P(b_i)$ of measuring $b_i$ as an outcome from $|\psi\rangle$ for some different observable[2] $\hat{B}$. Now from (4.6) we get that

$$P(b_i) = |\langle i|\psi\rangle|^2$$

where $|i\rangle$ is an eigenvector of $\hat{B}$ associated with $b_1$. Thus

$$\begin{aligned} P(b_i) &= |\lambda_1\langle i|\psi_1\rangle + \lambda_2\langle i|\psi_2\rangle|^2 = && (4.7)\\ &= |\lambda_1|^2|\langle i|\psi_1\rangle|^2 + |\lambda_2|^2|\langle i|\psi_2\rangle|^2 + 2\,\mathrm{Re}[\lambda_1\lambda_2^*\langle i|\psi_1\rangle\langle i|\psi_2\rangle^*]. && (4.8)\end{aligned}$$

But now if we compute the probability with a classical statistical mixture we obtain

$$P_{\mathrm{mix}}(b_i) = |\lambda_1|^2|\langle i|\psi_1\rangle|^2 + |\lambda_2|^2|\langle i|\psi_2\rangle|^2.$$

And so by comparing this with (4.7)

$$P(b_i) = P_{\mathrm{mix}}(b_i) + 2\,\mathrm{Re}[\lambda_1\lambda_2^*\langle i|\psi_1\rangle\langle i|\psi_2\rangle^*]. \quad (4.9)$$

This means that the predictions of the quantum mechanical theory depend on the complex number $\lambda_1$ and $\lambda_2$ not only in their modulus. The term on the

[2]From now on, without loss of generality, we identify the observables with their operator.

equation (4.9) which spot the difference between the quantum system and the statistical mixture is said *interference term*.

We now discuss the effect of a measurement on the state of the system. we can see measurements as an interactions of more physical systems. From this point of view after the interaction the systems have changed. Indeed if we measure the observable $\hat{A}$ that results in outcome $a_i$ and the measurement does not destroy the system, by applying immediately the same $\hat{A}$ we obtain again $a_i$ as outcome with probability $1$. For explaining this experimental phenomenon we need to accept that the state function $|\psi\rangle$ after being measured collapse onto the eigenstate $|i\rangle$ of $\hat{A}$ associated with the eigenvalue $a_i$. We now state the following postulate:

**Postulate 4.** *If a system is described by the vector $|\psi\rangle$ and we measure an observable $\hat{A}$, obtaining the outcome $a_i$, then immediately after the measurement the state of the system is given by*

$$\frac{P_i|\psi\rangle}{\sqrt{\langle\psi|P_i|\psi\rangle}},$$

*where $P_i$ is the projector operator over the eigenspace corresponding to $a_i$.*

Notice, now, that by measuring the observable $\hat{A}$, the likelihood of obtaining any outcome $a_i$ is given by

$$p_i = P(a_i) = \langle\psi|P_i|\psi\rangle$$

It is easy to see that it is true since *Postulate* 3.

Form probability theory we know that the average of an observable $\hat{A}$ is given by $\langle\hat{A}\rangle = \sum_i a_i p_i$. Moreover we can notice that since $\hat{A}$ is self-adjoint then by the spectral sum property we have

$$\hat{A} = \sum_i a_i P_i,$$

and therefore

$$\langle\hat{A}\rangle = \sum_i a_i \langle\psi|P_i|\psi\rangle = \langle\psi| \left(\sum_i a_i P_i\right) |\psi\rangle = \langle\psi|\hat{A}|\psi\rangle.$$

Therefore when we talk about the result of a measurement we mean the expectation value of the state computed over the operator.

