A REFORMULATION OF THE QUANTUM QUERY MODEL

Sebastian Alberto Grillo

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Franklin de Lima Marquezino

Rio de Janeiro
Março de 2017

A REFORMULATION OF THE QUANTUM QUERY MODEL

Sebastian Alberto Grillo

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Franklin de Lima Marquezino, D.Sc.

Prof. Jayme Luiz Szwarcfiter, Ph.D.

Prof. Severino Collier Coutinho, Ph.D.

Prof. Ernesto Fagundes Galvão, Ph.D.

Prof. Renato Portugal, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2017

Grillo, Sebastian Alberto

A reformulation of the quantum query model/Sebastian Alberto Grillo. – Rio de Janeiro: UFRJ/COPPE, 2017.

X, 56 p.: il.; 29, 7cm.

Orientador: Franklin de Lima Marquezino

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 49 – 52.

1. Quantum Computing. 2. Algorithm Analysis. 3. Computational Complexity. I. Marquezino, Franklin de Lima. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicado a mi familia, por su
apoyo a mis metas.

Agradecimentos

Agradeço ao LAC-PESC e ao grupo de computação quântica do LNCC pela ajuda e discussões úteis. Também agradeço ao meu orientador, a COPPE, a CAPES e o Brasil por tornar isto possível.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UMA REFORMULAÇÃO DO MODELO DE CONSULTA QUÂNTICA

Sebastian Alberto Grillo

Março/2017

Orientador: Franklin de Lima Marquezino

Programa: Engenharia de Sistemas e Computação

O modelo de *Quantum Query* (QQM) é uma ferramenta importante para a análise e desenvolvimento de algoritmos quânticos. Este modelo generaliza árvores de decisão, com a complexidade sendo definida pelo número mínimo de consultas a um oráculo a fim de calcular uma função definida para toda cadeia binária de seu domínio. Neste modelo são definidas medidas de complexidade para algoritmos de erro limitado e exatos. Desse modo, usam-se tais medidas para estudar a relação entre algoritmos de erros limitados e algoritmos exatos na computação clássica e quântica. Há várias questões abertas em relação a essas medidas, estimadas principalmente usando métodos de limite inferior.

A falta de abordagens para a construção de algoritmos quânticos é especialmente perceptível para o caso exato. Em contraste, vários resultados ótimos foram encontrados para algoritmos de erro limitado. Alguns resultados importantes são obtidos por reformulações do QQM usando modelos equivalentes, tais como *span programs*. Nesse sentido, o caso exato é menos compreendido do que os algoritmos de erro limitado. Neste trabalho propomos uma nova formulação para o QQM, chamada Formulação de *Block Set* (BSF), dando uma perspectiva alternativa para a análise de algoritmos quânticos exatos. Com esta formulação provamos um novo teorema de limite inferior para algoritmos quânticos exatos.

Usando algumas ideias da BSF, definimos uma simulação de algoritmos quânticos exatos por meios clássicos. A análise desta simulação nos dá uma condição necessária para acelerar a consulta quântica para algoritmos de erro limitado. Esta condição implica altos valores para uma norma-L_1 definida sobre a probabilidade de saída do algoritmo. A partir desta condição, obtemos outras condições necessárias adaptadas dentro da BSF.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A REFORMULATION OF THE QUANTUM QUERY MODEL

Sebastian Alberto Grillo

March/2017

Advisor: Franklin de Lima Marquezino

Department: Systems Engineering and Computer Science

The Quantum Query Model (QQM) is an important tool in the analysis and design of quantum algorithms. This model generalizes decision trees, where the complexity is defined as the minimum number of oracle queries required for calculating a given function f, for any input $x \in \{0,1\}^n$.

This model defines complexity measures for both bounded-error and exact algorithms. Thereby the QQM uses such measures for studying the relation between classical and quantum computing within bounded and exact settings. There are several open questions in relation to such measures, that are mainly estimated using lower bound methods.

A lack of frameworks for constructing quantum algorithms is specially noticeable for the exact case. In contrast, several optimal results were found for bounded error algorithms. Some important results where obtained by reformulations of the QQM using equivalent models, such as span programs. In this sense, the exact case is less understood than bounded-error algorithms. In this work, we propose a new formulation for the QQM called Block Set Formulation (BSF), giving an alternative perspective to the analysis of exact quantum algorithms. From this formulation we prove a new lower bound theorem for exact quantum algorithms.

Using some ideas from BSF, we define a simulation of exact quantum algorithms by classical means. The analysis of this simulation give us a necessary condition for quantum query speed-up of bounded-error algorithms. This condition implies high values for a L_1-norm defined over the output probability of the algorithm. From this condition we obtain other necessary conditions formulated within the BSF.

Contents

List of Figures

Chapter 1

Introduction

From the invention of valve technology through to the development of microelectronics, computing power has been growing continuously. This growth was based on the miniaturization of electronic components, a trend that approximates to the famous predictions of Gordon Moore, who stated that the number of transistors in a dense integrated circuit doubles nearly every two years [1]. However, the current technology is restricted to a scale reaching nanometres and it is close to reaching its limits [2].

In 1982, Richard Feynman introduced the idea of *quantum computing*, as he observed that the simulation of certain physical systems require an exponential number of variables. He argued then that such simulation could be impossible to be executed efficiently by classical computers. Feynman also described a quantum computer and observed that such device could simulate those physical systems efficiently [3]. Thereby, he stated that a quantum computer has the same computability power that a classical computer has, although possibly an improved efficiency. In 1994, Peter Shor formulated a quantum algorithm that factorizes integers with exponential gain over the best classical known algorithm [4]. Another great discovery occurred in 1996, when Grover presented a search algorithm with quadratic gain over the best possible classical algorithm, this algorithm being asymptotically optimal on the number of queries [5]. These two algorithms have very important potential applications and helped increase the interest in quantum computing.

Thus, identifying the complexity gaps between classical and quantum computing became an important theoretical problem. The quantum computational complexity field has two main models: the quantum Turing machine [6] and the quantum query model [7] or black box model. The quantum Turing machine is a generalization of the Turing machine, while the quantum query model is a generalization of the decision tree model [8]. Another analogy is that, if the decision tree model simplifies many aspects about Turing machines, we can say that the quantum query model simplifies the study of quantum algorithms in comparison with quantum Turing

machines. Thus, while classical and quantum Turing machines measure complexity using the number of steps or time, the decision tree and quantum query models measure complexity using the number of queries to a given array as input [8]. For the purposes of this work, decision tree and quantum query frameworks are limited to algorithms that calculate the output of some Boolean function. We can classify the Boolean functions that are calculated. If the domain of some function includes all possible arrays of fixed length we say that it is a total function and otherwise it is a partial function.

The quantum query model allows us to describe exact and bounded error quantum algorithms. An exact algorithm always gives the correct output for any input, while bounded error algorithms give the correct output with at least 2/3 probability for any input [7]. Most known quantum algorithms are bounded error [9], which is an evidence that obtaining a complexity gain over classical algorithms and avoiding error has been a challenge [10]. Some famous examples of exact quantum algorithms are *Deutsch's* algorithm and its generalization, the *Deutch-Jozsa* algorithm [11]. Deutsch's algorithm is easily combined with classical algorithmic steps, in such a way that we can define a family of quantum algorithms obtained by this strategy [12]. Semi-definite programming emerged as a powerful numerical tool for constructing quantum query algorithms, even exact algorithms [13], but the implicit logic behind approximated solutions is difficult to find [12]. In a research paper by Montanaro et al. [12], semi-definite programming is used for discovering some new exact quantum algorithms, where those algorithms cannot be obtained just using Deutsch's algorithm as subroutine. Lately, the algorithms described by Ambainis in [14] introduce new ideas for designing exact quantum algorithms. Finally, we remark an another recent paper by Ambainis [10], which presents a new powerful result, describing the first quantum exact algorithm with a super-linear advantage over exact classical algorithms for total functions. Even with these results, the toolbox for constructing exact quantum algorithms is still limited and developing new design methods is an interesting field [10].

Two important measures in quantum complexity are $Q_2(f)$ and $Q_E(f)$, the minimal number of queries for calculating f using a bounded error and exact quantum algorithms, respectively. Their classical analogues are $R_2(f)$ and $D(f)$, the minimal number of queries required for calculating f using a bounded error and deterministic classical algorithms, respectively. An important theoretical problem is finding tight inequalities relating these measures or other complexity measures like polynomial degree [15], sensitivity [16] or certificate complexity [8].

Recently, some reformulations of the quantum query model were developed [17]. These formulations use the concept of span programs and they had great success as a tool for obtaining efficient bounded error algorithms [18, 19]. In this work, we have

two main contributions. In our first contribution, we develop another formulation of the Quantum Query Model that could facilitate new tools for analysing quantum exact algorithms, analogously to span programs in relation to bounded error algorithms. In our second contribution, we identify necessary conditions for quantum speed-up in the QQM. These conditions are formulated by upper bounds that involves a L_1-norm measure and other measures defined over our new formulation.

In our **first contribution**, we prove that the application of any quantum query algorithm is equivalent to decomposing an unit vector in a sum of real vectors, where the final state of the quantum algorithm is equal to the same sum, although inverting the sign of some of those vectors. Each input determines which signs must be inverted, however such decomposition is the same for every input. We identify a set of properties fulfilled by the vectors of the decomposition. We call this set of properties the *Block Set Property* and any set with such property is a *Block Set*. We define the output state of a given Block Set as an analogy to the output state of a given quantum query algorithm. We prove that for any quantum query algorithm there is a Block Set and for any Block Set there is a quantum query algorithm, where the Gram matrix of their output states are equal. This result implies that the Block Set formulation characterizes the evolution of states in the Quantum Query Model. Thus, the Block Set formulation considers the same measurement step as the quantum query model. We prove that the Block Set formulation acts as a parametrization of the Gram matrix of output states, where each pair of elements in the Block Set controls a matrix and the sum of all those matrices is the Gram matrix. We use such parametrization for obtaining a linear system of equations, where a semi-definite solution is a necessary and sufficient condition for the existence of an exact quantum algorithm, with fixed complexity and calculated function. We study a special case of Block Sets, called *Orthogonal* Block Sets, which depends on a much simpler system of equations. We develop strategies for solving the simpler system of equations that produces Orthogonal Block Sets, including an iterative procedure that constructs the desired function gradually. As an example of the application of this approach, we obtain a family of algorithms that generalizes the Deutsch-Jozsa algorithm. Finally, we prove a theorem that lower-bounds the number of queries that a exact quantum query algorithm needs for computing a Boolean function, this new method offers an alternative for traditional methods. The method applies a basis of orthogonal functions for the Boolean cube.

From our first contribution, some other problems followed. One of those problems is identifying necessary conditions in a Block Set for a quantum speed-up in relation to classical algorithms. This problem leads to our **second contribution**. We start by considering that the probability for a given output is a linear combination of orthogonal functions, and those functions are the same basis applied for the lower-

bound method. We define a classical simulation for a Block Set, which depends on the linear combination that represents the output probability. Then, we analyse how expensive the classical simulation of some Block Set is in relation to a given measure. The measure applied here is the L_1-norm obtained from the basis for the Boolean cube. Particularly, we relate such L_1-norm with the error of the simulated algorithm. Thereby, a large L_1-norm is a necessary property for a hard classical simulation. We obtain an upper bound for the quotient between the queries for an optimal classical algorithm and the queries of the quantum algorithm, which formalizes our necessary condition. Notice that an optimal quantum query algorithm maximizes such quotient. From Block Sets we define measures that upper-bound L_1-norm, and this formulates alternative necessary conditions for quantum speed-up. In the Block Set, each pair of vectors controls a computation in the algorithm, which formalizes the notion of quantum parallelism.

The present document is organized as follows. In Chapter 2, we present an introduction to the QQM, quantum query complexity and brief descriptions of approaches for exact quantum algorithms. In Chapter 3, we present the first contribution of our work, which consists in a reformulation of the QQM. In Chapter 4, we show necessary conditions for quantum speed-up on the BSF, which is the second original contribution of this work. Finally, in Chapter 5, we discuss our results and the next steps of our research.

Chapter 2

The Quantum Query Model revisited

In this chapter we present an introduction of quantum and classical models, basic complexity methods and important related works in the last years. Section A.1 (on page 53) gives a complementary introduction to basic concepts.

2.1 The quantum query model

The Quantum Query Model (QQM), or Black Box Model, is a formulation that simplifies quantum algorithms in relation to other models, in the sense that we just compute a function given an input x, where we are specially interested in the queries on such input [8]. The query operator is the only operation with access to the input or black box, and we count the number of its occurrences. The other operations are represented by unitary operators and they are invisible to our complexity measure. This model is a very important tool for the study of search algorithms, because such algorithms also make queries to a search space, that we formulate as an input in the QQM. But QQM algorithms are also interesting, because we could replace our input or black box by a *white box* or circuit that actually implements the black box; that is the case of Shor's algorithm and this allow us to obtain algorithms for more complex models [7].

We consider a formulation of the QQM that consists of two registers: (i) the query register, with enough size for representing an integer $i \in \{0, .., n\}$; and (ii) the working memory.

The query register and the working memory are called the accessible memory. The accessible space H_A has as basis: $|i, w\rangle$ where $i \in \{0, .., n\}$ and w is an allowed configuration for the working memory. Thus, we define the corresponding spaces for each register (see Subsection A.1.1 on page 53):

- The query space H_Q which has as basis the vectors $\{|i\rangle : 0 \leq i \leq n\}$. This space should not be confused it with a space of possible inputs, that would have dimension 2^n.

- The work space H_W which has as basis the vectors $|w\rangle$, where w belongs to the possible set of values in the working memory.

Thus $H_A = H_Q \otimes H_W$ (see Subsection A.1.3 on page 54). So, if $|\Psi\rangle \in H_A$ then

$$|\Psi\rangle = \sum_{i=0}^{n} |i\rangle |\Psi_i\rangle , \tag{2.1}$$

where $|\Psi_i\rangle \in H_W$. The oracle operator O_x for an input $x \in \{0,1\}^n$ is defined as

$$O_x |i\rangle |\Psi_i\rangle = (-1)^{x_i} |i\rangle |\Psi_i\rangle , \tag{2.2}$$

where $x_0 = 0$. For this reason, x_0 is not considered part of the input and the query space has dimension $n+1$. For our formulation of the QQM, x_0 is very important, otherwise we cannot calculate a wide range of functions. There is an equivalent alternative formulation where x_0 does not exist in x and the query operator is defined as $O_x |i\rangle |b\rangle = |i\rangle |b \oplus i\rangle$, where $b \in \{0,1\}$ [7]. This equivalent formulation implies the same minimal number of queries for any function.

A QQM algorithm with output domain T is determined by

- A number of qubits in the working memory.

- A sequence of unitary operators $\{U_i : 0 \leq i \leq t\}$, that are defined over H_A. The operators are unitary because quantum isolated systems evolve by these type of linear operators (see Subsection A.1.2 on page 53), due to quantum mechanics postulates [20].

- A complete set of projectors (CSOP) (see Section A.2 and Subsection A.1.4 on page 54) over H_A, where the elements of the CSOP are indexed by elements of T. We access the information of the qubits by a measurement, the CSOP represents this final step. Notice that this freedom on the measurement is consequence of our freedom selecting unitary operators.

It is important to mention that the query operator is also an unitary operator, but it is different from set $\{U_i : 0 \leq i \leq t\}$ in the sense that it depends on x. The computation of the QQM algorithm for input x alternates unitary and query operators, thereby it produces a final state

$$\left|\Psi_x^f\right\rangle = U_t O_x U_{t-1} ... U_1 O_x U_0 |0,0\rangle . \tag{2.3}$$

We define the number of queries as the number of times that O_x appears in the computation. Using the CSOP, an output $z \in T$ is obtained with a probability $\pi_x(z) = \left\| P_z \left| \Psi_x^f \right\rangle \right\|^2$.

Definition 1. *A QQM algorithm computes a function $f : \{0,1\}^n \to T$ within error ε if for all input x, there is $\pi_x(f(x)) \geq 1 - \varepsilon$. An algorithm is exact if $\varepsilon = 0$ and it is bounded-error if $\varepsilon \leq \frac{1}{3}$.*

Notice that if we choose a bigger constant error than some $\varepsilon > \frac{1}{2}$ and maintain a fixed number of queries, the set of functions that can be computed under such conditions grow.

Within computational complexity, Boolean functions can also be classified as total or partial. A Boolean function is total if its domain is all the set $\{0,1\}^n$ and it is partial otherwise.

We show an example. Consider H_W an empty set, where the algorithm has initial state $|0\rangle$,

$$U_0 = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix}, \tag{2.4}$$

and $U_1 = I$. Notice that the operator U_0 is a 3x3 matrix, because H_A has dimension 3 as H_Q has dimension $n+1 = 3$ and H_W has dimension 1. Remember that H_Q does not represent the space of all possible inputs, otherwise it would have dimension 4. Thus, we don't include the qubits that contain input, for a similar formulation of quantum query that also includes a input space see Barnum et al.[13].

The measurement step is defined by a CSOP $\{P_i\}$, where

$$P_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

$$P_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and

$$P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Notice that

$$\left\| P_0 \left| \Psi_{00}^f \right\rangle \right\|^2 = \left\| P_0 \left| \Psi_{11}^f \right\rangle \right\|^2 = \left\| P_1 \left| \Psi_{01}^f \right\rangle \right\|^2 = \left\| P_1 \left| \Psi_{10}^f \right\rangle \right\|^2 = 1,$$

therefore the outputs have value 0 if we apply other elements from the CSOP on the same final states. This algorithm solves the Deutsch Problem exactly (see Subsection 2.5.1), in other words it separates sets $\{00, 11\}$ from $\{01, 10\}$ within error 0.

2.2 Decision trees

We present classical query algorithms using decision trees. Similarly to the QQM, these are models for algorithms that compute a function given a Boolean input x [8].

Deterministic decision trees are defined as rooted ordered binary trees, where each internal node has a variable x_i as label and each leaf has 1 or 0 as label. The tree evaluates an input x as follows. It starts at the root and it executes the following recursive procedure:

1. If it is a leaf, then stop and choose its label as output.

2. Otherwise, query the variable x_i that labels the root with that label. If $x_i = 0$ then apply this recursive procedure over the left sub-tree. If $x_i = 1$ then apply this recursive procedure over the right sub-tree.

A deterministic decision tree T computes f, if it outputs $f(x)$ for all $x \in D \subset \{0,1\}^n$, where D is the domain of f. The complexity or number of queries used by T is its depth. Notice that it is deterministic because it always gives the same output for a given input x. Finally, observe that each node of the tree is labelled with just one variable x_i, that models that our classical algorithms can obtain the value of just one variable per query.

We present two examples, first a trivial decision tree that outputs 1 without querying any variable, and second the decision tree of Figure 2.1 which queries variables x_i and x_j.

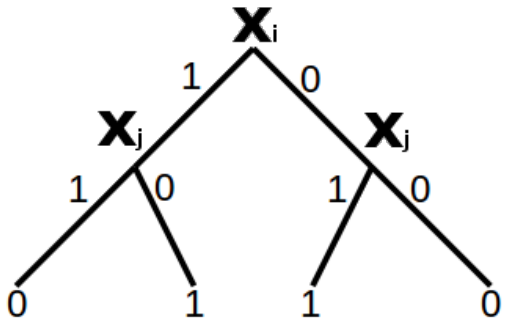

Figure 2.1: Deterministic decision tree that queries variables x_i and x_j. Notice that taking $i = 1$ and $j = 2$, it solves the Deutsch problem (see Subsection 2.5.1 on page 14).

Randomized decision trees are defined as a probability distribution μ over a finite set of deterministic decision trees. The evaluation of some input x starts by choosing a tree according to μ and follows by evaluating it as any other deterministic tree. The complexity is defined as the depth of the deepest deterministic tree T, such that $\mu(T) > 0$. Notice that the randomized model generalizes the deterministic model, because a deterministic tree is just a randomized tree that gives probability one to a single tree. At the same time the QQM generalizes both models, because we can simulate a randomized decision tree using a QQM algorithm with the same error probability and number of queries [8].

We present an example of randomized tree, giving probability $\frac{1}{3}$ to the tree that always outputs 1 and probability $\frac{n(n-1)}{3}$ to each different pairs i, j that determines a decision tree from Figure 2.1. This randomized tree solves the Deutsch-Jozsa problem within error $\frac{1}{3}$ (see Subsection 2.5.1).

2.3 Quantum Complexity

The QQM has an implicit optimization problem, that is calculating a given function $f : \{0,1\}^n \to T$, within error ε and using a minimal number of queries.

Definition 2. *There are the following complexity measures:*

- $D(f)$ *is the minimal number of queries for calculating* f*, by a deterministic decision tree.*
- $R_2(f)$ *is the minimal number of queries for calculating* f*, by a randomized decision tree.*
- $Q_2(f)$ *is the minimal number of queries for calculating* f*, by a quantum bounded-error algorithm.*
- $Q_E(f)$ *is the minimal number of queries for calculating* f*, by a quantum exact algorithm.*
- $Q_\varepsilon(f)$ *is the minimal number of queries for calculating* f*, by a quantum algorithm within error* ε*.*

We have that $n \geq D(f) \geq R_2(f) \geq Q_2(f)$ and $D(f) \geq Q_E(f) \geq Q_2(f)$ for any f [8]. We deduce the upper-bound n, just noticing that a deterministic decision tree can compute any function just checking all the terms x_i.

An interesting theoretical problem in quantum complexity, it is that of finding inequalities that relate these measures between them, or with other measures defined over Boolean functions that we will present in this chapter. The following inequality

is a very important bound in the complexity gain of QQM algorithms over their classical counterparts.

Theorem 1. *Let f be a total function, then $D(f) \leq 2^{12} Q_2(f)^6$*

This theorem [8] implies that if a QQM algorithm has at least bounded error precision, then it could obtain an exponential speed over deterministic classical algorithms only in promise problems, where it is guaranteed that input can take limited configurations. In the particular case of exact quantum algorithms, we can expect a lower potential gain due to bigger restrictions.

Theorem 2. *Let f be a total function, then $Q_E(f) = \Omega\left(D(f)^{1/3}\right)$.*

Theorem 2 says that an exact quantum algorithm cannot exceed a cubic gain over deterministic algorithms for total functions. We ignore if this lower bound is tight, we do not know if there exists an exact quantum algorithm reaching a cubical gain for total functions [10]. Nevertheless, exact quantum algorithms can reach an exponential gain over deterministic classical algorithms, as Deutsch-Jozsa algorithm [11]. But this potential complexity gain cannot be significant or even exist for some functions, for example if f is the function AND_n (see A.4 on page 55) then $Q_E(f) = D(f)$ [21].

2.4 Complexity measures and methods

A basic problem in the field is obtaining the value $D(f)$, $R_2(f)$, $Q_2(f)$ or $Q_E(f)$ for a given function f. The most often used strategy is obtaining a lower bound for such complexity measure and comparing such bound with the most efficient known algorithm. If the complexity of such algorithm is equal to the lower bound, then we know the complexity of f. If there is a gap between the complexity of the algorithm and the lower bound, then there are three possibilities:

- The algorithm can use fewer queries,
- The lower bound is not tight,
- Both possibilities hold.

Thus, the development of tight lower-bound methods is an active research direction. We will present a review of results for the most important lower bound approaches, without much mathematical details.

2.4.1 The Hybrid method

We can formulate a search problem, where our goal is finding a marked element k in $\{1, 2, .., n\}$. Such search problem is more difficult than a decision problem that consists in simply determining if there is some $x_i = 1$, for $x \in \{0,1\}^n$, because a search algorithm can be trivially transformed in a decision algorithm. Thus, a lower bound for OR_n is also a lower bound for our original search problem [7]. The first developed lower bound method in the field is the hybrid method [22], which gives a tight lower bound for OR_n. The basic idea of this method can be described as follows. Suppose that we already have a QQM algorithm for OR_n. This method starts analysing how orthogonal must be the final states $\left|\Psi_{0^n}^f\right\rangle$ and $\left|\Psi_x^f\right\rangle$, where 0^n represents a space without marked elements and x represents a space with one marked element. The orthogonality of those final states is a necessary condition if we need to distinguish one from the other with enough precision (A.3 on page 55), at the measurement step. A QQM algorithm starts with the initial state $|0,0\rangle$ for all inputs, thereby each query modifies the orthogonality of the state of the algorithm for 0^n and y, at each step. So the Hybrid method uses an upper bound for the rate of orthogonalization in final states, which finally implies a lower bound for OR_n or the following theorem by Bennett et al. [22].

Theorem 3. *If a bounded-error QQM algorithm recognizes* 0^n *from the set* $B = \{x \in \{0,1\}^n : |\{i : x_i = 1\}| = 1\}$ *and* $|B| = \Omega(n)$*, then it requires at least* $\Omega(\sqrt{n})$ *queries.*

This complexity result demonstrated that Grover's algorithm is optimal, because Grover's algorithm executes a search in time $O\left(\sqrt{N}\right)$, with high probability [5]. At the same time the existence of Grover's algorithm shows that the hybrid method is tight for this problem.

2.4.2 Polynomial method

The polynomial method for the QQM is based on two properties. First, $\left|\Psi_x^f\right\rangle$ can be decomposed in a linear combination using a basis for H_A, where the amplitude for each element i from the basis is a n-variate polynomial $p_i(x)$, with degree at most $2t$ (t is the number of queries). Second, any Boolean function f can be interpolated by a n-variate polynomial $q(x)$. Thereby, the degree of $q(x)$ cannot exceed the degree of $p_i(x)$, if the algorithm calculates f. So, this is a method based in the polynomial measures that we define formally below [23].

Definition 3. *We say that a n-variate polynomial* $q(x)$ *represents a Boolean function* $f : D \to \{0,1\}$*, if* $f(x) = q(x)$ *for* $x \in D \subset \{0,1\}^n$*. Then the minimum degree for a polynomial that represents* f *is denoted as* $deg(x)$*.*

The polynomial representing total functions is unique [7].

Definition 4. *We say that a n-variate polynomial $q(x)$ approximates a Boolean function $f : D \to \{0,1\}$, if $|f(x) - q(x)| \leq \frac{1}{3}$ for $x \in D \subset \{0,1\}^n$. Then the minimum degree for a polynomial that approximates f is denoted as $\widetilde{deg}(x)$.*

Thus, we have the following theorems proved by Beals et al. [23].

Theorem 4. *Let f be a Boolean function, then $Q_E(f) \geq \frac{deg(x)}{2}$.*

Theorem 5. *Let f be a Boolean function, then $Q_2(f) \geq \frac{\widetilde{deg}(x)}{2}$.*

The polynomial method has proved to be a powerful method, due to diverse applications in quantum complexity like the collision problem [24] or direct product theorems [25, 26].

2.4.3 Adversary methods

This approach was born as an extension of the hybrid method and can also be considered a generalization of the block sensitivity approach [7]. Instead of assuming orthogonality between individual final quantum states in a hypothetical algorithm, this approach assumes the orthogonality between sets of final quantum states, where each set corresponds to a different desired output for an algorithm that calculates f. Depending on f and the tolerated error ε, a value V that measures the orthogonality of the sets is calculated. This method produces an upper-bound R for the rate of orthogonalization of such sets at each step of the algorithm, similarly to the Hybrid method. Finally, V/R gives us a lower-bound for calculating f with error ε in the QQM model [7]. In the first years after its introduction, the original Adversary method [27] was modified in several formulations [13, 28–30], but those first modifications resulted in equivalent methods to the original [31]. Nevertheless, this approach received deeper modifications that increased its power, as the negative weight adversary method [32] and the multiplicative adversary method [33]. It is known that the negative weight adversary method is able to obtain better lower-bounds than the original adversary method [32] and it also offers important reformulations of the QQM [17].

We introduce some notations first. A function $g : S \to T$, where $S \subset \mathbb{Z}_m^n$. A hermitian matrix Γ, with rows and columns indexed by elements of S. If there is $(g(x) = g(y) \Leftrightarrow \Gamma[x,y] = 0)$, we say that Γ is an adversary matrix. We denote $\|M\|$ by the spectral norm of the matrix M. If M is a real matrix, then $M \geq 0$ says that the entries of M are non negative. $M \circ N$ denotes the entry-wise (Hadamard) product, between matrices M and N. Thus, we can introduce the spectral formulation of the adversary methods developed by Hoyer et al. [32].

Definition 5. *The complexity measure for the adversary method is defined by*

$$ADV(f) = \max_{\substack{\Gamma \geq 0 \\ \Gamma \neq 0}} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|},$$

where Γ *is a symmetric adversary matrix, and* $D_i[x,y] = 1$ *if* $x_i \neq y_i$ *and otherwise* $D_i[x,y] = 0$.

Theorem 6. *For any function* g *as above, one has*

$$Q_\varepsilon(g) \geq \frac{1 - 2\sqrt{\varepsilon(1-\varepsilon)}}{2} ADV(g).$$

Definition 6. *The complexity measure for the negative weight adversary method is*

$$ADV^{\pm}(f) = \max_{\Gamma \neq 0} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|}$$

Theorem 7. *For any function* g *as above, one has*

$$Q_\varepsilon(g) \geq \frac{1 - 2\sqrt{\varepsilon(1-\varepsilon)} - 2\varepsilon}{2} ADV^{\pm}(g).$$

If g *is Boolean* ($|T| = 2$)*, then*

$$Q_\varepsilon(g) \geq \frac{1 - 2\sqrt{\varepsilon(1-\varepsilon)}}{2} ADV^{\pm}(g).$$

As $ADV^{\pm}(f) \geq ADV(f)$, it is easy to see that the negative weight adversary method obtains the same or better lower-bounds for Boolean functions, than the original adversary method. Also, we can see that the adversary methods are similar to the polynomial method, in the sense that we can define a complexity measure for g that is associated to a lower bound theorem. Jointly with the polynomial method, the adversary approach is a fundamental tool in quantum complexity.

2.5 Exact quantum algorithms

In this last section we present a review of exact quantum algorithms in the QQM. The techniques are not mentioned in detail, we are just interested in presenting the versatility and application of those ideas. This is enough information for a comparison with future proposals.

2.5.1 Deutsch-Jozsa algorithm and parity trees

Before we introduce the applications of Deutsch-Jozsa algorithm, we present Deutsch's problem. Let $x \in \{0,1\}^n$ be a vector that contains exactly $n/2$ ones (balanced), n ones or 0 ones. We have the problem of determining if x is balanced. It is easy to see that a classical algorithm would need $n/2+1$ queries to x for solving such problem, where each query just determines one variable x_i from x (see 2.2 on page 8). In contrast, the Deutsch-Jozsa algorithm solves it with just one query, which implies an exponential gain [11]. This is not a problem with practical applications, but the Deutsch-Jozsa gave important insights on the power of quantum computing. Thus, it is an important theoretical result. The particular case were $n = 2$ is known as Deutsch's algorithm. While $n > 2$ implies the promise of x presenting determinate configurations, $n = 2$ implies a total function, particularly the operation $x_1 \oplus x_2$ with a single query. For almost two decades, using Deutsch's algorithm as subroutine with classical steps was the basic strategy for designing quantum exact algorithms [12]. The algorithms that are constructed in such way are modelled using a tree like the deterministic decision tree, but with internal nodes that also can be labelled with predicates of the form $x_i \oplus x_j$. Thus, the value of such a predicate decides which sub-tree will be taken. Notice that labelling an internal node with such a predicate is equivalent to calling a subroutine of Deutsch's algorithm. This tree is a particular case of the parity tree model [12].

2.5.2 Semi-definite programming

Let f be a Boolean function, t an integer, E_i a matrix such that $\langle x| E_i |y\rangle = (-1)^{x_i+y_i}$, and F_i a diagonal matrix where $\langle x| F_i |x\rangle = 1 \Leftrightarrow f(x) = i$ and otherwise $\langle x| F_i |x\rangle = 0$. Find the 2^n dimensional real symmetric matrices Γ_i for $i \in \{0,1\}$, and M_i^j for $0 \leq i \leq n$ and $0 \leq j \leq t-1$; satisfying the following conditions

$$\sum_{i=0}^{n} M_i^0 = E_0,$$

$$\sum_{i=0}^{n} M_i^j = \sum_{i=0}^{n} E_i \circ M_i^{j-1}$$

(for $1 \leq j \leq t-1$),

$$\Gamma_0 + \Gamma_1 = \sum_{i=0}^{n} E_i \circ M_i^{t-1},$$

$$F_0 \circ \Gamma_0 \geq (1-\varepsilon) F_0,$$

and

$$F_1 \circ \Gamma_1 \geq (1-\varepsilon) F_1.$$

This semi-definite program is denoted as $SDP(f, t, \varepsilon)$

Theorem 8. *There is a QQM algorithm that calculates f, within an error ε and t queries if and only if $SDP(f, t, \varepsilon)$ has a solution [13].*

This reformulation allows us to apply powerful numerical methods, but it is difficult to infer the logic from such numerical results. Nevertheless, that it is a valid strategy for discovering new quantum algorithms. This approach led to the discovery of new exact quantum algorithms, that cannot be constructed simply using Deutsch's algorithm as subroutine. Specifically, there was obtained a general exact algorithm that distinguishes inputs with Hamming weight $n/2$ from those inputs with Hamming weight in the set $\{0, 1, n-1, n\}$. Semi-definite programming also showed that such algorithms can be quite common [12].