The last characterization about observables and expectation values leads to some important considerations about the *phases* of wavefunctions. Usually in physics this term can assume different meanings depending upon the context. Now we focus on what phase is in our case and which is its role in the description of the states. Consider, for example, the state $e^{i\theta}|\psi\rangle$, where $|\psi\rangle$ is a state vector and $\theta$ is a real number. We say that the states $e^{i\theta}|\psi\rangle$ and $|\psi\rangle$ are equal up to a *global phase* $e^{i\theta}$. Indeed, the state $e^{i\theta}|\psi\rangle$ does not describe a state different than $|\psi\rangle$, this is due to the statistics of measurements. If we have a system described by the state $e^{i\theta}|\psi\rangle$ and we want to measure an observable $\hat{A}$ then we should compute

$$\langle\psi|e^{-i\theta}\hat{A}e^{i\theta}|\psi\rangle = e^{-i\theta}e^{i\theta}\langle\psi|\hat{A}|\psi\rangle = \langle\psi|\hat{A}|\psi\rangle.$$

Straightforwardly as we said before this means that the quantum system considered can be described by both $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$. On the other hand the *relative phase* marks a difference between two state. Let us consider as an example the states

$$\frac{|0\rangle+|1\rangle}{\sqrt{2}} \qquad \text{and} \qquad \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

In the first case the amplitude of the state $|1\rangle$ is $1/\sqrt{2}$ and in the second is $-1/\sqrt{2}$. Nevertheless the probability of measuring the state $|1\rangle$ is $1/2$ in both the cases. The problem arises when we measure in a different basis of eigenvectors, i.e. when we want to measure with an operator $\hat{A}$ that has a different eigenbasis. But also the relative phases have to be considered as we spotted above in (4.9), because it is the cause of the interference term in the computation of the probability.

We are interested in the description of composite quantum systems, that is quantum system composed by two, or more, distinct quantum systems which interact each others. Mathematically we describe this larger system in this way:

**Postulate 5.** *The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if* *we have $n$ numbered systems described by the states $\{|\psi_i\rangle\}_{i=1,\dots,n}$, then the joint state of the global system is $|\Psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$, where $\otimes$ is the tensor product.*

Heuristically, we can give the following reason for choosing the tensor product. If we wanted to describe the composite system $AB$ composed by the two different systems $A$ described by $|A\rangle$ and $B$ described by $|B\rangle$, then we would describe the entire system by $|A\rangle|B\rangle$ and by considering the superposition principle we can notice that an easy way for describing it is, indeed, by means of tensor product. Let us see it specifically with the following example: consider two half spin particles, therefore $\mathcal{H}_1, \mathcal{H}_2 = \text{Span}\{|0\rangle, |1\rangle\}$, where $|0\rangle$ is the spin down state and $|1\rangle$ is the spin up one, and the two following systems

$$\text{System A:} \quad |A\rangle = \alpha_1|1\rangle + \alpha_0|0\rangle, \qquad \text{System B:} \quad |B\rangle = \beta_1|1\rangle + \beta_0|0\rangle$$

Now,

$$\begin{aligned}
|A\rangle|B\rangle &= (\alpha_1|1\rangle + \alpha_0|0\rangle)(\beta_1|1\rangle + \beta_0|0\rangle) \\
&= \alpha_1\beta_1|1\rangle|1\rangle + \alpha_1\beta_0|1\rangle|0\rangle + \alpha_0\beta_1|0\rangle|1\rangle + \alpha_0\beta_0|0\rangle|0\rangle \\
&= \sum_{i,j=0}^{1} \alpha_i\beta_j|i\rangle|j\rangle \text{``}=\text{''} \sum_{i,j=0}^{1} \alpha_i\beta_j|i\rangle \otimes |j\rangle \\
&= (\alpha_1|1\rangle + \alpha_0|0\rangle) \otimes (\beta_1|1\rangle + \beta_0|0\rangle) \\
&= |A\rangle \otimes |B\rangle
\end{aligned}$$

Of course this is not a derivation of the postulate, in fact it is true if and only if we are able to substitute "$=$" with $=$, which is exactly the consequence of the postulate. Nevertheless this example can explain us why we chose the framework of a tenor space for describing composite quantum systems.