2.5.3 Delayed measurement

QQM algorithms are commonly formulated with a final measurement step, but intermediate measurements do not affect the power of the model. Thus, an algorithm with intermediate measurements can be transformed in one with just a final measurement [20]. Using intermediate measurement steps could facilitate the formulation of recursive exact algorithms. Algorithms with this characteristic are used for obtaining $EXACT_{k,n}$ and $THRESHOLD_{k,n}(x)$ (see A.4 on page 55) with $\max\{k, n-k\}$ and $\max\{k, n-k+1\}$ queries, respectively [14]. This approach was employed for obtaining an algorithm that reaches the optimal $Q_E\left(DJ_n^k\right) = k+1$. Function $DJ_n^k(x)$ generalizes Deutsch-Jozsa problem, (see A.4). This approach also gives good results for $EXACT_{k,l}^n$[34], (see A.4).

2.5.4 P-computing

Taking $p \in [-1, 1]$, we say that a QQM algorithm $\mathcal{A}$ p-computes a function f, if there is a quantum state $|\Psi\rangle$ with the following properties:

1. $\mathcal{A}|\Psi\rangle = |\Psi\rangle$ if $f = 0$.
2. If $f(x) = 1$, then $\mathcal{A}|\Psi\rangle = p|\Psi\rangle + \sqrt{1-p^2}\mathcal{A}|\Phi\rangle$ for some state $|\Phi\rangle$ orthogonal to $|\Psi\rangle$. Vector $|\Phi\rangle$ may depend on x.

Using recursive properties of this framework for function NE^i, it is possible to obtain a QQM algorithm that calculates NE^d with $\mathcal{O}\left(2048^{\frac{d}{8}}\right)$ queries. These recursive

properties allow us to transform an efficient algorithm for a fixed d in a general efficient algorithm for any i. A deterministic decision tree requires 3^d queries for solving NE^d, this implies the first known QQM exact algorithm that has a super-linear advantage over classical, because $Q_E\left(NE^i\right) = \mathcal{O}\left(D\left(NE^i\right)^{0.8675..}\right)$ [10].

Chapter 3

Alternatives to the Quantum Query Model

In the last chapter, we presented the few currently available approaches for design of exact QQM algorithms. The parity trees from Deutsch's algorithm can provide exact QQM algorithms for any function, but recent results proved that this approach has complexity limitations [12]. Semi-definite programming give us the most efficient algorithms for functions of fixed size, but it is difficult to generalize those numerical solutions to infinite sets of functions. Delayed measurement and p-computing approaches obtained important complexity results, but the applicability of those ideas to other families of algorithms is not clear. The toolbox for designing exact QQM algorithms has grown with these results, but there is still work to do in developing frameworks for a wider range of Boolean functions [10].

In this work, we believe that constructing QQM exact algorithms could be a hard theoretical problem itself or just a consequence of using an inappropriate formulation. The second hypothesis becomes stronger, if we consider the results of span programs. This reformulation of the QQM obtained optimal algorithms for an important family of Boolean functions, where those algorithms admits a bounded-error [18].

This chapter is organised as follows. In Section 3.1, we analyse a reformulation of the QQM. In Section 3.2, we apply this reformulation to the construction and analysis of exact quantum algorithms. In Section 3.3, we obtain a lower bound method for exact quantum query complexity.

3.1 Characterizing quantum query with multiple CSOP

We introduce some notations first. A sequence of unitary operators $U_nU_{n-1}...U_0$ will be denoted as $\widetilde{U}_n$. Unless mentioned otherwise, we consider that a CSOP $\{P_k : 0 \leq k \leq n\}$ is composed by projectors P_i acting on the subspace $|i\rangle \otimes H_W$. It is convenient to denote operators $\widetilde{P}_i^j = \widetilde{U}_j^\dagger P_i \widetilde{U}_j$. In the following Lemma 1 we prove that $\left\{\widetilde{P}_k^j : 0 \leq k \leq n\right\}$ is a CSOP for any constant j.

Lemma 1. *If $\{P_z : z \in T\}$ is a CSOP and U is an unitary operator, then $\left\{U^\dagger P_z U : z \in T\right\}$ is a CSOP.*

Proof. First we proof that the sum of operators is equal to the unity operator:

$$\sum_{z\in T} U^\dagger P_z U = U^\dagger(\sum_{z\in T} P_z)U = U^\dagger I_H U = I_H \tag{3.1}$$

and second that they are pairwise orthogonal:

$$\left\langle \Phi|U^\dagger P_y^\dagger U U^\dagger P_z U|\Psi\right\rangle = \left\langle \Phi|U^\dagger P_y^\dagger P_z U|\Psi\right\rangle = 0. \tag{3.2}$$

□

We represent an algorithm without measurement step by a 7-tuple

$$\mathcal{A} = (t, n, m, H_Q, H_W, \Psi, \{U_i\}),$$

such that $\dim(H_Q) = n+1$, $\dim(H_W) = m$, $|\Psi\rangle \in H_A$ is an unit vector and the unitary operators in $\{U_i : 0 \leq i \leq t+1\}$ are defined on $H_Q \otimes H_W$. These elements represent all the information that we require for describing an algorithm using $t+1$ queries and an initial state $|\Psi\rangle$, ignoring the measurement step. Without loss of generality, we could always take $|0,0\rangle$ as the initial state, and change U_0 accordingly. However, we will keep an arbitrary Ψ in the above notation, since it will be convenient in the following developments.

3.1.1 Block set formulation

Definition 7. *Let us denote $a = (a_0, a_1, .., a_t)$ for $0 \leq a_i \leq n$ and $\mathbb{Z}_{n+1} = \{0, 1, .., n\}$. We say that an indexed set of vectors*

$$\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$$

is associated with $\mathcal{A} = (t, n, m, H_Q, \ H_W, \Psi, \{U_i\})$ *if*

$$|\Psi(a)\rangle = \widetilde{P}^t_{a_t}..\widetilde{P}^1_{a_1}\widetilde{P}^0_{a_0}|\Psi\rangle . \tag{3.3}$$

Intuitively, each unitary operator rotates a vector state and each query cuts pieces of such vector, for a posterior sign change of those pieces. Each projector $P^i_{a_i}$ represents one of such cuts and multiples cuts produce more of such pieces. The vector $|\Psi(a)\rangle$ represents such independent pieces after all of such cuts. Using the operators defined above, we have a valid expression for any vector, given by

$$|\Psi\rangle = \left(\sum_{k_t=0}^{n} \widetilde{P}^t_{k_t}\right) ... \left(\sum_{k_0=0}^{n} \widetilde{P}^0_{k_0}\right)|\Psi\rangle \tag{3.4a}$$

$$= \sum_{k_t=0}^{n} ... \sum_{k_0=0}^{n} |\Psi(k_0, ..., k_t)\rangle . \tag{3.4b}$$

Observe that, if $|\Psi\rangle$ is an unit vector, then $\{|\Psi(k_0, ..., k_t)\rangle\}$ is associated with some algorithm $\mathcal{A}$ with $t+1$ queries and initial state $|\Psi\rangle$. Thus, Eq. (3.4) can be a decomposition of a given initial state. Considering that $\left\{\widetilde{P}^j_i : 0 \leq i \leq n\right\}$ is a CSOP for each fixed j and

$$O_x|\Psi\rangle = \sum_{i\in\{k:x_k=0\}} P_i|\Psi\rangle - \sum_{i\in\{k:x_k=1\}} P_i|\Psi\rangle , \tag{3.5}$$

then we have that

$$\widetilde{U}^\dagger_j O_x \widetilde{U}_j|\Psi\rangle = \sum_{i\in\{k:x_k=0\}} \widetilde{P}^j_i|\Psi\rangle - \sum_{i\in\{k:x_k=1\}} \widetilde{P}^j_i|\Psi\rangle . \tag{3.6}$$

Eq. (3.6) shows that the query operator between the unitary operators has a simple behaviour, determined by the operators that define our decomposition. We will see that this property allows us to represent with simplicity any QQM algorithm for a given input x, using our decomposition.

Theorem 9. *If an indexed set* $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}^{t+1}_{n+1}\right\}$ *is associated with an algorithm* $\mathcal{A} = \big(t, n, m, \ H_Q, H_W, \Psi, \{U_i\}\big)$*, then*

$$\widetilde{U}^\dagger_t O_x U_t ... U_1 O_x U_0|\Psi\rangle = \sum_{k_t=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t} x_{k_i}} |\Psi(k_0, ..., k_t)\rangle . \tag{3.7}$$

Proof. We prove the theorem by induction. First, we check that Eq. 3.7 is true when $t = 0$: Evaluating for $t = 0$: $\widetilde{U}^\dagger_0 O_x U_0|\Psi\rangle = U^\dagger_0 O_x U_0|\Psi\rangle = \sum_{k_0=0}^{n}(-1)^{x_{k_0}}|\Psi(k_0)\rangle$.

Now, let $t' > 0$ and assume that Eq. 3.7 is also true for some $t = t'$. Then, we have

$$\widetilde{U}_{t'}^{\dagger} O_x ... O_x U_0 \left|\Psi\right\rangle = \sum_{k_{t'}=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t'} x_{k_i}} \left|\Psi\left(k_0, ..., k_{t'}\right)\right\rangle .$$

If we apply Eq. (3.6), then we find that

$$\widetilde{U}_{t'+1}^{\dagger} O_x ... O_x U_0 \left|\Psi\right\rangle$$

$$= \sum_{k_{t'}=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t'} x_{k_i}} \left(\sum_{k_{t'+1}=0}^{n} (-1)^{\delta_{k_{t'+1}}(S_x)} \widetilde{P}_{k_{t'+1}}^{t'+1} \left|\Psi\left(k_0, ..., k_{t'}\right)\right\rangle \right) .$$

A reordering of the summations gives

$$\widetilde{U}_{t'+1}^{\dagger} O_x ... O_x U_0 \left|\Psi\right\rangle = \sum_{k_{t'+1}=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t'+1} x_{k_i}} \widetilde{P}_{k_{t'+1}}^{t'+1} \left|\Psi\left(k_0, ..., k_{t'}\right)\right\rangle .$$

Using Definition 7, we have

$$\widetilde{U}_{t'+1}^{\dagger} O_x ... O_x U_0 \left|\Psi\right\rangle = \sum_{k_{t'+1}=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t'+1} x_{k_i}} \left|\Psi\left(k_0, ..., k_{t'+1}\right)\right\rangle ,$$

which means that Eq. 3.7 is also true for $t = t' + 1$. Thus, Eq. 3.7 is true for all $t \geq 0$. □

Corollary 1. *Let us denote* $\left|\bar{\Psi}\left(k_0, ..., k_t\right)\right\rangle = U_{t+1}\widetilde{U}_t \left|\Psi\left(k_0, ..., k_t\right)\right\rangle \ \forall k_i \in \mathbb{Z}_{n+1}$. *If* $\left\{\left|\Psi\left(k\right)\right\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ *is associated with* $\mathcal{A} = (t, n, m, H_Q, H_W, \Psi, \{U_i\})$ *then*

$$U_{t+1} O_x U_t ... U_1 O_x U_0 \left|\Psi\right\rangle = \sum_{k_t=0}^{n} ... \sum_{k_0=0}^{n} (-1)^{\sum_{i=0}^{t} x_{k_i}} \left|\bar{\Psi}\left(k_0, ..., k_t\right)\right\rangle . \tag{3.8}$$

Corollary 1 implies that any QQM algorithm can be characterized by a sum of invariant vectors, whose signs depend on the input.

Figure 3.1 shows an example of a generic QQM algorithm with two queries. Observe that algorithm $\mathcal{A}$ mentioned in Corollary 1 is equivalent to the algorithm $\mathcal{A}$ mentioned in Theorem 9, if we ignore the measurement step. This is due to the fact that both algorithms have the same Gram matrices for the final states $\left|\Psi_x^f\right\rangle$. Thereby, the last unitary operator U_{t+1} can be ignored and we conclude that an algorithm without measurement is fully described by the vectors of the decomposition, see Eq. (3.4).

We can ask which properties allow a set of vectors to be associated to some

arbitrary QQM algorithm. The answer of such a question would offer an alternative description of QQM algorithms. The next definition and theorems advance in that direction.

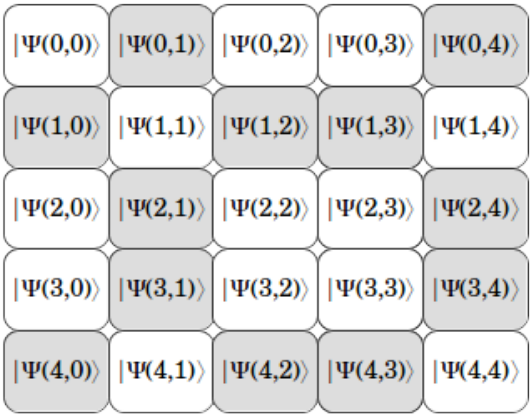

Figure 3.1: Schematic representation of $\widetilde{P}_x^1 \widetilde{P}_x^0 |\Psi\rangle$ using Eq.(3.7), where $x = 1001$. Grey boxes represent vectors with inverted sign respect to the decomposition of $|\Psi\rangle$.

Definition 8 (Block Set)**.** *Let $n, t \geq 0$. An indexed set*

$$\left\{|\Psi(k)\rangle \in H_1 \otimes H_2 : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$$

is a Block Set *for an ordered pair of Hilbert spaces (H_1, H_2), if:*

1. *The vectors defined as $|\Psi^i(a_0, .., a_{t-i})\rangle = \sum_{k_1=0}^{n} ... \sum_{k_i=0}^{n} |\Psi(a_0, .., a_{t-i}, k_1, .., k_i)\rangle$ satisfy $\langle \Psi^i(b_0, .., b_{t-i}) \mid \Psi^i(c_0, .., c_{t-i})\rangle = 0$ if $b_{t-i} \neq c_{t-i}$ for $0 \leq i \leq t$.*
2. *There is unitarity for the sum of vectors $\sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \| |\Psi(k_0, .., k_t)\rangle \|^2 = 1$.*
3. *The spaces $H(i, j)$ satisfy $\dim(H(i, j)) \leq \dim(H_2)$ $\forall i, j$, where $H(i, j)$ represents the space generated by $C_j^i = \{|\Psi^{t-i}(a_0, .., a_{i-1}, j)\rangle : a_k \in \mathbb{Z}_{n+1}\}$.*
4. *The space H_1 satisfies $n = \dim(H_1) - 1$.*

Theorem 10. *Let $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ be an indexed set of vectors, that is* associated *with $\mathcal{A} = (t, n, m, H_Q, H_W, \Psi, \{U_i\})$, then such vectors are a Block Set for (H_Q, H_W).*

Proof. The proof is divided into four parts, each corresponding to one of the Block Set properties from Definition 8 (on page 21).

1. Considering that $\left\{\widetilde{P}_k^{t-i} : 0 \leq k \leq n\right\}$ is a CSOP and $|\Psi^i(a_0, a_1, .., a_{t-i})\rangle = \widetilde{P}_{a_{t-i}}^{t-i} .. \widetilde{P}_{a_1}^1 \widetilde{P}_{a_0}^0 |\Psi\rangle$ then $\langle \Psi^i(b_1, .., b_{t-i}) \mid \Psi^i(c_1, .., c_{t-i})\rangle = 0$ for all $b_{t-i} \neq c_{t-i}$.

2. Using induction:

   - For $t=0$, $\sum_{k_0=0}^{n} \left\| \left|\Psi\left(k_0\right)\right\rangle \right\|^2 = \sum_{k_0=0}^{n} \left\| \widetilde{P}_{k_0}^{0} \left|\Psi\right\rangle \right\|^2 = 1$.
   - If $\sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \left\| \left|\Psi\left(k_0,..,k_t\right)\right\rangle \right\|^2 = 1$

     $\Rightarrow \sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \left\| \widetilde{P}_{k_t}^{t} .. \widetilde{P}_{k_1}^{1} \widetilde{P}_{k_0}^{0} \left|\Psi\right\rangle \right\|^2 = 1$

     $\Rightarrow \sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \left(\sum_{k_{t+1}=0}^{n} \left\| \widetilde{P}_{k_{t+1}}^{t+1} \widetilde{P}_{k_t}^{t} .. \widetilde{P}_{k_1}^{1} \widetilde{P}_{k_0}^{0} \left|\Psi\right\rangle \right\|^2 \right) = 1$

     $\Rightarrow \sum_{k_0=0}^{n} ... \sum_{k_{t+1}=0}^{n} \left\| \left|\Psi\left(k_0,..,k_{t+1}\right)\right\rangle \right\|^2 = 1.$

3. Consider that the spaces generated by $\left\{ \left|\Psi^i\left(a_0,..,a_{t-i-1},j\right)\right\rangle : a_k \in \mathbb{Z}_{n+1} \right\}$ and $\left\{ \widetilde{P}_j^{t-i} \widetilde{P}_{a_{t-i-1}}^{t-i-1} ... \widetilde{P}_{a_0}^{0} \left|\Psi\right\rangle : a_k \in \mathbb{Z}_{n+1} \right\}$ are the same space. That is a subspace of the space generated by the set of vectors $\left\{ \widetilde{U}_{t-i}^{\dagger} \left|j\right\rangle \left|w\right\rangle : w \in H_W \right\}$, this larger space has dimension $\dim\left(H_W\right)$.

4. $\dim\left(H_Q\right) = n+1$.

□

Lemma 2. *If $\left\{P_z^1 : z \in T\right\}$ and $\left\{P_z^2 : z \in T\right\}$ are two CSOP on H, where each H_z^i is the range of P_z^i and $\dim\left(H_z^i\right)$ is constant $\forall i, z$. Then, there is a unitary matrix U, such that $U^\dagger P_z^1 U = P_z^2 \ \forall z \in T$.*

Proof. Suppose that B_z^i is an orthonormal basis of H_z^i, U_1 is a unitary matrix whose rows are the elements of $\bigcup_z B_z^1$ and U_2 is an unitary matrix whose rows are the elements of $\bigcup_z B_z^2$. There exists U_1 and U_2, where for each i the i-th row belongs to $H_{z_i}^1$ and $H_{z_i}^2$, then $\left(U_1^\dagger U_2\right)^\dagger P_z^1 \left(U_1^\dagger U_2\right) = P_z^2$. □

Theorem 11. *If $\left\{ \left|\Psi\left(k\right)\right\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1} \right\}$ is a Block Set for $\left(H_Q, H_W\right)$, then it is associated with some algorithm $\mathcal{A} = \left(t, n, m, H_Q, H_W, \Psi, \left\{U_i\right\}\right)$.*

Proof. We must describe the elements of $\mathcal{A}$ from the Block Set. Values t and n are easily obtained from $\left\{ \left|\Psi\left(k\right)\right\rangle \in H_Q \otimes H_W : k_i \in \mathbb{Z}_{n+1} \right\}$ and m is obtained from H_W, but we need to construct the other elements:

The initial state can be $\left|\Psi\right\rangle = \sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \left|\Psi\left(k_0,..,k_t\right)\right\rangle$. The vector $\left|\Psi\right\rangle$ must be an unit vector. Using $\left|\Psi^i\left(a_0,..,a_{t-i}\right)\right\rangle = \sum_{j=0}^{n} \left|\Psi^{i-1}\left(a_0,..,a_{t-i},j\right)\right\rangle$ and the first property of the Block Set definition, we have that $\left\|\Psi^i\left(a_0,..,a_{t-i}\right)\right\|^2 = \sum_{j=0}^{n} \left\|\Psi^{i-1}\left(a_0,..,a_{t-i},j\right)\right\|^2$. If we apply last expression recursively in $\left|\Psi\right\rangle$ and

use the second property of the Block Set definition, we get: $\|\Psi\|^2 = \sum_{k_0=0}^{n} ... \sum_{k_t=0}^{n} \| |\Psi(k_0,..,k_t)\rangle \|^2 = 1$.

In this part of the proof we constrain the unitary operators for $\mathcal{A}$, we obtain such operators by a previous construction of the CSOP sequence that fulfils Eq (3.3) (on page 19) with the given Block Set. Take $H_1^i = \bigoplus_j H(i,j)$ and a second space H_2^i orthogonal to the first, such that $H_A = H_1^i \oplus H_2^i$. Considering that the third property of Definition 8 (on page 21) implies $\dim(H(i,j)) \leq \dim(H_W)$; if $B(i)$ is an orthogonal basis for H_2^i, then we can take $(\dim(H_W) - \dim(H(i,j)))$ linearly independent elements from $B(i)$ for each pair i,j and denote such set as B_j^i. The space generated by B_j^i is represented as $\widehat{H}(i,j)$. We define $\widetilde{H}(i,j) = \widehat{H}(i,j) \oplus H(i,j)$, which has the same dimension as H_W. For each i, we can impose the condition of $\left\{B_j^i : 0 \leq j \leq n\right\}$ being a disjoint collection of sets, due to $0 \leq j \leq n$ and $\dim(H_A) = (n+1)\dim(H_W)$. Thus, if $j_1 \neq j_2$, then $\widetilde{H}(i,j_1)$ and $\widetilde{H}(i,j_2)$ are orthogonal spaces. Thereby, there is a CSOP $\left\{\widetilde{P}_j^i : 0 \leq j \leq n\right\}$ for each i; such that the range of the projector $\widetilde{P}_j^i$ is $\widetilde{H}(i,j)$. Also, there is an unitary operator $\widetilde{U}_i$ such that $\widetilde{U}_i^\dagger P_j \widetilde{U}_i = \widetilde{P}_j^i$ by Lemma 2 (on page 22) and the definition of the CSOP $\{P_k : 0 \leq k \leq n\}$. The unitary operators are obtained from $U_0 = \widetilde{U}_0$ and $U_i = \widetilde{U}_i \widetilde{U}_{i-1}^\dagger$ for $i > 0$. □

Thus, we proved that for any QQM algorithm without measurement there is a Block Set, and for any Block Set there is a QQM algorithm without measurement. The Block Set model is almost complete, except for one detail: any algorithm is associated with one Block Set; but we can find a Block Set that is associated with more than one QQM algorithm, because the unitary operator after last query has no influence by Eq. (3.7) (on page 19). The next theorem implies that there is no problem if we do not have a bijective relation between models, because algorithms with the same Gram matrix for final states can output the same results depending on the measurement step.

Theorem 12. *Consider two different algorithms that are associated to the same Block Set* $\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\}$*, then the final states of such algorithms form the same Gram matrix.*

Proof. A set of vectors $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ associated to an algorithm is affected just by the unitary operators that appear before the last query. If two algorithms are associated to such a set; by Corollary 1 (on page 20), the final state of the algorithms is equal to the same linear combination of vectors from the Block Set for a given x, but they can be different in the final unitary operator applied over each sum. As consequence, $\left\langle \Psi_x^f \right| \left. \Psi_y^f \right\rangle$ is invariant in both algorithms. □

Definition 9. *Let* $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ *be a Block Set, then its output state for input* x *is defined as:*

$$\left|\Psi_x^f\right\rangle = \sum_{k_t=0}^{n} ... \sum_{k_0=0}^{n} -1^{\left(\sum_{i=0}^{t} x_{k_i}\right)} |\Psi(k_0, ..., k_t)\rangle . \tag{3.9}$$

Notice that Definition 9 implies that an associated Block Set and QQM algorithm produce equivalent outputs, the relation between both models is represented by Figure 3.2. The equivalence of outputs is represented by equal Gram matrices, which implies that we can compute the same function within same margin of error from two associated QQM and BSF algorithms. This formulation maintains the space H_A and we can consider that the measurement step is identical to the original model. Notice that several QQM algorithms can be equivalent to a single BSF algorithm, this implies less redundancy in the BSF.

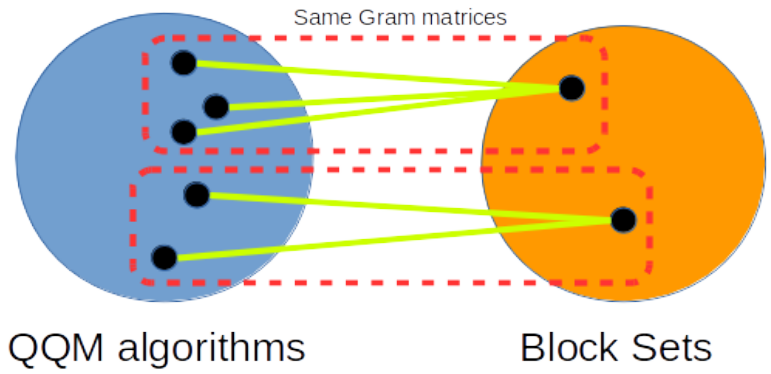

Figure 3.2: The figure shows that the relation between both models is not bijective. The red line means that all BSF or QQM algorithm inside it have the same Gram matrix.

We present a simple example of Block Set associated to a single query QQM algorithm, in particular a quantum algorithm for Deutsch's problem. Recall that we have defined Deutsch's problem in subsection 2.5.1 (on page 14). We consider the same example seen on page 7. Thus we take H_W as an empty set, a initial state $|0\rangle$,

$$U_0 = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix}, \tag{3.10}$$

and $U_1 = I$. The final measurement CSOP is not important for our purposes. Taking the CSOP $\{P_k\}$ that was defined in Section 3, there is $P_i = |i\rangle\langle i|$. From Definition 7 on 18, we have the following elements of the Block Set:

$$|\Psi(0)\rangle = U_0^\dagger P_0 U_0 |0\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.11}$$

$$|\Psi(1)\rangle = U_0^\dagger P_1 U_0 |0\rangle = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}, \tag{3.12}$$

and

$$|\Psi(2)\rangle = U_0^\dagger P_2 U_0 |0\rangle = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{bmatrix}. \tag{3.13}$$

We denote by $\left\{\left|\Psi_x^f\right\rangle\right\}$ and $\left\{\left|\widetilde{\Psi}_x^f\right\rangle\right\}$ the final states of Deutsch's algorithm and the Block Set, respectively. We take $x = x_n..x_2x_1$. From Definition 9 (on page 23) or Theorem 9 (on page 19), we have

$$\left|\widetilde{\Psi}_{00}^f\right\rangle = |\Psi(0)\rangle + |\Psi(1)\rangle + |\Psi(2)\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{3.14}$$

$$\left|\widetilde{\Psi}_{01}^f\right\rangle = |\Psi(0)\rangle - |\Psi(1)\rangle + |\Psi(2)\rangle = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \tag{3.15}$$

$$\left|\widetilde{\Psi}_{10}^f\right\rangle = |\Psi(0)\rangle + |\Psi(1)\rangle - |\Psi(2)\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.16}$$

and

$$\left|\widetilde{\Psi}_{11}^f\right\rangle = |\Psi(0)\rangle - |\Psi(1)\rangle - |\Psi(2)\rangle = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}. \tag{3.17}$$

As

$$\left|\Psi_{00}^f\right\rangle = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = -\left|\Psi_{11}^f\right\rangle \tag{3.18}$$

and

$$\left|\Psi_{01}^{f}\right\rangle = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = -\left|\Psi_{10}^{f}\right\rangle, \tag{3.19}$$

then $\left\{\left|\Psi_{x}^{f}\right\rangle\right\}$ and $\left\{\left|\widetilde{\Psi}_{x}^{f}\right\rangle\right\}$ have the same Gram matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}. \tag{3.20}$$

We can see that for both algorithms the states corresponding to inputs $X = \{00, 11\}$ are orthogonal to the states corresponding to inputs $Y = \{01, 10\}$. Then for both algorithms, there are CSOPs that discriminate X from Y within error 0. That shows that such QQM and BSF algorithms are equivalent.

3.1.2 Gram matrices and Block Sets

In this section we start using Block Sets as an equivalent parametrization of QQM algorithms and we consider the elements in a Block Set as such parameters. We are interested in knowing how each element affects the final Gram Matrix of output states. Thereby, our objective is obtaining a Gram Matrix that allows calculating a given function.

Suppose that inputs x and y give different outputs for a given function, then the QQM algorithm must induce $\left\langle\Psi_{x}^{f}|\Psi_{y}^{f}\right\rangle$ to approximate to zero. We define $|A\rangle$, $|B\rangle$, $|C\rangle$, $|D\rangle \in H_A$, where $|A\rangle$ is the sum of components $|\Psi(a)\rangle$ whose sign is positive in $\left|\Psi_{x}^{f}\right\rangle$ and $\left|\Psi_{y}^{f}\right\rangle$, $|B\rangle$ is the sum of negative components in $\left|\Psi_{y}^{f}\right\rangle$ but positive in $\left|\Psi_{x}^{f}\right\rangle$, $|C\rangle$ is the sum of negative components in $\left|\Psi_{x}^{f}\right\rangle$ and $\left|\Psi_{y}^{f}\right\rangle$, and $|D\rangle$ is the sum of negative components in $\left|\Psi_{x}^{f}\right\rangle$ but positive in $\left|\Psi_{y}^{f}\right\rangle$. Then $|\Psi\rangle = |A\rangle + |B\rangle + |C\rangle + |D\rangle$, $\left|\Psi_{x}^{f}\right\rangle = |A\rangle + |B\rangle - |C\rangle - |D\rangle$ and $\left|\Psi_{y}^{f}\right\rangle = |A\rangle - |B\rangle - |C\rangle + |D\rangle$.

Expanding $\left\langle\Psi_{x}^{f}|\Psi_{y}^{f}\right\rangle$ and $\langle\Psi|\Psi\rangle$, we have, respectively:

$$\begin{aligned} \left\langle\Psi_{x}^{f}|\Psi_{y}^{f}\right\rangle = \; & \langle A|A\rangle - \langle A|B\rangle - \langle A|C\rangle + \langle A|D\rangle \,.. \\ & + \langle B|A\rangle - \langle B|B\rangle - \langle B|C\rangle + \langle B|D\rangle \,.. \\ & - \langle C|A\rangle + \langle C|B\rangle + \langle C|C\rangle - \langle C|D\rangle \,.. \\ & - \langle D|A\rangle + \langle D|B\rangle + \langle D|C\rangle - \langle D|D\rangle \end{aligned} \tag{3.21}$$

and

$$\langle\Psi|\Psi\rangle = \begin{array}{c} \langle A|A\rangle + \langle A|B\rangle + \langle A|C\rangle + \langle A|D\rangle .. \\ + \langle B|A\rangle + \langle B|B\rangle + \langle B|C\rangle + \langle B|D\rangle .. \\ + \langle C|A\rangle + \langle C|B\rangle + \langle C|C\rangle + \langle C|D\rangle .. \\ + \langle D|A\rangle + \langle D|B\rangle + \langle D|C\rangle + \langle D|D\rangle \end{array} = 1. \tag{3.22}$$

Summing equations (3.21) and (3.22), one obtains:

$$\left\langle \Psi_x^f | \Psi_y^f \right\rangle = 2\left(\left\langle \Psi_x^+ | \Psi_y^+ \right\rangle + \left\langle \Psi_x^- | \Psi_y^- \right\rangle\right) - 1. \tag{3.23}$$

Where $|\Psi_x^+\rangle = \frac{\left|\Psi_x^f\right\rangle + |\Psi\rangle}{2}$ and $|\Psi_x^-\rangle = \frac{-\left|\Psi_x^f\right\rangle + |\Psi\rangle}{2}$, analogously for y.

Lemma 3. *Let $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ be a complex Block Set for (H_Q, H_W), where its output states and some CSOP gives a function $f : \{0,1\}^n \to \{0,1\}$ within error ϵ. Then, there exists a Block Set $\left\{\left|\widehat{\Psi}(k)\right\rangle \in \widehat{H}_Q \otimes \widehat{H}_W : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ whose elements have real terms for some $\left(\widehat{H}_Q, \widehat{H}_W\right)$, where its output states and a CSOP calculate f within the same error.*

Proof. If we can use the outputs from $\{|\Psi(k)\rangle\}$ for computing f within error ϵ, then Theorem 11 (on page 22) and Theorem 9 (on page 19) imply that there is a quantum query algorithm that computes f within error ϵ and $t+1$ queries. Also, there is a quantum algorithm that computes f within error ϵ in $t+1$ queries, if and only if a semi-definite program $P(f, t+1, \epsilon)$ has solution [13]. Using the solution of $P(f, t+1, \epsilon)$ we can obtain a QQM algorithm, where unitary matrices and states can be taken as real [12]. This QQM algorithm calculates f, within error ϵ and $t+1$ queries. Thus, using Corollary 1 (on page 20), there is a Block Set that has the same output states of this real QQM algorithm and all its elements are real vectors. □

Using this lemma we can restrict Block Sets to be real, without loss of generality. The following lemma shows a simplification that comes from this real case.