For understanding deeply the next property of quantum mechanics it is suitable to have a look at the basis of composite systems. This theoretical result helps us to understand how the basis of the single systems are related to the composite one:

**Theorem 4.1.** *Let $\{\mathcal{H}_i\}_{i=1,\dots,n}$ be a family of Hilbert spaces and*

$$\mathcal{H} = \bigotimes_{i=1}^{n} \mathcal{H}_i$$

*be their tensor product. Then every element $|\Psi\rangle \in \mathcal{H}$ can be written as*

$$|\Psi\rangle = \sum_{k=1}^{t} |\psi_1\rangle^k \otimes \cdots \otimes |\psi_n\rangle^k,$$

*where $|\psi_i\rangle^k \in \mathcal{H}_i$ and $t \geq 1$.*

In our case, for describing quantum systems we required states with unitary norm therefore *Theorem* 4.1 holds up to normalization of the final state.

Let us consider again the example above of a composite system obtained by two half-spin particles. They have only two elements in the basis: $\mathcal{H}_1, \mathcal{H}_2 =$ Span$\{|0\rangle, |1\rangle\}$. Therefore according to *Theorem* 4.1 we obtain that the basis of the state $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ is

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}.$$

Thanks to *Postulate* 5 we can introduce now one counter-intuitive phenomenon of quantum mechanics: *entanglement*. For the sake of simplicity let us consider only a simple case. Let $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ be a composite system made of two distinct ones. By *Theorem* 4.1 we have that a general element of $\mathcal{H}$ is

$$|\psi\rangle = \sum_{i,j} c_{ij} |i\rangle |j\rangle$$

By definition a state is said *entangled* if it cannot be written as a simple tensor product of a state $|\phi_1\rangle$ belonging to $\mathcal{H}_1$ and a state $|\phi_2\rangle$ belonging to $\mathcal{H}_2$. In the other hand, if such states exist, the state

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle$$

is called *separable*. By considering the two half-spin particles we see that the state

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)$$

is entangled. Instead,

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|1\rangle)$$

can be separated

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle.$$

The idea of entanglement is very related to information. Indeed, if we consider the state $|\psi_1\rangle$ and $|\psi_2\rangle$ and we measure them on the second system we obtain very different results. Let us define the observable $P_1 = |1\rangle\langle 1|$, which is a projector[3] onto the eigenspace generated by the eigenvector $|1\rangle$. If we measured the second system by means of the operator $(\mathbb{I} \otimes P_1)$, which means that we are applying the measurement only on the second system, in the state $|\psi_2\rangle$, we obtain as outcome the state represented by $|1\rangle$, thus the up spin, and according to *Postulate* 3 the final state is

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle$$

which is $|\psi_2\rangle$ itself. And therefore we have no information about the first system, which is still in a superposition of space and therefore inaccessible without measurements.

But now, if we repeated the same measurement on $|\psi_1\rangle$ we would obtain as outcome or $|0\rangle$ or $|1\rangle$ both with probability $1/2$. And in the case we obtained as outcome $|1\rangle$, for the other case is analogue, the final state would be

$$\frac{(\mathbb{I} \otimes P_1)|\psi_1\rangle}{\sqrt{\langle\psi_1|(\mathbb{I} \otimes P_1)|\psi_1\rangle}} = \frac{\frac{1}{\sqrt{2}}|1\rangle|1\rangle}{\frac{1}{\sqrt{2}}} = |1\rangle|1\rangle$$

Therefore if we knew that the state is entangled, after applying the measurement we would have information about the first system as well.

Finally, we have to analyze the tensor product for the operators as well. Now, we are going to formalize this concept by considering the simple case with a composite system of two distinct ones, but it is straightforward to generalize. Let $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, an operator $A$ acting on $\mathcal{H}_1$ and an operator $B$ acting on $\mathcal{H}_2$. $A$ and $B$ can be considered as observables to be measured or as unitary operator for evolving the system. Then considering the states $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$ we define

$$(A \otimes B)(|\psi_1\rangle \otimes |\psi_2\rangle) := (A|\psi_1\rangle) \otimes (B|\psi_2\rangle).$$

Moreover we stress that this notation for operators is suitable in a tensor framework due to *Theorem* 5. In fact if we have and entangled state $|\psi\rangle \in \mathcal{H}$ and

[3]This notation is consistent. Indeed, if we have a vector $v \in \mathbb{C}^n$ we straightforwardly see that $vv^*$ is an orthogonal projector onto the space spanned by $v$. Therefore if we consider Dirac notation we obtain that $|v\rangle\langle v| = vv^*$ is a projector.