Lemma 4. *If $\left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ is a real Block Set. Then $\langle \Psi_x^+ \mid \Psi_x^- \rangle = 0$, for any $x \in \{0,1\}^n$.*

Proof. We have $\langle \Psi_x^+ \mid \Psi_x^- \rangle = \frac{1}{4}\left(\||\Psi\rangle\|^2 + \left\langle \Psi_x^f \mid \Psi \right\rangle - \left\langle \Psi \mid \Psi_x^f \right\rangle - \left\|\left|\Psi_x^f\right\rangle\right\|^2\right)$. A real Block Set implies that $|\Psi\rangle$ and $\left|\Psi_x^f\right\rangle$ are real unit vectors, then $\left\langle \Psi_x^f \mid \Psi \right\rangle = \left\langle \Psi \mid \Psi_x^f \right\rangle$. Finally $\||\Psi\rangle\| = \left\|\left|\Psi_x^f\right\rangle\right\| = 1$ implies that $\langle \Psi_x^+ \mid \Psi_x^- \rangle = 0$. □

Theorem 13. *If A, B, C and D are real. Then*

$$\left\langle \Psi_x^f \mid \Psi_y^f \right\rangle = 2\left(\||A\rangle\|^2 - 2\langle A \mid C\rangle + \||C\rangle\|^2\right) - 1. \tag{3.24}$$

Proof. Applying Lemma 4 to $\left(|\Psi_x^+\rangle, |\Psi_x^-\rangle\right)$ and $\left(\left|\Psi_y^+\right\rangle, \left|\Psi_y^-\right\rangle\right)$ we get:

$$\langle A+B \mid C+D\rangle = \langle A \mid C\rangle + \langle B \mid C\rangle + \langle A \mid D\rangle + \langle B \mid D\rangle = 0 \tag{3.25}$$

and

$$\langle A+D \mid B+C\rangle = \langle A \mid B\rangle + \langle D \mid B\rangle + \langle A \mid C\rangle + \langle D \mid C\rangle = 0. \tag{3.26}$$

From equations (3.25) and (3.26) it follows that:

$$-\left(\langle A \mid C\rangle + \langle B \mid D\rangle\right) = \langle A \mid B\rangle + \langle C \mid D\rangle = \langle B \mid C\rangle + \langle A \mid D\rangle . \tag{3.27}$$

From equation (3.23) we have:

$$\left\langle \Psi_x^f \mid \Psi_y^f \right\rangle =$$

$$2\left(\||A\rangle\|^2 + \langle A \mid D\rangle + \langle B \mid A\rangle + 2\langle B \mid D\rangle + \langle C \mid B\rangle + \||C\rangle\|^2 + \langle D \mid C\rangle\right) - 1. \tag{3.28}$$

Applying equation (3.27) on equation (3.28), there is finally:

$$\left\langle \Psi_x^f \mid \Psi_y^f \right\rangle = 2\left(\||A\rangle\|^2 - 2\langle A \mid C\rangle + \||C\rangle\|^2\right) - 1. \tag{3.29}$$

□

This last theorem gives us a way to obtain the Gram Matrix of final states, directly from a given Block Set.

Let $\mathcal{B} = \left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ be a Block Set for (H_Q, H_W). We denote $k = (k_0, k_1, ..., k_t)$ and the following subsets of $\mathcal{B}$:

1. $\mathcal{B}_x^+ = \left\{|\Psi(k)\rangle \in H_A : (-1)^{\sum_{i=0}^t x_{k_i}} = 1\right\}$.
2. $\mathcal{B}_x^- = \left\{|\Psi(k)\rangle \in H_A : (-1)^{\sum_{i=0}^t x_{k_i}} = -1\right\}$.

Then $\widetilde{A}_{xy} = \mathcal{B}_x^+ \cap \mathcal{B}_y^+$ and $\widetilde{C}_{xy} = \mathcal{B}_x^- \cap \mathcal{B}_y^-$.

Notice that $\mathcal{B}_x^+$ and $\mathcal{B}_x^-$ are the sets of the positive and negative terms in Eq. (3.7) (on page 19), respectively. So for each pair x, y; $\widetilde{A}_{xy}$ and $\widetilde{C}_{xy}$ contain vectors of a Block Set, whose sum is $|A\rangle$ and $|C\rangle$, respectively.

Lemma 5. *Let $\mathcal{B} = \left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ be a Block Set for (H_Q, H_W), where:*

- *$P(k)$ is the set of pairs (x, y) such that $|\Psi(k)\rangle \in \widetilde{A}_{xy}$.*
- *$Q(k)$ is the set of pairs (x, y) such that $|\Psi(k)\rangle \in \widetilde{C}_{xy}$.*

Then $P(k) = \{x : (x_{k_0} \oplus ... \oplus x_{k_t}) = 0\}^2$ and $Q(k) = \{x : (x_{k_0} \oplus ... \oplus x_{k_t}) = 1\}^2$.

Proof. Using the definitions of $\widetilde{A}_{xy}$ and $\widetilde{C}_{xy}$, we have

$$P(k) = \left\{(x,y) : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = 1\right) \wedge \left((-1)^{\sum_{i=0}^{t} \delta_{k_i}(S_y)} = 1\right)\right\} \tag{3.30}$$

and

$$Q(k) = \left\{(x,y) : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = -1\right) \wedge \left((-1)^{\sum_{i=0}^{t} \delta_{k_i}(S_y)} = -1\right)\right\}. \tag{3.31}$$

As x does not have any influence in the Boolean value to the predicates of y and vice-versa, then the set of possible values for x and y form the Cartesian products

$$P(k) = \left\{x : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = 1\right)\right\} \times \left\{x : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = 1\right)\right\} \tag{3.32}$$

and

$$Q(k) = \left\{x : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = -1\right)\right\} \times \left\{x : \left((-1)^{\sum_{i=0}^{t} x_{k_i}} = -1\right)\right\}. \tag{3.33}$$

□

We introduce the following square matrices, whose entries only take values zero or one. Rows and columns are indexed by the elements of $\{0,1\}^n$.

- $\bar{P}_{k,h}$, where the element in row x and column y has value 1 $\iff$ $(x,y) \in (P(k) \cap P(h))$.
- $\bar{Q}_{k,h}$, where the element in row x and column y has value 1 $\iff$ $(x,y) \in (Q(k) \cap Q(h))$.
- $\bar{R}_{k,h}$, where the element in row x and column y has value 1 $\iff$ $(x,y) \in (P(k) \cap Q(h))$.

Theorem 14. *Let $\mathcal{B} = \left\{|\Psi(k)\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ be a real Block Set for (H_Q, H_W), then the Gram matrix of their output states $\left\{\left|\Psi_x^f\right\rangle\right\}$ is:*

$$G = 2\left(\sum_{k,h}\left(\bar{P}_{k,h} - 2\bar{R}_{k,h} + \bar{Q}_{k,h}\right)\langle\Psi(k) \mid \Psi(h)\rangle\right) - J. \tag{3.34}$$

where J is the matrix with all elements equal to 1.

Proof. Using Theorem 13, for all (x,y):

$$\left\langle \Phi_x^f \mid \Phi_y^f \right\rangle =$$

$$2\left(\sum_{|\Phi(k_1)\rangle,|\Phi(k_2)\rangle\in\widetilde{A}_{xy}} \langle \Phi\left(k_1\right) \mid \Phi\left(k_2\right)\rangle\right)$$

$$-4\left(\sum_{|\Phi(k_1)\rangle\in\widetilde{A}_{xy},|\Phi(k_2)\rangle\in\widetilde{C}_{xy}} \langle \Phi\left(k_1\right) \mid \Phi\left(k_2\right)\rangle\right)$$

$$+2\left(\sum_{|\Phi(k_1)\rangle,|\Phi(k_2)\rangle\in\widetilde{C}_{xy}} \langle \Phi\left(k_1\right) \mid \Phi\left(k_2\right)\rangle\right) - 1 \tag{3.35}$$

As:

- $\left(\left|\Phi\left(k_1\right)\right\rangle, \left|\Phi\left(k_2\right)\right\rangle \in \widetilde{A}_{xy}\right) \iff \left((x,y) \in P_{k_1} \wedge (x,y) \in P_{k_2}\right) \iff$ $(x,y) \in P_{k_1} \cap P_{k_2} \iff \bar{P}_{k_1,k_2}\left[x,y\right] = 1.$
- $\left(\left|\Phi\left(k_1\right)\right\rangle \in \widetilde{A}_{xy}, \left|\Phi\left(k_2\right)\right\rangle \in \widetilde{C}_{xy}\right) \iff \left((x,y) \in P_{k_1} \wedge (x,y) \in Q_{k_2}\right) \iff$ $(x,y) \in P_{k_1} \cap Q_{k_2} \iff \bar{R}_{k_1,k_2}\left[x,y\right] = 1.$
- $\left(\left|\Phi\left(k_1\right)\right\rangle, \left|\Phi\left(k_2\right)\right\rangle \in \widetilde{C}_{xy}\right) \iff \left((x,y) \in Q_{k_1} \wedge (x,y) \in Q_{k_2}\right) \iff$ $(x,y) \in Q_{k_1} \cap Q_{k_2} \iff \bar{Q}_{k_1,k_2}\left[x,y\right] = 1.$

Then Equation (3.35) turns into Equation (3.34). □

This theorem implies that products of components in the Block Set determine the Gram matrix of final states. Each matrix $\bar{P}$, $\bar{R}$ and $\bar{Q}$ acts like masks over the Gram matrix. There is also a particular case computationally less powerful, however with a simpler Gram matrix representation.

Definition 10. *A Block Set* $\mathcal{B} = \left\{\left|\Psi\left(k\right)\right\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ *for* (H_Q, H_W) *is orthogonal, if all its elements are pairwise orthogonal.*

Corollary 2. *If* $\mathcal{B} = \left\{\left|\Psi\left(k\right)\right\rangle \in H_A : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ *is an orthogonal real Block Set for* (H_Q, H_W)*, then the Gram matrix of their output states* $\left\{\left|\Psi_x^f\right\rangle\right\}$ *is*

$$G = 2\left(\sum_k \left(\bar{P}_{k,k} + \bar{Q}_{k,h}\right) \left\|\left|\Psi\left(k\right)\right\rangle\right\|^2\right) - J.$$

Proof. Apply $\langle \Psi\left(k\right) \mid \Psi\left(h\right)\rangle = 0$ for $k \neq h$ and $\bar{R}_{k,k} = 0$ for all k; in Eq. (3.34). □

Now, we extend our last example of Block Set obtained from Deutsch's algorithm. All one dimensional Block Sets are orthogonal, thus this algorithm is represented by Corollary 2. Thus, we are only interested in matrices controled by a single element

of the Block Set, $|\Psi(k)\rangle$, that is matrices of the form $\overline{P}_{k,k} + \overline{Q}_{k,k}$. The matrices controled by each element k are

$$\overline{P}_{0,0} + \overline{Q}_{0,0} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{3.36}$$

$$\overline{P}_{1,1} + \overline{Q}_{1,1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.37}$$

and

$$\overline{P}_{2,2} + \overline{Q}_{2,2} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{3.38}$$

Thus, taking the equations (3.11), (3.12) and (3.13) (on page 25), we calculate

$$M = \sum_{i=0}^{2} \left(\overline{P}_{i,i} + \overline{Q}_{i,i}\right) \langle \Psi(i) | \Psi(i) \rangle = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}. \tag{3.39}$$

We finally obtain the Gram matrix of Deutsch's algorithm from Corollary 2

$$G = 2(M) - \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}. \tag{3.40}$$

We can see that M is the sum of our weighted masks and obtaining $M[x, y] = \frac{1}{2}$ guarantees the orthogonality between the final states of x and y.

3.2 Towards a framework for constructing quantum exact query algorithms

In the last section we identify matrices that describe the influence of each element from the Block Set, in this section we apply such matrices in constructing systems

of equations that characterize the existence of exact quantum algorithms.

We define next the set of unknowns $\left\{w_{kh} : k, h \in \mathbb{Z}_{n+1}^{t+1}\right\}$, two disjoint sets $X, Y \subset \{0,1\}^n$ and the set $SI_i(k') = \left\{k \in (\mathbb{Z}_{n+1})^{t+1} : \forall j,\ (0 \leq j \leq i) \wedge \left(k'_j = k_j\right)\right\}$. We define the following equations:

1. For each $(x, y) \in X \times Y$

$$\sum_{k,h \in \mathbb{Z}_{n+1}^{t+1}} \left(\bar{P}_{k,h}[x,y] - 2\bar{R}_{k,h}[x,y] + \bar{Q}_{k,h}[x,y]\right) w_{kh} = \frac{1}{2}. \tag{3.41}$$

2. For each $i \in \mathbb{Z}_{t+1}$ and $k', h' \in (\mathbb{Z}_{n+1})^{i+1}$, such that $k'_i \neq h'_i$

$$\sum_{k \in SI_i(k')} \left(\sum_{h \in SI_i(h')} w_{kh} \right) = 0. \tag{3.42}$$

3. A unique equation

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} w_{kk} = 1. \tag{3.43}$$

The union of these equations is denoted by $E(t, n, X, Y)$.

Theorem 15. *Consider a function $f : \{0,1\}^n \to \{0,1\}$ where $(x \in X) \wedge (y \in Y)$ implies that $f(x) \neq f(y)$. Then there is an exact quantum algorithm for f, that uses $t+1$ queries, if and only if $E(t, n, X, Y)$ has a real solution for $\left\{w_{kh} : k, h \in \mathbb{Z}_{n+1}^{t+1}\right\}$, where these values form a positive semi-definite matrix with such indices.*

Proof. Proving that the condition is necessary; if a quantum query algorithm $\mathcal{A}$ computes f exactly within $t+1$ queries, then a set $\left\{|\Psi(k)\rangle : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$ is associated to the algorithm $\mathcal{A}$ and by Theorem 10 (on page 21) this set is a Block Set. If such Block Set is complex, Lemma 3 (on page 27) says that there is another real Block Set $\left\{\left|\widehat{\Psi}(k)\right\rangle : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$, whose output states can compute function f exactly.

We take $w_{k_1 k_2} = \left\langle \widehat{\Psi}(k_1) \mid \widehat{\Psi}(k_2) \right\rangle$. Considering that f is computed exactly; if $x \in X$, $y \in Y$ then $f(x) \neq f(y)$ and the output states of $\mathcal{A}$ are orthogonal $\left\langle \widehat{\Psi}_x^f \mid \widehat{\Psi}_y^f \right\rangle = 0$. By Theorem 14 (on page 29), Eq. (3.41) for (x, y) is satisfied, because $\mathcal{A}$ and the Block Set have the same Gram matrix for their output states. Eq. (3.42) follows straightforward by property 1 in Definition 8. Eq. (3.43) comes from property 2 in Definition 8. Finally, $\{w_{k_1 k_2}\}$ forms a positive semi-definite matrix, because it was defined as the Gram Matrix of $\left\{\left|\widehat{\Psi}(k)\right\rangle\right\}$.

In the proof about the second condition, considering that the values for $\{w_{k_1 k_2}\}$ form a positive semi-definite matrix, then it is also a Gram matrix for some set of vectors $\left\{|\Psi(k)\rangle : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$. Such vectors fulfil property 1 in Definition 8 (on

page 8) by Eq. (3.42) and property 2 in Definition 8 by Eq. (3.43). Giving the appropriate dimension to the spaces H_1 and H_2, then properties 3 and 4 in Definition 8 are satisfied and vectors $\{\Psi(k)\}$ form a Block Set. The sets of output states $\left\{\left|\Psi_x^f\right\rangle, x \in X\right\}$ and $\left\{\left|\Psi_x^f\right\rangle, x \in Y\right\}$ generate two orthogonal spaces, by Eq. (3.41) and Theorem 14. Thus there is a CSOP that can measure the correct output exactly. Finally, we conclude that a quantum query algorithm associated to $\{\Psi(k)\}$ and the CSOP computes f exactly in $t+1$ queries, by Theorems 9 and 11 (on pages 19 and 22, respectively). □

System $E(t, n, X, Y)$ has a problem, it has an exponential number of variables. Solving these equations numerically is impractical and there are also some difficulties in using such a system as an analytical tool. Part of the problem is maintaining the semi-definite property in the solution. However, we can consider special cases of the general formulation. For example, we can set some variables to be equal to zero, that allows us to construct particular families of exact quantum algorithms more easily. The next theorem uses such a strategy in obtaining a more practical tool.

Take $\widehat{E}(t, n, X, Y)$ to be the union of the following equations:

1. For each pair $(x, y) \in X \times Y$ add the equation:

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} \left(\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y]\right) w_{kk} = \frac{1}{2}. \tag{3.44}$$

2. Also add the unique equation

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} w_{kk} = 1. \tag{3.45}$$

Corollary 3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function, such that $(x \in X) \wedge (y \in Y)$ implies $f(x) \neq f(y)$. If the system $\widehat{E}(t, n, X, Y)$ has a solution over the non-negative real numbers, then a quantum query algorithm computes f in exactly $t+1$ queries.*

Proof. If the Block Set is orthogonal (see Definition 10 on page 30) and computes f, the condition $w_{k_1 k_2} = \langle \Psi(k_1) \mid \Psi(k_2) \rangle$ implies that $(k_1 \neq k_2 \Rightarrow w_{k_1 k_2} = 0)$. Therefore Eq. (3.43) becomes equal to Eq. (3.45). Eq. (3.42) vanishes and as $\bar{R}_{k,k} = 0$ then Eq. (3.41) produces Eq. (3.44). Finally, a matrix formed by $w_{k_1 k_2}$ gives non-negative values in the diagonal and zero in the rest, which guarantees the positive semi-definite property. □

Corollary 3 is a simpler tool than Theorem 15, because each $k \in \mathbb{Z}_{n+1}^{t+1}$ has a influence to the Gram matrix that is independent of the other elements in $\mathbb{Z}_{n+1}^{t+1}$.

Defining T_k as the set of pairs (x, y) such that $\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y] = 1$, we can claim that the influence of T_k on the Gram matrix is given by w_{kk} and the intersection of those sets determines the regions of $\{0,1\}^n \times \{0,1\}^n$ that fulfil Eq. (3.44). Satisfying such equation is equivalent to those regions having value 0 in the Gram matrix, and thus such sets condition which inputs can be calculated exactly for a given algorithm. But the weight that we can assign to each k is limited by Eq. (3.45). We can observe that increasing t increases the possible shapes for T_k and enlarges the range of possible Gram matrices that we can construct. We can consider a random procedure for obtaining exact quantum algorithms by assigning weights for some set of variables $\left\{w_{kk} : k \in L \subset \mathbb{Z}_{n+1}^{t+1}\right\}$ until we reach the limit of Eq. (3.45), the final step is finding interesting sets X and Y such that $\left(x \in X \wedge y \in Y \Leftrightarrow \sum_k \left(\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y]\right) w_{kk} = \frac{1}{2}\right)$. In conclusion, designing exact algorithms using Corollary 3 is about analysing intersections between elements in $\left\{T_k : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$, which needs Corollary 2 (on page 30).

Using Corollary 3, we have two properties about orthogonal block sets. Taking vector $\overline{x} \in \{0,1\}^n$ for $x \in \{0,1\}^n$, such that $\forall i,\ x_i \neq \overline{x}_i$. For all k: $\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y] = \bar{P}_{k,k}[\overline{x}, \overline{y}] + \bar{Q}_{k,k}[\overline{x}, \overline{y}]$ and thereby all possible Gram matrices G obtained using the corollary satisfy $G[x, y] = G[\overline{x}, \overline{y}]$. Second, defining $p(k)$ as the set of permutations for k, then $\bar{P}_{k,k} + \bar{Q}_{k,k} = \bar{P}_{k',k'} + \bar{Q}_{k',k'}$ for all $k' \in p(k)$. Suppose that we assign random values to the unknowns $W(k) = \{w_{k'k'} : k' \in p(k)\}$. If $v = \sum_{k' \in W(k)} w_{k'k'}$ remains constant, this implies no change over the Gram Matrix.

System $\widehat{E}(t, n, X, Y)$ represents a clear view on how Orthogonal BSF algorithms act over the Gram matrix of final states. But, we can obtain the same algorithms using a smaller system. Let $(x \oplus y) \in \{0,1\}^n$ be the xor bitwise operation for $x, y \in \{0,1\}^n$. System $\widetilde{E}(t, n, X, Y)$ is defined as the union of the following equations:

1. For each $(x, y) \in X \times Y$

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} \bar{P}_{k,k}[0^n, x \oplus y]\, w_{kk} = \frac{1}{2}. \tag{3.46}$$

2. An unique equation

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} w_{kk} = 1. \tag{3.47}$$

Theorem 16. *System $\widehat{E}(t, n, X, Y)$ is equivalent to the system $\widetilde{E}(t, n, X, Y)$.*

Proof. We have the identity $\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y] = \bar{P}_{k,k}[0^n, x \oplus y]$. Finally

$$\sum_{k \in \mathbb{Z}_{n+1}^{t+1}} \left(\bar{P}_{k,k}[x, y] + \bar{Q}_{k,k}[x, y]\right) w_{kk} = \sum_{k \in \mathbb{Z}_{n+1}^{t+1}} \bar{P}_{k,k}[0^n, x \oplus y]\, w_{kk}. \tag{3.48}$$

□

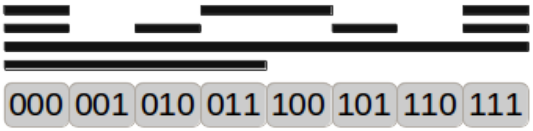

Figure 3.3: Every black layer represents the influence of some formula over each x. From top to bottom, these are $x_1 \oplus x_2$, $x_1 \oplus x_3$, x_0 and x_3. If we give weight $\frac{1}{4}$ to these formulas, then any x with exactly two layers over it is orthogonal to 000. In this case 001, 100 and 101.

Notice that system $\widetilde{E}(t, n, X, Y)$ is the same as $\widehat{E}(t, n, 0^n, Z)$, where we define $Z = \{x \oplus y : (x, y) \in X \times Y\}$. This implies that if an exact Orthogonal BSF algorithm separates 0^n from Z, then it also can be used for separating X from Y without error. Thus for Orthogonal BSF algorithms we simplify our problem, just determining which sets can be separated from 0^n for a bounded t. We state a problem denoted as the XOR-WEIGHTED problem: find the values $\{w_{kk}\}$ where a set of Boolean formulas

$$\left\{ \bigoplus_i x_{k_i} : x_0 = 0, k \in K \subset \mathbb{Z}_{n+1}^{t+1} \right\},$$

has each formula associated to a value $w_{kk} > 0$, such that (i) $\sum_{k \in K} w_{kk} = 1$ and (ii) for any $z \in Z$ the sum of values from formulas not satisfied by z is $\frac{1}{2}$. Therefore, systems $\widetilde{E}(t, n, X, Y)$ and $\widehat{E}(t, n, 0^n, Z)$ are equivalent to solving the XOR-WEIGHTED problem and solving the XOR-WEIGHTED problem is equivalent to obtaining an exact BSF algorithm that separates X from Y within $t+1$ queries. Figure 3.3 shows an example of this alternative formulation.

3.2.1 A generalization of the Deutsch-Jozsa algorithm

We present an example of BSF algorithm, that was obtained by our analytical tools. Assume that n is even and $n > 2t$. We take a set $\{k_i / 0 < i \leq n\} \subset \mathbb{Z}_{n+1}^{t+1}$, with $k_i = (r(i), r(i+1), ..., r(i+t))$, such that $r(i) = i$ for $i \leq n$ and $r(i) = (i-n)$ for $i > n$. Define $S(x)$ as the number of satisfied Boolean clauses $\phi_i = x_{r(i)} \oplus x_{r(i+1)} \oplus ... \oplus x_{r(i+t)}$, such that $0 < i \leq n$. Taking $w_{k_i k_i} = \frac{1}{n}$ for all $0 < i \leq n$, solves $\widehat{E}(t, n, X, Y)$ for $X = \{0^n, 1^n\}$ and $Y = \left\{x \in \{0,1\}^n / S(x) = \frac{n}{2}\right\}$. We affirm that $\widehat{E}(t, n, X, Y)$ is solved by the next observations. Equation

$$\sum_i \left(\bar{P}_{k_i, k_i}[0^n, y] + \bar{Q}_{k_i, k_i}[0^n, y] \right) w_{k_i k_i} = \frac{1}{2} \tag{3.49}$$

is true only if there are $\frac{n}{2}$ matrices $\bar{P}_{k_i,k_i}$ equal to 1 in column y and row 0^n, because matrices $\bar{Q}_{k_i,k_i}$ only have values zero on row 0^n. This is equivalent to $S(y) = \frac{n}{2}$. Finally, considering that $\left(S(x) = \frac{n}{2} \Rightarrow S(\overline{x}) = S(x)\right)$, then Eq. (3.49) is satisfied for $\overline{y}$. Applying $G[x,y] = G[\overline{x},\overline{y}]$, there is

$$\sum_i \left(\bar{P}_{k_i,k_i}[1^n,y] + \bar{Q}_{k_i,k_i}[1^n,y]\right) w_{k_i k_i} = \frac{1}{2} \tag{3.50}$$

for any y where $S(y) = \frac{n}{2}$.

Using Corollary 3 (on page 33), an exact quantum algorithm computes two different outputs for X and Y. We detail the first two cases:

- For $t = 0$ we have a BSF algorithm equivalent to Deutsch-Jozsa algorithm [11].
- For $t = 1$ we have a BSF algorithm that separates $\{0^n, 1^n\}$ from all x, where in x occurs $n/2$ times that the i-th bit is equal to the next one, defining the first bit as next of the last bit.

3.3 A lower bound for exact quantum algorithms

In this part, we apply the BSF approach for developing a lower bound result for exact quantum algorithms. This lower bound is stated for functions with Boolean domain and arbitrary output.

We introduce some mathematical constructions. We take a basis for the linear space generated by Boolean functions [35]. That is a set of functions

$$\mathcal{F}_k^n : \{0,1\}^n \to \{1,-1\},$$

where each $\mathcal{F}_k^n(x) = \prod_{i=0}^{n-1}(-1)^{x_{k_i}}$ is indexed by vectors $k \in \mathbb{Z}_n^n$.

If $k \neq h$, there is the possibility of having $\mathcal{F}_k^n = \mathcal{F}_h^n$. Thus, we define an equivalence relation $k \sim h$, for $k,h \in \mathbb{Z}_n^n$ where $\mathcal{F}_k^n = \mathcal{F}_h^n$. We take the sets $\mathbb{S}_n$ and $[k] \in \mathbb{S}_n$ as the quotient set of our relation and the equivalence class for element k, respectively. We also take a) $\mathbb{F}_n$, whose elements are functions indexed by $\mathbb{S}_n$ where $\mathcal{F}_{[h]}^n = \mathcal{F}_k^n$ iff $k \in [h]$, and b) $\mathbb{F}_n(m) \subset \mathbb{F}_n$, where $\mathcal{F}_{[k]} \in \mathbb{F}_n(m)$ iff $[k]$ has an element h such that the number of non-zero terms is not bigger than $2m$. We introduce $\mathcal{F}_{[k,h]}^n : \{0,1\}^n \to \{0,1\}$, such that it has value 1 iff $\mathcal{F}_{[k]}^n(x) = \mathcal{F}_{[h]}^n(x) = 1$. Observe that $\mathcal{F}_{[k,h]}^n(x) = \frac{\mathcal{F}_{[k]}^n(x) + \mathcal{F}_{[h]}^n(x) + \mathcal{F}_{[k\circ h]}^n(x) + 1}{4}$, such that the set $[k \circ h] \in \mathbb{S}_n$ is an equivalence class where $\mathcal{F}_{[k\circ h]}^n(x) = 1$ iff $\mathcal{F}_{[k]}^n(x) = \mathcal{F}_{[h]}^n(x)$.

In order to obtain a exact quantum algorithm that computes a given function, some regions in the Gram matrix of final states must to be equal to 0. More specifi-

cally, if some exact quantum algorithm separates an input x from some set of inputs Y, then $G_{[x,x\oplus y]} = 0$ for all $y \in Y$. Using the values of the Gram matrix on row x, we can define a function $\frac{1}{2}\left(G_{[x,x\oplus w]} + 1\right)$ depending on w. If the quantum algorithm applies no more than k queries, then such function can be decomposed in a sum of functions $\alpha_{[h]}\mathcal{F}^n_{[h]}(w)$, where each $\mathcal{F}^n_{[h]} \in \mathbb{F}_n(k)$ and $\sum_{[h]\in\mathbb{S}_n} \alpha_{[h]} = 1$. Thus, we have a necessary condition for an exact quantum algorithm separating x from Y in k queries. This condition is the existence of some function g whose sum of coefficients $\alpha_{[h]}$ is equal or bigger than 1 (where we just consider functions from $\mathbb{F}_n(k)$) and such function has value $\frac{1}{2}$ in $z = x \oplus y$ for all $y \in Y$.

Let $F : \{0,1\}^n \to \mathbb{R}$ be a function, we define an operation

$$F * \mathcal{F}^n_{[h]} = \sum_{x\in\{0,1\}^n} F(x)\,\mathcal{F}^n_{[h]}(x),$$

notice that

$$\frac{1}{2\sqrt{2^n}}\left(G_{[x,x\oplus w]} + 1\right) * \mathcal{F}^n_{[h]} = \alpha_{[h]}.$$

We can obtain a lower-bound method by taking the minimum k, such that there is a function g that satisfies some of our conditions. First, we need a function g that maximizes $\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} g * \mathcal{F}^n_{[h]}$. We must introduce additional notations. We define $\overline{a}$ as a vector whose all its terms are a, and $\rho(i) = 0$ if i is even, $\rho(i) = 1$ if i is odd. Let $X_i \subset \{0,1\}^n$ be m disjoint sets, where each $x \in X_i$ has a set $Z(x) = \{x \oplus y : y \in X_j \, and \, j \neq i\}$. We have a family of functions $g^k_y(x)$ where a) $g^k_y\left(\overline{0}\right) = 1$, b) $g^k_y(x) = \frac{1}{2}$ for $x \in Z(y)$, c) $g^k_y(x) = 1$ for x, such that

$$\sum_{i=0}^{2k}\sum_{j=0}\binom{|x|}{i-2j-\rho(i)}\binom{n-|x|}{2j+\rho(i)} > \frac{\sum_{i=0}^{2k}\binom{n}{i}}{2} \tag{3.51}$$

and d) $g^k_y(x) = 0$ otherwise. Next theorem applies the introduced ideas.

Theorem 17. *If condition* $\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} g^k_y * \mathcal{F}^n_{[h]} \geq 1$ *is fulfilled for all* $y \in \bigcup_i X_i$, *then there is an exact quantum algorithm that produces different outputs for each* X_i *and it applies at least* k *queries.*

Proof. Consider a quantum algorithm that separates $x \in X_i$ from $\bigcup_{j\neq i} X_j$, by applying k queries and error zero. The Gram matrix representation of Theorem 14 (on page 29) at row x, gives us

$$\frac{1}{2}\left(G_{[x,x\oplus y]} + 1\right) = \sum_{[h]\in\mathbb{S}_n} \alpha_{[h]}\mathcal{F}^n_{[h]}(y) + \sum_{[h_i]\neq[h_j]} \alpha_{[h_i,h_j]}\mathcal{F}^n_{[h_i,h_j]}(y). \tag{3.52}$$

Taking $\overline{T}_{h_1,h_2} = \overline{P}_{h_1,h_2} - 2\overline{R}_{h_1,h_2} + \overline{Q}_{h_1,h_2}$ where we consider the matrices in Eq. (3.34) (on page 29), we can see that first sum in Eq. 3.52 follows from $\overline{T}_{h_1,h_2}$ when $h_1 = h_2$ and second sum follows from $\overline{T}_{h_1,h_2}$ when $h_1 \neq h_2$. Thereby, there are $\sum_{[h]\in\mathbb{S}_n} \alpha_{[h]} = 1$ and $\sum_{[h_i]\neq[h_j]} \alpha_{[h_i,h_j]} = 0$, which give

$$\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} \frac{1}{2}\left(G_{[x,x\oplus y]} + 1\right) * \mathcal{F}^n_{[h]} = \sqrt{2^n}. \tag{3.53}$$

We can formulate the necessary condition of orthogonality between the final states indexed by $x \in X_i$ and $\bigcup_{j\neq i} X_j$, where the algorithm is restricted to k queries. Such condition is the existence of a function $g : \{0,1\}^n \to [0,1]$, where $g(x) = \frac{1}{2}$ for $x \in \bigcup_{j\neq i} X_j$, $g\left(\overline{0}\right) = 1$ and $\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} g * \mathcal{F}^n_{[h]} \geq \sqrt{2^n}$. Function $g^k_y(x)$ satisfies such restrictions and maximizes $\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} g * \mathcal{F}^n_{[h]}$. In other words, for $x \notin \bigcup_{j\neq i} X_j - \left\{\overline{0}\right\}$ we have $g^k_y(x) = 1$ if there are more functions in $\mathbb{F}_n(k)$ with value 1 than -1 in x, and $g^k_y(x) = 0$ otherwise. Observe that $\sum_{i=0}^{2k} \binom{n}{i}$ represents the cardinality of set $\mathbb{F}_n(k)$ and

$$\sum_{i=0}^{2k}\sum_{j=0} \binom{|x|}{i-2j-\rho(i)} \binom{n-|x|}{2j+\rho(i)}$$

is the number of functions contained in $\mathbb{F}_n(k)$ that have value 1 in x. □

Therefore, this theorem implies another lower-bound different from main approaches like Polynomial and Adversary methods. The lower bound is the minimum value k such that all functions g^k_y satisfy the condition. We give an example for this approach, we apply it to a total function f that give different outputs to $X_1 = \left\{\overline{0},\overline{1}\right\}$ and $X_2 = \{0,1\}^n - X_1$. For all $y \in X_1$, there is $g^k_y(x) = \widehat{g}_y(x) + \widetilde{g}_y(x)$. Where such functions satisfies these properties a) $\widehat{g}_y\left(\overline{0}\right) = 1$ and $\widehat{g}_y(x) = \frac{1}{2}$ for $x \neq \overline{0}$, b) $\widetilde{g}_y\left(\overline{1}\right) = \frac{1}{2}$ and $\widetilde{g}_y(x) = 0$ for $x \neq \overline{1}$. Function g^k_y is selected under the assumption $k \leq \left\lfloor \frac{n}{2} \right\rfloor$, because Eq. (3.51) for $x = \overline{1}$ takes form

$$\sum_{i=0}^{k} \binom{n}{2i} > \frac{\sum_{i=0}^{2k} \binom{n}{i}}{2}.$$

Thus, there is

$$\sum_{\mathcal{F}^n_{[h]}\in\mathbb{F}_n(k)} \widehat{g}_y * \mathcal{F}^n_{[h]} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2k} \binom{n}{i}$$

and

$$\sum_{\mathcal{F}_{[h]}^n \in \mathbb{F}_n(k)} \widetilde{g}_y * \mathcal{F}_{[h]}^n = \frac{1}{2\sqrt{2^n}} \sum_{i=0}^{k} \left(\begin{array}{c} n \\ 2i \end{array} \right).$$

Taking $k = \left\lceil \frac{4n}{10} \right\rceil$, there is

$$\sum_{i=0}^{\left\lceil \frac{4n}{10} \right\rceil} \left(\begin{array}{c} n \\ 2i \end{array} \right) > 4 \left[\sum_{i=2\left\lceil \frac{4n}{10} \right\rceil}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) \right].$$

The calculations and selection of k can be improved, nevertheless, that result is enough for proving

$$\sum_{\mathcal{F}_{[h]}^n \in \mathbb{F}_n\left(\left\lceil \frac{4n}{10} \right\rceil\right)} g_y^{\left\lceil \frac{4n}{10} \right\rceil} * \mathcal{F}_{[h]}^n \geq \sqrt{2^n},$$

which implies the following lower bound $Q_E(f) = \Omega(n)$.