$\mathcal{H} = \text{Span}\{|\phi_i\rangle|\rho_j\rangle\}_{i,j}$, then

$$\begin{aligned}(A \otimes B)|\psi\rangle = (A \otimes B)\sum_{i,j} c_{ij}|\phi_i\rangle|\rho_j\rangle &= \sum_{i,j} c_{i,j}(A \otimes B)|\phi_i\rangle|\rho_j\rangle \\ &= \sum_{i,j} c_{ij}(A|\phi_i\rangle) \otimes (B|\rho_j\rangle),\end{aligned}$$

where $|\phi_i\rangle \in \mathcal{H}_1$ and $|\rho_j\rangle \in \mathcal{H}_2$.

And finally we wonder whether the notions of separability and entangled hold for operators as well as states, and what are the consequences of these property. Since both the space of unitary operators, over $\mathbb{C}$, and the space of Hermitian matrices, over $\mathbb{R}$, are vector spaces, then we are allowed to consider the tensor product of two of them.

We can state the correspondent definitions. Let $M$ be an operator acting on $\mathcal{H}$. $M$ is said to be *non-separable* if it cannot be written as a simple tensor product of an operator $A$ acting on $\mathcal{H}_1$ and an operator $B$ acting on $\mathcal{H}_2$. Conversely, if such operators exist the operator

$$M = A \otimes B$$

is called *separable*.

The concept of non-separability for operators is strictly related to entanglement. We show this connection by means of the two half-spin particles example. Let us define the unitary operator $M$ as follow:

$$M : \begin{cases} |0\rangle|0\rangle & \mapsto & |0\rangle|0\rangle \\ |0\rangle|1\rangle & \mapsto & |0\rangle|1\rangle \\ |1\rangle|0\rangle & \mapsto & |1\rangle|1\rangle \\ |1\rangle|1\rangle & \mapsto & |1\rangle|0\rangle \end{cases} \tag{4.10}$$

This operator is clearly an unitary one. It is even more clear to spot the unitarity when we give a matrix representation. This is possible because we are working in finite dimension. Therefore we can have a matrix description by computing the tensor product as the Kronecker product, i.e. given $V$ and $W$ vector spaces over $\mathbb{C}$, or $\mathbb{R}$

$$\cdot \otimes \cdot : V \times W \longrightarrow V \otimes W$$

$$\left( v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix} = w \right) \mapsto v \otimes w = \begin{pmatrix} \vdots \\ v_i \cdot w_1 \\ \vdots \\ v_i \cdot w_k \\ \vdots \end{pmatrix},$$

for $i = 1, \ldots, n$, $v \in V$ and $w \in W$. Thus, in our case, if we consider $\mathcal{H}_1, \mathcal{H}_2 =$ Span$\{e_1, e_2\}$, where $e_i$ is a canonical basis vector, by computing Kronecker product in (4.10) we can see the action of $M$ as

$$M : \begin{cases} e_1 & \mapsto & e_1 \\ e_2 & \mapsto & e_2 \\ e_3 & \mapsto & e_4 \\ e_4 & \mapsto & e_3 \end{cases}$$

And finally its matrix representation is

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

which is clearly unitary.

Moreover this matrix is non-separable and it can create entanglement. Indeed, if we consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle,$$

which is not entangled. But, now, by applying $M$

$$M|\psi\rangle = M\left[\frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle)\right] = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle),$$

which as we saw above is an entangled state.

## Graph theory

A graph is an ordered pair of two sets $G = (V, E)$ satisfying the relation $E \subseteq V \times V$, i.e. the set that contains every $\{v, w\}$ such that $v, w \in V$. To avoid notation ambiguities we shall require that $V \cap E = \varnothing$. The set $V$ is called the vertex set, or nodes set, and usually we refer at it with $V(G)$. The set $E$ is called the set of edges and usually we refer at it with $E(G)$. The usual way of representing graphically graphs is by means of dot for each vertex and joining two of these dots by a line if the corresponding two vertices form an edge. The position of the nodes and the shape of lines are irrelevant, all the properties of graphs are independent from its general graphical representation. Therefore we say that two graphs $G$ and $H$ are *isomorphic* if they can be represented by the same graph or, in a formal way, if there exist a bijective function $f : V(G) \to V(H)$ such that if $\{u, v\} \in E(G)$ then $\{f(u), f(v)\} \in E(H)$. This function is called a ***graph isomorphism***.

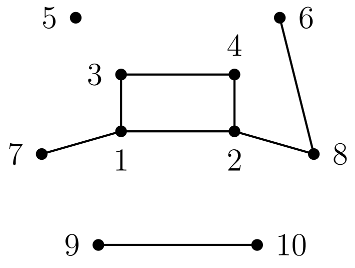


Figure 4.2: Graph with set of nodes $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and with set of edges $E = \{\{1, 2\}, \{1, 3\}, \{1, 7\}, \{2, 4\}, \{2, 8\}, \{3, 4\}, \{6, 8\}, \{9, 10\}\}$.

A vertex $v \in V(G)$ is said to be *incident* to an edge $e \in E(G)$ when $v \in e$ and in this case $e$ is an edge at $v$. The vertices in an edge $e$ are called *ends*. Two nodes that are ends of the same edge are said to be *adjacent*. Adjacency is a equivalence relation and we are going to use the notation $x \sim y$ for saying that $\{x, y\} \in E(G)$. Given a vertex $v \in V(G)$ we are going to call every other node $w \in V(G)$ such that $v \sim w$ a *neighbour* of $v$. We define the *degree* of a vertex $v \in V(G)$ as its number of neighbours and we denote it with $\delta(v)$.

A graph[4] $G$ is said to be *regular* when there exists an integer $n \geq 1$ such that $\delta(v) = n$ for any $v \in V(G)$.

Given a graph $G$, a *subgraph* $H = (V', E')$ is a graph such that $V' \subseteq V(G)$ and

$$E' \subseteq E(G) \cap (V' \times V').$$

[4]From now on we refer to a graph $G = (V, E)$ just with the first capital letter

And given a subset $V' \subseteq V$ we define as *induced subgraph* the subgraph in which the edge set relation is an equality.

Now we gave the basic structure and the basic notion of what a graph is and how to describe it. Let us give a look at different properties and classes of graphs and the first basic substructure.

Given a graph $G$ a *path* is a non empty subgraph $P$ of $G$ such that if $V(P) = \{x_0, \dots, x_k\}$ then

$$1 \leq \sum_{j=1}^{k} |\{x_i\} \cap \{x_i, x_j\}| \leq 2,$$

where $i = 1, \dots, k$ and $\{x_i, x_j\} \in E(P)$.

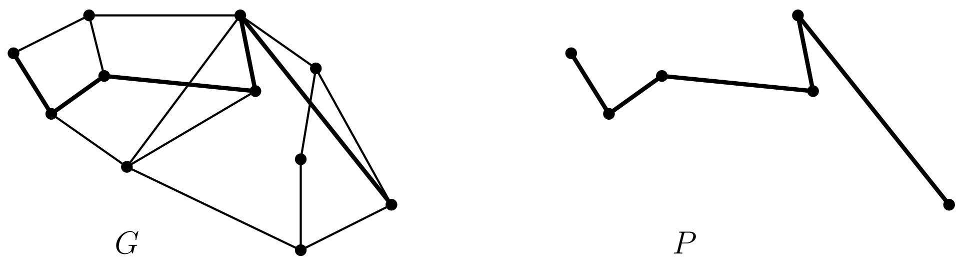


Figure 4.3: Example of a path $P$ as a subgraph of $G$ and as a single graph.