Chapter 4

Quantum speed-up and quantum query

A central problem in quantum computing is understanding the conditions that produce a computational advantage in relation to classical computing. Such question can be studied from two approaches a) determining which *functions* or b) determining which *algorithms* allow an advantage between quantum and classical computing. The first approach is the core of quantum query complexity, this mainly consists in obtaining bounds or relations for complexity measures and evaluating their tightness [8, 36]. The second approach tries to identify quantum features that are hard to simulate for classical sources [37]. Among the earliest attempts within this second approach, there is the hypothetical quantum parallelism on quantum algorithms [38]. Another quantum feature is quantum entanglement [39], that is identified as a necessary condition for quantum speed-up in pure-state algorithms [40]. In the literature, we find other quantum properties for explaining such quantum gain [41–44]. However, these quantum features were directly applied to models equivalent to Turing machines. In this chapter, we study a property restricted just to quantum queries.

4.1 A simulation defined over decompositions

In this section, we introduce a simulation of quantum query algorithms by classical algorithms. The idea applies the following corollary from Theorem 9 (on page 19).

Corollary 4. *Given an indexed set of vectors* $\left\{|\Psi(k)\rangle \in \mathcal{H} : k \in \mathbb{Z}_{n+1}^{t+1}\right\}$*, such that a CSOP* $\{P_z : z \in T\}$ *defines its final measurement, the probability of obtaining* z *given an input* x *is*

$$\pi_x(z) = \sum_{k,h} (-1)^{\gamma_x(k)+\gamma_x(h)} \langle\Psi(k)| P_z |\Psi(h)\rangle , \tag{4.1}$$

where $\gamma_x(k) = \sum_{i=0}^{t} x_{k_i}$.

Proof. Taking Eq. 3.7 (on page 19) from Theorem 9

$$\left|\Psi_x^f\right\rangle = \widetilde{U}_t^{\dagger} O_x U_t \ldots O_x U_0 \left|\Psi\right\rangle = \sum_k (-1)^{\sum_{i=0}^{t} x_{k_i}} \left|\Psi(k)\right\rangle.$$

As $P_z{=}P_z^2$, then

$$\pi_x(z) = \left\|P_z \left|\Psi_x^f\right\rangle\right\|^2 = \sum_{k,h} (-1)^{\sum_{i=0}^{t} x_{k_i} + \sum_{i=0}^{t} x_{h_i}} \left\langle\Psi(k)\right| P_z \left|\Psi(h)\right\rangle.$$

□

We apply again the basis for the Boolean cube [35] and functions introduced in Section 3.3 (on page 36).

Lemma 6. *Suppose that* $\mathcal{A}$ *is a quantum algorithm that applies* t *queries. Let* $\pi_x(1) = \sum_{[k]\in\mathbb{S}_n} \alpha_{[k]} \mathcal{F}_{[k]}^n$ *be the output probability of obtaining* 1 *from algorithm* $\mathcal{A}$*. If the class* $[k]$ *contains an element* $k = (k_0, k_1, .., k_{n-1})$ *with* $2t+1$ *distinct terms* $k_i \neq 0$*, then* $\alpha_{[k]} = 0$*.*

Proof. The proof consists in rewriting Corollary 4. □

Theorem 18. *Consider a quantum algorithm* $\mathcal{A}$ *that computes* $f : S \to \{0,1\}$ *for* $S \subset \{0,1\}^n$*, within error* ε*. There is a classical algorithm that computes* f *within error*

$$\tilde{\varepsilon} = \frac{\varepsilon + L(\pi_x(1))}{1 + 2L(\pi_x(1))},$$

using the same number of queries as $\mathcal{A}$*.*

Proof. Applying Lemma 6, the output probability of algorithm $\mathcal{A}$ can be denoted as $\pi_x(1) = \sum_{[k]\in\mathbb{S}_n} \alpha_{[k]} \mathcal{F}_{[k]}^n$. Take $\mathcal{D}([k])$ to be a deterministic classical algorithm which outputs 1 with probability

$$\widehat{\pi}_x^{[k]}(1) = \frac{1}{2} + sgn\left(\alpha_{[k]}\right)\left(\frac{1}{2}\mathcal{F}_{[k]}^n\right), \tag{4.2}$$

for $[k] \in \mathbb{S}_n$ and within n queries. Let $\mathcal{R}$ be a classical randomized algorithm that simply executes:

- An algorithm $\mathcal{D}([k])$, with probability $\frac{2\left|\alpha_{[k]}\right|}{1+2L(\pi_x(1))}$.
- An algorithm that outputs 0 for any x, with probability $\frac{1}{1+2L(\pi_x(1))}$.

Denoting $\widehat{\pi}_x(1)$ the probability of obtaining output 1 given input x by algorithm $\mathcal{R}$, we have

$$\widehat{\pi}_x(1) = \frac{\sum_{[k]} 2\left|\alpha_{[k]}\right| \widehat{\pi}_x^{[k]}(1)}{1 + 2L(\pi_x(1))} \tag{4.3}$$

and applying Eq. 4.2, we have

$$\widehat{\pi}_x(1) = \frac{\sum_{[k]} \left|\alpha_{[k]}\right| + \sum_{[k]} \alpha_{[k]} \mathcal{F}_{[k]}^n}{1 + 2L(\pi_x(1))}. \tag{4.4}$$

Thus, we need to prove the theorem for two cases. (i) For x such that $f(x) = 1$, there is $\varepsilon \geq 1 - \pi_x(1) = 1 - \sum_{[k]} \alpha_{[k]} \mathcal{F}_{[k]}^n$. Then, we have

$$1 - \widehat{\pi}_x(1) = 1 - \frac{\left(L(\pi_x(1)) + \sum_{[k]} \alpha_{[k]} \mathcal{F}_{[k]}^n\right)}{1 + 2L(\pi_x(1))}$$

$$= \frac{1 + L(\pi_x(1)) - \sum_{[k]} \alpha_{[k]} \mathcal{F}_{[k]}^n}{1 + 2L(\pi_x(1))} \leq \tilde{\varepsilon}.$$

(ii) For x such that $f(x) = 0$, there is $\varepsilon \geq \pi_x(1) = \sum_{[k]} \alpha_{[k]} \mathcal{F}_{[k]}^n$ and we have

$$\widehat{\pi}_x(1) \leq \frac{\varepsilon + L(\pi_x(1))}{1 + 2L(\pi_x(1))} = \tilde{\varepsilon}. \tag{4.5}$$

□

We presented a classical simulation for a given quantum algorithm, although producing a big error. The next theorem shows a reduced error, by using probabilistic amplification.

Theorem 19. *Consider a quantum algorithm $\mathcal{A}$ that computes a function $f : S \to \{0,1\}$ for $S \subset \{0,1\}^n$, within error ε and applying t queries. Then, we can compute f within error $\exp\left(-\frac{k}{2(1-\tilde{\varepsilon})}\left(\frac{1}{2} - \tilde{\varepsilon}\right)^2\right)$, where $\tilde{\varepsilon} = \frac{\varepsilon + L(\pi_x(1))}{1 + 2L(\pi_x(1))}$, by a classical algorithm with kt queries.*

Proof. We apply the Chernoff bound [45]. If we take k, p, β where $0 \leq p \leq 1$, $0 \leq \beta \leq 1$ and $0 \leq k$, then

$$\sum_{i=0}^{m} \binom{k}{i} p^i (1-p)^{k-i} \leq \exp\left(-\beta^2 kp/2\right), \tag{4.6}$$

for $m = \lfloor (1-\beta) kp \rfloor$.

We obtain a new algorithm $\widehat{\mathcal{R}}$ from classical algorithm $\mathcal{R}$ within error $\tilde{\varepsilon}$ (recall that $\mathcal{R}$ is constructed for Theorem 18). Algorithm $\widehat{\mathcal{R}}$ is a probability amplification of $\mathcal{R}$, that is the execution of algorithm $\mathcal{R}$ k times and the selection of the most frequent result or randomly if two results have same frequency. The random variable X represents the number of correct answers in the repetition. Take $\beta = 1 - \frac{1}{2(1-\tilde{\varepsilon})}$ and $p = (1 - \tilde{\varepsilon})$ in Eq. (4.6), then

$$\mathbb{P}\left[X \leq \left\lfloor \frac{k}{2} \right\rfloor\right] \leq \exp\left(-\frac{k}{2\left(1-\tilde{\varepsilon}\right)}\left(\frac{1}{2} - \tilde{\varepsilon}\right)^2\right) \tag{4.7}$$

is an upper-bound for $\widehat{\mathcal{R}}$. □

4.2 Upper bounds for quantum speed-up

In this part, we introduce measures for quantum algorithms that can make our simulation less efficient. Quantum speed-up can occur only when there is not an efficient enough classical simulation. Therefore, if a condition makes difficult any classical simulation, then it is a necessary condition for a quantum speed-up. Next theorem shows how L_1-norm affects our simulation. Notice that there is still the possibility of a classical efficient simulation for some cases where such conditions apply.

Theorem 20. *Consider a function $f : \{0,1\}^n \to \{0,1\}$ that is computed within error $\varepsilon > 0$ and t queries, using a quantum query algorithm. Defining a function*

$$F_\varepsilon(x) = \left\lceil \frac{-16\ln(\varepsilon)(1+x)(1+x-\varepsilon)}{(1-2\varepsilon)^2} \right\rceil, \tag{4.8}$$

we have

$$\frac{R_\varepsilon(f)}{t} \leq F_\varepsilon\left(L\left(\pi_x(1)\right)\right), \tag{4.9}$$

where $R_\varepsilon(f)$ is the minimum number of queries for computing f within error ε by a classical algorithm.

Proof. We simulate the quantum algorithm using the classical algorithm from Theorem 19 and within an error that does not exceed ε for function f. From Eq. (4.7), we have

$$\varepsilon = \exp\left(-\frac{k}{2\left(1-\tilde{\varepsilon}\right)}\left(\frac{1}{2} - \tilde{\varepsilon}\right)^2\right). \tag{4.10}$$

Considering that $\frac{R_\varepsilon(f)}{t} \leq \lceil 2k \rceil$, if we obtain k from Eq. (4.10), then we get Eq. (4.9). □

The following example is an application of Theorem 20. We take Deutsch-Jozsa algorithm, that implies the next output probability for inputs of size n

$$\pi_x(1) = \frac{1}{n^2}(n - 2|x|)^2.$$

We obtain the terms $\{\alpha_{[k]}\}$ applying the orthogonality property of functions $\mathcal{F}^n_{[k]}$. Considering that the algorithm applies only one query, by Lemma 6 (on page 41) we have 3 kinds of equivalent classes that index our basis functions. First, there is the class $[k_0]$ which indexes the constant function $\mathcal{F}^n_{[k_0]}$. Then we have $\alpha_{[k_0]} = \frac{1}{n}$. Second, there are the classes which contain at least one element k, such that $k_i \neq 0$ for just one i. If $[k]$ is in that class, then $\alpha_{[k]} = 0$. Third, there are $\frac{n(n-1)}{2}$ classes which contain at least one element k, where there are just two indices $i \neq j$ such that $k_i \neq 0$, $k_j \neq 0$ and $k_i \neq k_j$. If $[k]$ is in that class, then $\alpha_{[k]} = \frac{2}{n^2}$. Thus we have $\sum |\alpha_{[k]}| = 1$ and it implies that

$$R_\varepsilon \leq \left\lceil \frac{-16\ln(\varepsilon)(2-\varepsilon)}{(1-2\varepsilon)^2} \right\rceil.$$

In absolute values, that is not quite tight because $R_0 = 2$. Our interest is in the asymptotic behaviour and that shows perfectly that Deutsch-Jozsa algorithm has a classical simulation with a constant number of queries and fixed error.

Observe that L_1-norm is defined over the algorithm itself, thus it does not necessarily depends on the function. Then, we can obtain an explicit expression for the L_1-norm from the algorithm itself. Denote the vectors h^1, h^2 in $\mathbb{Z}^t_{n+1}$ and $[k]$ in $\mathbb{S}_n$. We consider $(h^1, h^2) \sim [k]$, if

$$(-1)^{\sum_i x_{h^1_i} + \sum_i x_{h^2_i}} = (-1)^{\sum_i x_{k_i}},$$

for some $k \in [k]$.

Thus, if we consider a $t-$query algorithm, then we have the following equation

$$L(\pi_x(1)) = \sum_{[k]\in\mathbb{S}_n} \left| \sum_{\substack{h^1, h^2 \in \mathbb{Z}^t_{n+1} \\ (h^1, h^2) \sim [k]}} \left\langle \Psi\left(h^1\right) \middle| P_1 \middle| \Psi\left(h^2\right) \right\rangle \right|. \tag{4.11}$$

The last expression give us a straightforward upper bound for $L(\pi_x(1))$:

$$\widetilde{L}(\pi_x(1)) = \sum_k \sum_h |\langle \Psi(k)| P_z |\Psi(h)\rangle|. \tag{4.12}$$

These equations come from the state decomposition in Definition 7 (on page 18). Theorem 9 (on page 19) states that each quantum algorithm has a state decomposition, thus next theorem shows relations between metrics on such decomposition with the gap between quantum and classical query.

Theorem 21. *We use the same hypothesis of Theorem 20, denote $\#S$ as the cardinality of set S and define $d = \#\{k : |\Psi(k)\rangle \neq 0\}$. Then*

$$\frac{R_\varepsilon(f)}{t} \leq F_\varepsilon\left(\widetilde{L}(\pi_x(1))\right), \tag{4.13}$$

$$\frac{R_\varepsilon(f)}{t} \leq F_\varepsilon\left(\left(\sum_k \| |\Psi(k)\rangle \|\right)^2\right), \tag{4.14}$$

$$\frac{R_\varepsilon(f)}{t} \leq F_\varepsilon(d), \tag{4.15}$$

and

$$\frac{R_\varepsilon(f)}{t} \leq F_\varepsilon\left(\frac{1}{\min_k \langle\Psi(k)|\ \Psi(k)\rangle}\right). \tag{4.16}$$

Proof. Considering that F_ε is an increasing function, then Eq. (4.13) follows from Eq. (4.12) and Theorem 20. Eq. (4.14) comes from Eq. (4.12) by

$$|\langle\Psi(k)| P_z |\Psi(h)\rangle| \leq \| |\Psi(k)\rangle \| \, \| |\Psi(h)\rangle \|, \tag{4.17}$$

that implies

$$L(\pi_x(1)) \leq \left(\sum_k \| |\Psi(k)\rangle \|\right)^2. \tag{4.18}$$

If we apply Theorem 9 (on page 19), we have

$$\langle\Psi|\ \Psi\rangle = \sum_{k,h} \langle\Psi(k)|\ \Psi(h)\rangle = 1, \tag{4.19}$$

that is combined with $\sum_k \| |\Psi(k)\rangle \| \leq \sqrt{d}\sum_k \langle\Psi(k)|\ \Psi(k)\rangle$ and Eq. (4.18), to give

$$L(\pi_x(1)) \leq d. \tag{4.20}$$

Eq. (4.16) comes from $d\left(\min_k \langle\Psi(k)|\ \Psi(k)\rangle\right) \leq 1$.

□

Chapter 5

Conclusion

We developed a new approach for quantum query and applied such tool to the study of exact quantum algorithms and quantum speed-up. Exact quantum algorithms frameworks have few developments in comparison to bounded error algorithms and it is more challenging. Understanding what makes quantum computing faster than classical computing is an important objective in computer science, however previous research did not cover this issue in detail for quantum query. As we have two main results, we divide the conclusion in two parts.

5.1 Results and potential extensions of the Block Set approach

In this part of the work, we obtained theoretical results. The main theoretical result was a reformulation of the Quantum Query Model, where unitary operators are replaced by sign inversions applied over a set of vectors. This model proposes a different interpretation of how quantum query computes a function. Another result is the application of this model to designing exact quantum algorithms [10], which developed a linear system that allows the analysis of a family of quantum exact algorithms. Finally, we developed a lower bound method for exact quantum algorithms, thus it gives a validation for our model. This approach leaves open questions and possible developments:

- Most of complexity tools were developed using the QQM formulation ([23],[27]). If we start from other formulations like BSF, can we obtain better relations between complexity measures (D, R_2, Q_E, Q_2)?