We can think about paths as a sequences of distinct nodes, indeed if $V(P) = \{x_0, \dots, x_k\}$ we could say that $P = x_0 x_1 \cdots x_k$ is the path and where the first and the last vertices are called *endpoints*. In this way we know even the edge set, in fact from the above sequence we can define as the set of edges

$$E(P) = \{\{v, w\} | \exists 0 \leq i < k : vw = x_i x_{i+1} \text{ or } wv = x_i x_{i+1}\}.$$

Moreover it is straightforward to see that by defining paths in this way the requirement (4.2) is satisfied. In this description of paths it is trivial to see that if $\bar{x} \neq x_i$, $i = 1, \dots, k$ then both $P' = \bar{x}P$ and $P'' = P\bar{x}$ are paths.

We define the *length* of a path as the cardinality of the edge set, i.e. $|E(P)|$. Every time we add a vertex to a path $P$ we are building a new path $P'$ which length is increased by $1$. And of course we can define the sum $P + Q$ of two distinct paths $P$ and $Q$, i.e. where all the nodes are distinct, whether at least one endpoint of $P$ is a adjacent to another endpoint of $Q$. The length of the new path $P + Q$ is the sum of the lengths.

We say that a graph $G$ is *connected* if for any couple of vertices $v, w \in V(G)$ there exists a path $P$ which has as endpoints $v$ and $w$. If this property does not hold the graph is said to be non connected.

The property of being connected is an equivalence relation on the vertex set (it is straightforward to see). We define as *connected components* the equivalence classes of this relation. We can notice that if a graph is connected then it has a unique connected component.

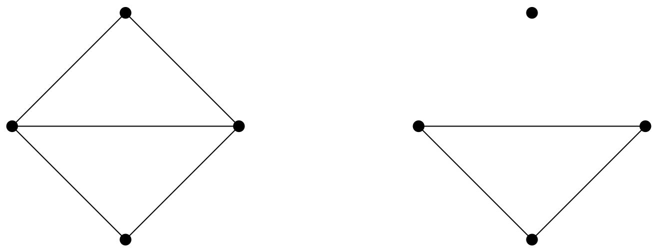


Figure 4.4: An example of a connected graph and of a non connected one

We define now another important structure: *cycles*. If $P = x_0 \cdots x_{k-1}$ is a path with length $k \geq 3$ and $x_0 \sim x_{k-1}$ then we can define the cycle $C := P + x_{k-1}x_0$. Cycles are the only regular graph with degree $2$. A graph which does not have any cycle as a subgraph is called *acyclic*.

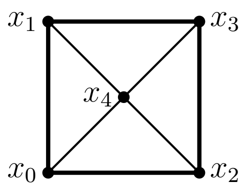


Figure 4.5: A graph where is highlighted a cycle with $4$ nodes; in this case the sequence is $C = x_0x_1x_2x_3$.

Now we have to consider a specific class of graphs that are used when we give an orientation to edges. Thus, we define a *directed graph*, or *digraph*, an ordered couple of sets $D = (V, A)$, where $V(D)$, in this case $V$, is called the vertex set, or node set, and $A(D)$, in this case $A$, is called the arcs set. For avoiding misunderstandings from now on we refer to the graphs defined previously as *undirected graphs*. Differently from the definition of undirected graphs, the arcs set needs to be of the following form

$$A := \{(v, w) | v, w \in V(D)\}.$$

Now we are interested on the edge orientations and this is also suggested by the notation. In fact, in the definition of undirected graph edge the set is an unordered pair of nodes $\{v, w\} = \{w, v\}$ and, contrary, in this case the edge $(v, w)$ stress the fact that we are going from the node $v$ to the node $w$. Thus a vertex $w \in V(D)$ is *adjacent* to another vertex $v \in V(D)$ whether $(v, w) \in A(D)$ and we denote this relation by means of $v \to w$. Like undirected graphs we can represent digraphs by means of balls and links.

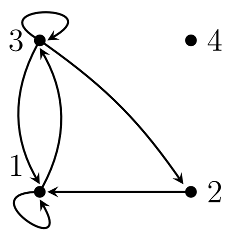


Figure 4.6: Example of digraph $D = (V, A)$, where $V = \{1, 2, 3, 4\}$ and $A = \{(1, 1), (1, 3), (2, 1), (3, 1), (3, 2), (3, 3)\}$.

Moreover we can notice that we have no any constraints that relates vertex set and arcs set and therefore loops are now allowed. An arc is said to be a *loop* whether it has the form $(v, v)$ with $v \in V(D)$.

Now that we have introduced the different basic structure of digraphs, we want to present the concepts we showed for the undirected graph from a digraphs point of view.

Given an arc $(v, w) \in A(D)$, $v$ and $w$ are said to be respectively the *tail* and the *head* of the arc. We redefine the *degree* of a vertex as $\delta(v) = \delta^+(v) + \delta^-(v)$, where $\delta^+(v)$, called the *ingoing degree*, is the number of edges in which $v$ is the head and $\delta^-(v)$, called the *outgoing degree*, is the number of edges in which $v$ is the head. A vertex $v$ that has $\delta^+(v) = 0$ is said to be *source*, on the contrary a vertex $v$ that has $\delta^-(v) = 0$ is called *sink*.

Every digraph can be associated to an undirected graph in this way: given $D$ a digraph, we can define the undirected graph $G = (V, E)$ where $V = V(D)$ and $E$ is obtained from $A(D)$ by leaving out all the loops and by changing every couple $(v, w)$ with $\{v, w\}$. This new undirected graph is called *underlying graph*.

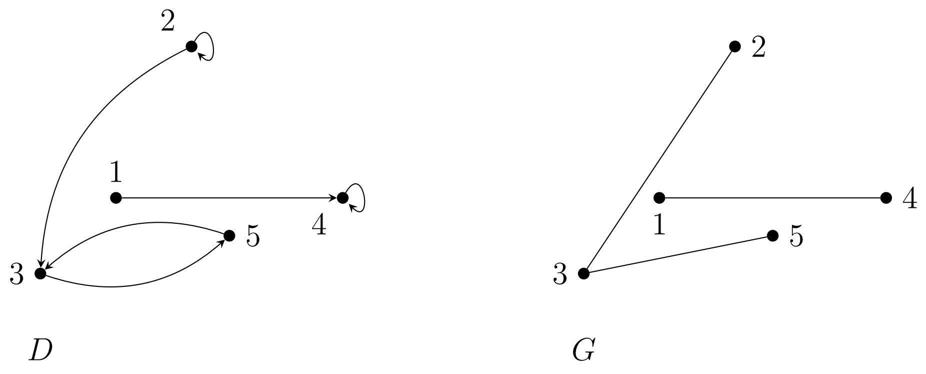


Figure 4.7: A digraph $D = (V, A)$, where $V = \{1, 2, 3, 4, 5\}$ and $A = \{(1, 4), (2, 2), (2, 3), (3, 5), (4, 4), (5, 3)\}$, with its underlying graph $G$ where $V(G) = V$ and $E(G) = \{\{1, 4\}, \{2, 3\}, \{3, 5\}\}$.

We say that $H = (V', A')$ is a subgraph of the digraph $D$ if $V' \subseteq V(D)$, $A' \subseteq A(D)$ and the underlying graph of $H$ is a subgraph of the underlying graph of $D$. Given a subset of vertices $V' \subseteq V(D)$ then we say that $H$ is induced by $V'$ whether contains all the arcs that has as both the tails and the heads in $V'$.

By considering the underlying graph of a digraph we can also study its property as an undirected graph. So for example we would say that a digraph is connected whenever its underlying graph is.

Finally we define the digraph counterpart of paths and cycles, called directed paths and directed cycles. We give directly the sequential notation. A *directed path* is describe by a sequence of distinct nodes $P = x_0 \cdots x_k$ where $V(P) = \{x_0, \ldots, x_k\}$ is the nodes set and $A(P) = \{(x_i, x_{i+1}) | 0 \leq i < k\}$. The length of a directed path is $|A(P)|$. Given a digraph $D$, if for any $v, w \in V(D)$ there exists a directed path $P$ form $v$ to $w$ then $D$ is said to be *strictly con-*

*nected*. It is straightforward to see that for directed paths hold every property of paths.