- It is possible to relax the precision using the BSF tools, for example by obtaining approximate solutions to a system $E(t, n, X, Y)$, but such thing does not guarantee a bounded error. The system in Theorem 15 (on page 32) can

be extended, by finding sufficient and/or necessary conditions for a bounded error algorithm?

- The condition ($k_1 \neq k_2 \Rightarrow w_{k_1 k_2} = 0$) imposed in Corollary 3 (on page 33) could be weakened, the goal is obtaining powerful yet complicated models than the orthogonal BSF framework. Can we develop tools for constructing exact quantum algorithms under some stronger BSF case?

5.2 Results and potential extensions of the L_1-norm approach

We proved a necessary condition for the hardness on the classical simulation of quantum query algorithm. This condition is stated as a high L_1-norm defined from the output probability. An important feature about L_1-norm is its dependency on both evolution and measurement steps. A broadly studied property like quantum entanglement is limited just on the quantum states, that implies that an inadequate measurement step can nullify the computational power obtained in the evolution stage. However, the tightness of L_1-norm for estimating quantum speed-up depends on a simulation and the relation between such simulation and the optimal classical simulation is unknown.

As it was proved in Section 3.1 (on page 18), a state decomposition can be an alternative formulation for quantum query algorithms, such decomposition is stated as a set of vectors related to a given algorithm. The BSF suggests the importance of quantum parallelism, because each pair of vectors in the Block Set controls a function in the Boolean cube, finally the sum of all those parametrized functions produces an output probability function. The L_1-norm is related to the Block Set. That implies that (a) several non-zero vectors in the Block Set ($\#\left(k : |\Psi\left(k\right)\rangle \neq 0\right)$), and (b) low values for the minimum product between such vectors $\left(\min_k \langle\Psi\left(k\right)|\ \Psi\left(k\right)\rangle\right)$ must be present for a hard classical simulation. We can relate both measures to quantum parallelism. A low value for $\#\left\{k : |\Psi\left(k\right)\rangle \neq 0\right\}$ implies less combinations of vectors that add functions to the output probability function. Big values for $\min_k \langle\Psi\left(k\right)|\ \Psi\left(k\right)\rangle$ produces lower values for $\#\left\{k : |\Psi\left(k\right)\rangle \neq 0\right\}$ and it gives to the output probability function a shape closer to the basis functions of the Boolean cube. Remember that basis functions of the Boolean cube can be efficiently simulated by classical means.

We leave some open questions:

- The asymptotic relation between $\frac{R_\varepsilon(f)}{t}$, L_1-norm and the other measures is unknown, where t represents the number of steps for a quantum algorithm computing f within error ε.

- How close is the relation between L_1-norm and quantum entanglement?
- The quantum query model can formulate algorithms such as quantum-walk-based searches on graphs, thus those algorithms are influenced by L_1-norm. Nevertheless, that kind of algorithms are designed in other *ad hoc* frameworks, thus the L_1-norm condition requires an adaptation to such models.

Bibliography

[1] MOORE, G. E. "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff." *IEEE Solid-State Circuits Newsletter*, v. 3, n. 20, pp. 33–35, 2006.

[2] DUBASH, M. "Moore s Law is dead, says Gordon Moore", *Techworld (April 2005)*, 2005.

[3] FEYNMAN, R. P. "Simulating physics with computers", *International journal of theoretical physics*, v. 21, n. 6, pp. 467–488, 1982.

[4] SHOR, P. W. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pp. 124–134. IEEE, 1994.

[5] GROVER, L. K. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219. ACM, 1996.

[6] BERNSTEIN, E., VAZIRANI, U. "Quantum complexity theory", *SIAM Journal on Computing*, v. 26, n. 5, pp. 1411–1473, 1997.

[7] KAYE, P., LAFLAMME, R., MOSCA, M. *An introduction to quantum computing*. Oxford University Press, 2007.

[8] BUHRMAN, H., DE WOLF, R. "Complexity measures and decision tree complexity: a survey", *Theoretical Computer Science*, v. 288, n. 1, pp. 21–43, 2002.

[9] JORDAN, S. "Quantum algorithm zoo". http://math.nist.gov/quantum/zoo/, 2015.

[10] AMBAINIS, A. "Superlinear advantage for exact quantum algorithms", *SIAM Journal on Computing*, v. 45, n. 2, pp. 617–631, 2016.

[11] DEUTSCH, D., JOZSA, R. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London A: Mathematical,*

Physical and Engineering Sciences, v. 439, pp. 553–558. The Royal Society, 1992.

[12] MONTANARO, A., JOZSA, R., MITCHISON, G. "On exact quantum query complexity", *Algorithmica*, v. 71, n. 4, pp. 775–796, 2015.

[13] BARNUM, H., SAKS, M., SZEGEDY, M. "Quantum query complexity and semi-definite programming". In: *Computational Complexity, 2003. Proceedings. 18th IEEE Annual Conference on*, pp. 179–193. IEEE, 2003.

[14] AMBAINIS, A., IRAIDS, J., SMOTROVS, J. "Exact quantum query complexity of EXACT and THRESHOLD", *arXiv preprint arXiv:1302.1235*, 2013.

[15] NISAN, N., SZEGEDY, M. "On the degree of Boolean functions as real polynomials", *Computational complexity*, v. 4, n. 4, pp. 301–313, 1994.

[16] COOK, S., DWORK, C., REISCHUK, R. "Upper and lower time bounds for parallel random access machines without simultaneous writes", *SIAM Journal on Computing*, v. 15, n. 1, pp. 87–97, 1986.

[17] LEE, T., MITTAL, R., REICHARDT, B. W., et al. "Quantum query complexity of state conversion". In: *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pp. 344–353. IEEE, 2011.

[18] REICHARDT, B. W., SPALEK, R. "Span-program-based quantum algorithm for evaluating formulas". In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 103–112. ACM, 2008.

[19] BELOVS, A. "Span programs for functions with constant-sized 1-certificates". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pp. 77–84. ACM, 2012.

[20] NIELSEN, M. A., CHUANG, I. L. *Quantum computation and quantum information*. Cambridge university press, 2010.

[21] AMBAINIS, A., GRUSKA, J., ZHENG, S. "Exact query complexity of some special classes of Boolean functions", *arXiv preprint arXiv:1404.1684*, 2014.

[22] BENNETT, C. H., BERNSTEIN, E., BRASSARD, G., et al. "Strengths and weaknesses of quantum computing", *SIAM journal on Computing*, v. 26, n. 5, pp. 1510–1523, 1997.

[23] BEALS, R., BUHRMAN, H., CLEVE, R., et al. "Quantum lower bounds by polynomials", *Journal of the ACM (JACM)*, v. 48, n. 4, pp. 778–797, 2001.

[24] AARONSON, S., SHI, Y. "Quantum lower bounds for the collision and the element distinctness problems", *Journal of the ACM (JACM)*, v. 51, n. 4, pp. 595–605, 2004.

[25] AARONSON, S. "Limitations of quantum advice and one-way communication". In: *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pp. 320–332. IEEE, 2004.

[26] KLAUCK, H., ŠPALEK, R., DE WOLF, R. "Quantum and classical strong direct product theorems and optimal time-space tradeoffs", *SIAM Journal on Computing*, v. 36, n. 5, pp. 1472–1493, 2007.

[27] AMBAINIS, A. "Quantum lower bounds by quantum arguments". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pp. 636–643. ACM, 2000.

[28] ZHANG, S. "On the power of Ambainis lower bounds", *Theoretical Computer Science*, v. 339, n. 2, pp. 241–256, 2005.

[29] AMBAINIS, A. "Polynomial degree vs. quantum query complexity". In: *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pp. 230–239. IEEE, 2003.

[30] LAPLANTE, S., MAGNIEZ, F. "Lower bounds for randomized and quantum query complexity using Kolmogorov arguments". In: *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pp. 294–304. IEEE, 2004.

[31] ŠPALEK, R., SZEGEDY, M. "All quantum adversary methods are equivalent". In: *International Colloquium on Automata, Languages, and Programming*, pp. 1299–1311. Springer, 2005.

[32] HOYER, P., LEE, T., SPALEK, R. "Negative weights make adversaries stronger". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 526–535. ACM, 2007.

[33] SPALEK, R. "The multiplicative quantum adversary". In: *Proc. 23rd IEEE Complexity*.

[34] AMBAINIS, A., IRAIDS, J., NAGAJ, D. "Exact quantum query complexity of $EXACT_{k,l}^n$". 2016.

[35] DE WOLF, R. “A Brief Introduction to Fourier Analysis on the Boolean Cube”. Theory of Computing, Graduate Surveys, 2008.

[36] AARONSON, S., A. A. “The Need for Structure in Quantum Speedups”. arXiv preprint arXiv:0911.0996, 2009.

[37] ABBOTT, A. A., CALUDE, C. S. “Understanding the quantum computational speed-up via de-quantisation”, *arXiv preprint arXiv:1006.1419*, 2010.

[38] DEUTSCH, D. “Quantum theory, the Church-Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, v. 400, pp. 97–117. The Royal Society, 1985.

[39] JOZSA, R. “Entanglement and quantum computation”. arXiv preprint quant-ph/9707034, 1997.

[40] JOZSA, R., LINDEN, N. “On the role of entanglement in quantum-computational speed-up”, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*.

[41] DATTA, A., SHAJI, A., CAVES, C. M. “Quantum discord and the power of one qubit”, *Physical Review Letters*.

[42] BELL, J. S. “On the problem of hidden variables in quantum mechanics”, *Reviews of Modern Physics*.

[43] KOCHEN, S., SPECKER, E. P. “The problem of hidden variables in quantum mechanics”, *Reviews of Modern Physics*.

[44] HOWARD, M., WALLMAN, J., VEITCH, V., et al. “Contextuality supplies the magic for quantum computation”, *Nature*.

[45] ANGLUIN, D., VALIANT, L. “Fast probabilistic algorithms for Hamiltonian circuits and matchings”, *Journal of Computer and System Sciences*.

[46] HARDY, L. “Quantum theory from five reasonable axioms”, *arXiv preprint quant-ph/0101012*, 2001.

[47] DE WOLF, R. “Quantum Computing: Lecture Notes”. http://homepages.cwi.nl/ rdewolf/qcnotes.pdf, 2016.

[48] AARONSON, S. “PHYS771 Quantum Computing Since Democritus”. http://www.scottaaronson.com/democritus/, 2006.

Appendix A

Preliminary notions

In this appendix, we present necessary notions in mathematics and quantum mechanics. Section A.1 is intended to help the introduction of quantum algorithms. Section A.2 presents a mathematical concept that helps as an alternative to the formulation presented in Subsection A.1.4. Section A.3 presents a consequence of quantum measurement. Finally Section A.4 introduces Boolean functions that are necessary for understanding the state of art for exact quantum query algorithms.

A.1 Postulates of quantum mechanics

In this section we remark the mathematical formulation of quantum mechanics. The postulates work for both finite or infinite Hilbert spaces, but for our purposes we consider that the finite case is enough. These notions may be helpful for Chapter 2. Other helpful references are an article by Lucien Hardy [46], or lecture notes by Ronald de Wolf [47] and Scott Aaronson [48].

A.1.1 State postulate

The state of a quantum system is formulated as a unit vector from a Hilbert space $\mathcal{H}$.

For example, we have a qubit defined over a Hilbert Space $\mathcal{H}$ with basis $\{|0\rangle, |1\rangle\}$, where $|0\rangle$ and $|1\rangle$ represent classical states where a bit has value 0 and 1, respectively. Thus, the state of the system in $\mathcal{H}$ is described as $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$ and $\alpha_0, \alpha_1 \in \mathbb{C}$.

A.1.2 Evolution postulate

Given a closed quantum system, its evolution on time is described by a unitary operator.

In other words, if $|\Psi_1\rangle$ describes the state of a isolated quantum system and $|\Psi_2\rangle$ describes the state of the same system in other time, then there exists a unitary U operator such that $|\Psi_2\rangle = U|\Psi_1\rangle$.

A.1.3 Composition of systems postulate

We can treat two physical systems as one combined system. In that case, the state space of the combined quantum systems is the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ between the spaces $\mathcal{H}_1, \mathcal{H}_2$ corresponding to each subsystem. If the first system is in state $|\Psi_1\rangle$ and second system is in state $|\Psi_2\rangle$, then the combined state of both systems is described by $|\Psi_1\rangle \otimes |\Psi_2\rangle$.

This work only considers finite spaces, thus for our purposes the tensor product can be considered a Kronecker product. For example, if we consider two qubits, then we have the spaces $\mathcal{H}_1, \mathcal{H}_2$ whose basis are $\{|0\rangle, |1\rangle\}$. Thus, space of the composite system is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, where $|i\rangle \otimes |j\rangle = |ij\rangle$. If the first qubit is in state $|\Psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the second qubit is in state $|\Psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, then the composition state is $|\Psi_3\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$.

Not all states of the composite system can be represented as tensor products of the states from the subsystems, for example two qubits in state $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$. In that case we say that such subsystems are entangled and we cannot describe the state of one subsystem without the others. This does not violate the first postulate, because entanglement implies that each system is not a closed system anymore.

A.1.4 Measurement postulate

Let $B = \{|\varphi_i\rangle\}$ be a orthogonal basis for the state space $\mathcal{H}$ of the system. We can perform a measurement in relation to basis B over state $\sum_i \alpha_i |\varphi_i\rangle$ such that it outputs label i with probability α_i^2 and leaves system in state $|\varphi_i\rangle$.

For example, take the state $|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. If we chose basis $B_1 = \{|0\rangle, |1\rangle\}$, we will get both states $|0\rangle$ and $|1\rangle$ with probability $\frac{1}{2}$. If we chose basis $B_2 = \left\{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right\}$, we will get states $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ with probability 0 and 1, respectively.

A.2 Complete set of orthogonal projectors

Let H be a finite Hilbert space and T a finite set. Two operators A and B are orthogonal if $\langle\Psi| A^\dagger B |\Phi\rangle = 0$ for all $|\Phi\rangle$,$|\Psi\rangle \in H$. A complete set of orthogonal projectors (CSOP) is a set $\{P_z : z \in T\}$. Where each element P_z is a projector,

orthogonal to the others and satisfy

$$\sum_{z \in T} P_z = I_H, \tag{A.1}$$

such that I_H as the identity operator on H. The concept of CSOP is applied in an alternative definition of quantum measurement. The chosen formulation for quantum query takes such measurement definition (see Section 2.1).

A.3 State guessing bound

Suppose that we have two given quantum states $|\Psi_X\rangle$ and $|\Psi_Y\rangle$. Where $|\langle\Psi_X|\Psi_Y\rangle| = \delta$. Consider a procedure for determining if $Z = X$ or $Z = Y$, for an input $|\Psi_Z\rangle$. Then, this procedure will guess correctly at most with probability $\frac{1}{2}+\frac{1}{2}\sqrt{1-\delta^2}$. Reaching such probability requires an optimal measurement or CSOP.

This bound implies that we can separate two quantum states without error if and only if both states are orthogonal (see Subsection 3.1.2 and Section 3.2).

A.4 Important Boolean functions

Let $|x|$ be the Hamming weight of $x \in \{0,1\}^n$. Then we define the next total Boolean functions:

- $OR_n(x) = 1 \iff |x| = 1$.
- $AND_n(x) = 1 \iff |x| = n$.
- $PARITY_n(x) = 1 \iff |x|$ is odd.
- $MAJ_n(x) = 1 \iff |x| > \frac{n}{2}$.
- $EXACT_{k,n}(x) = 1 \iff |x| = k$.
- $THRESHOLD_{k,n}(x) \iff |x| \geq k$.
- Let $NE : \{0,1\}^3 \to \{0,1\}$ be a function where: $NE(x_1,x_2,x_3) = 1$ if $x_i \neq x_j$ for some $i,j \in \{1,2,3\}$, and $NE(x_1,x_2,x_3) = 0$, if $x_1 = x_2 = x_3$. There is the recursive sequence of functions $\{NE^i\}$, where $NE^0(x) = x$ and for $d > 0$

$$NE^d(x_1,..,x_{3^d}) =$$

$$NE\left(NE^{d-1}(x_1,..,x_{3^{d-1}}), NE^{d-1}(x_{3^{d-1}+1},..,x_{2.3^{d-1}}), NE^{d-1}(x_{2.3^{d-1}+1},..,x_{3^d})\right)$$

- $DJ_n^k(x) = \begin{cases} 1 & if\ |x| = \frac{n}{2} \\ 0 & if\ |x| \leq k\, or\ |x| \geq n-k \end{cases}$
  where $0 \leq k < \frac{n}{2}$ and n is even.
- $EXACT_{k,l}^n(x) = 1 \iff |x| \in \{k, l\}$.

These functions are mentioned in Chapter 2, where we review the state of art in quantum algorithms and lower bound methods.