If $P = x_0 \cdots x_{k-1}$ is a path with length $k \geq 3$ and $x_{k-1} \to x_0$ then $C := P + x_{k-1}x_0$ is a *directed cycle*.

# Bibliography

[1] D. Bacon (2017). “Quantum approach to the unique sink orientation problem", in Physical Review A 96.1 .

[2] G. Benenti, G. Casati and G. Strini (2004). “Principle of Quantum Computation and Information", Singapore, World Scientific.

[3] V. Bosshard (2015). “Classical and quantum algorithms for USO recognition, Master’s thesis", ETH Zürich.

[4] G. Brassard, P. Hoyer, M. Mosca and A. Tapp (2002). “Quantum amplitude amplification and estimation", Contemporary Mathematics, 305, pages 53-74.

[5] M. Conforti, G. Cornuéjols and G. Zambelli (2014). “Integer Programming", Springer International Publishing.

[6] B. Gärtner and I. Schurr (2006). “Linear Programming and Unique Sink Orientations", in SODA ’06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 749-757.

[7] L.K. Grover (1996). “A fast quantum mechanical algorithm for database search", Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing. STOC ’96.

[8] S. Jiang, Z. Song, O. Weinstein and H. Zhang (2020). “Faster Dynamic Matrix Inverse for Faster LPs".

[9] V. Klee and G. Minty (1972). “How good is the simplex algorithm?", in Inequalities III (Proceedings of the Third Symposium on Inequalities held at the University of California).

[10] Y. Lee and A. Sidford (2015). “Efficient inverse maintenance and faster algorithms for linear programming", in FOCS ’15 Foundations of Computer Science.

[11] A.L. Nielsen and I.L. Chuang (2010). “Quantum Computation and Quantum Informtion", New York, Cambridge University Press.

[12] I.A. Schurr (2004). “Unique sink orientations of cubes", Doctoral dissertation, ETH Zurich.

[13] T. Szabó and E. Welzl (2001). “Unique sink orientations of cubes", In Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pages 547-555.

[14] P.M. Vaidya (1989). “Speeding-up linear programming using fast matrix multiplication", in 30th Annual Symposium on Foundations of Computer Science.

[15] D.B. West (2001). “Introduction to Graph Theory", Delhi, Pearson Education.

[16] C. Zalka (1999). “We think that the importance of the algorithm we proposed lies in the exploitation of the heritage property of unique sink orientation hypercube", in Physical Review A60.

# Notation

| | |
|---|---|
| $\oplus$ | Sum in the integer modulo set $\mathbb{Z}_2^n$ or symmetric difference |
| $\otimes$ | Tensor product |
| $\varnothing$ | Empty set |
| $(\cdot\|\cdot)$ | Juxtaposition of two matrices or two vectors |
| $2^A$ | Power set of $A$ |
| $\|\psi\rangle$ | Dirac notation. It represent a state in quantum mechanics |
| $\Theta(f(n))$ | $= \{g(n)\|\exists c_1, c_2 \in \mathbb{R} : c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n), n \to \infty\}$ |
| $\Omega(f(n))$ | $= \{g(n)\|\exists c \in \mathbb{R} : g(n) \geq c \cdot f(n), n \to \infty\}$ |
| $\nabla f(x)$ | Gradient vector of the function $f$ |
| $\|A\|$ | Cardinality of the set $A$ |
| $A(D)$ | Arc set of the digraph $D$ |
| $A^\dagger$ | Conjugate transpose of the matrix $A$ |
| $B \setminus A$ | Set difference |
| $c^T$ | Transpose vector of $c$ |
| $C(A)$ | Vector space spanned by the columns of $A$ |
| $E(G)$ | Edge set of the graph $G$ |
| $\mathbb{I}_n$ | Identity gate over $n$ qubits |
| $i \to j$ | Arch from $i$ to $j$ in a directed graph |
| $Im(f)$ | Image of the map $f$ |
| $ker(f)$ | Kernel of the map $f$ |
| $\lceil m \rceil$ | Ceiling function |
| $[n]$ | ={1,. . . ,n} |