Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FINAL DE CARRERA

QUANTUM APPS:

Simulació dels algorismes de Shor i Grover

Estudis: Enginyeria de Telecomunicació
Autor: David Arcos Gutiérrez
Director: Lluís Ametller Congost
Ponent: Núria Ferrer Anglada
Data: 5 de juliol de 2013

ÍNDEX

Índex de figures **7**

Índex de taules **8**

Agraïments **11**

Resum **12**
Resum del projecte . . . 12
Resumen del proyecto . . . 12
Abstract . . . 12

Notació i format **13**

1 INTRODUCCIÓ **15**
1.1 Motivació i context general . . . 15
1.2 Estructura del projecte . . . 16
1.3 Objectius . . . 16
1.3.1 Objectius principals . . . 16
1.3.2 Objectius secundaris . . . 17

2 BLOC TEÒRIC I: FÍSICA QUÀNTICA **19**
2.1 Fonaments físics . . . 19
2.2 Hipòtesi de Planck . . . 20
2.3 Efecte fotoelèctric . . . 21
2.4 Dualitat ona-partícula . . . 22
2.5 Ones de De Broglie . . . 24
2.6 Principi d'indeterminació de Heisenberg . . . 25
2.7 Equació de Schrödinger . . . 26
2.8 Els postulats de la mecànica quàntica . . . 27

3 BLOC TEÒRIC II: COMPUTACIÓ QUÀNTICA **29**
3.1 Notació de Dirac . . . 29
3.2 Qubits, qutrits i qudits . . . 30
3.3 Esfera de Bloch . . . 31
3.4 Sistemes amb n qubits . . . 32
3.5 Mesura . . . 33
3.6 Portes quàntiques d'un qubit . . . 33
3.7 Portes quàntiques de n qubits . . . 36
3.8 Avaluació de funcions quàntiques . . . 37
3.9 Algorisme de Deutsch-Jozsa . . . 37
3.10 Algorisme de Shor . . . 39
3.10.1 Descripció . . . 39
3.10.2 L'algorisme pas a pas . . . 40
3.11 Algorisme de Grover . . . 45
3.11.1 Descripció . . . 45
3.11.2 Iteració de Grover . . . 46

4 BLOC TEÒRIC III: MODELS MATEMÀTICS **49**
4.1 Introducció . . . 49
4.2 Aritmètica modular, ordres i phi d'Euler . . . 49
4.3 Algorisme de fraccions contínues . . . 53
4.4 Criptografia basada en RSA . . . 54
4.5 Teoria de grafs . . . 55

5 BLOC TEÒRIC IV: COMPUTACIÓ ACUMULATIVA **61**
5.1 Funcions acumulatives . . . 61
5.2 *Swap* **acumulatiu de dos qubits** . . . 62
5.3 *Swap* **acumulatiu de tres qubits** . . . 63
5.4 *Swap* **acumulatiu general** . . . 66

6 BLOC PRÀCTIC I: OBJECTIVE-C I XCODE **69**
6.1 Introducció . . . 69
6.2 Estructura del llenguatge Objective-C . . . 70
6.3 Llibreries i recursos externs . . . 71

7 BLOC PRÀCTIC II: QGT 01 **73**
7.1 Introducció . . . 73
7.2 Descripció . . . 73
7.3 Estructura . . . 75
7.4 Implementació . . . 77
7.5 Presentació . . . 79

8 BLOC PRÀCTIC III: QGT 02 **81**
8.1 Introducció . . . 81
8.2 Descripció . . . 81
8.3 Estructura . . . 82
8.4 Implementació . . . 84
8.5 Presentació . . . 85

9 BLOC PRÀCTIC IV: QGT 03 **87**
9.1 Introducció . . . 87
9.2 Descripció . . . 87
9.3 Estructura . . . 90
9.4 Implementació . . . 91
9.5 Presentació . . . 94
9.6 Simulacions . . . 95

10 BLOC PRÀCTIC V: QGT 04 **97**
10.1 Introducció . . . 97
10.2 Descripció . . . 98
10.3 Estructura . . . 101
10.4 Implementació . . . 102
10.5 Presentació . . . 109
10.6 Simulacions . . . 110

11 SÍNTESI I CONCLUSIONS **117**
11.1 Conclusions . . . 117
11.2 Continuació . . . 118

Bibliografia i referències **119**

ÍNDEX DE FIGURES

2.1 Cinquè congrés Solvay 20
2.2 Experiment de Millikan 22
2.3 Experiment de Young 23
2.4 Pou infinit 26

3.1 Esfera de Bloch 32
3.2 Portes lògiques d'un bit 34
3.3 Porta Hadamard 34
3.4 Porta fase S 34
3.5 Porta fase T 34
3.6 Porta *Phase Shift* 34
3.7 Portes de Pauli 35
3.8 Esquema d'una porta general 35
3.9 Porta Control-Not 36
3.10 Esquema d'una funció quàntica 37
3.11 Circuit general de Deutsch 38
3.12 Circuit de Shor 40
3.13 Distribució de probabilitat en $|x\rangle$ després de la QFT 43
3.14 Circuit general de Grover 46
3.15 Bloc G de Grover 46
3.16 Iteració de Grover 47

4.1 Graf d'ordre 4 56

5.1 Funció acumulativa 62

7.1 Pantalla principal del QGT 01 74
7.2 Submenús del QGT 01 75
7.3 Esquema dels objectes del QGT 01 76
7.4 Objecte *storyboard* del QGT 01 77
7.5 Informació de la versió penjada del QGT 01 79

7.6 Detalls de la versió penjada del QGT 01 80
7.7 Dades del QGT 01 a l'App Store 80

8.1 Vistes d'edició de grafs del QGT 02 82
8.2 Esquema dels objectes del QGT 02 83
8.3 Informació de la versió penjada del QGT 02 85
8.4 Detalls de la versió penjada del QGT 02 86
8.5 Dades del QGT 02 a l'App Store 86

9.1 Pantalla inicial del QGT 03 88
9.2 Pantalla del circuit del QGT 03 88
9.3 Animació d'una iteració al QGT 03 89
9.4 Gràfic de probabilitat del registre al QGT 03 89
9.5 Esquema dels objectes del QGT 03 91
9.6 Informació de la versió penjada del QGT 03 94
9.7 Detalls de la versió penjada del QGT 03 94
9.8 Dades del QGT 03 a l'App Store 95

10.1 Vista *Intro* del QGT 04 98
10.2 Vista *Shor* del QGT 04 99
10.3 Algorisme de Shor en procés en el QGT 04 99
10.4 Resultat de l'algorisme en el QGT 04 100
10.5 Vista *Arithmetic* del QGT 04 100
10.6 Esquema dels objectes del QGT 04 102
10.7 Informació de la versió penjada del QGT 04 109
10.8 Detalls de la versió penjada del QGT 04 109
10.9 Dades del QGT 04 a l'App Store 110

ÍNDEX DE TAULES

4.1 Ordre dels elements de $(\mathbb{Z}/N)^*$ 51
4.2 Ordre dels elements de $(\mathbb{Z}/P^v)^*$ 52

5.1 *Swap* acumulatiu de 2 qubits 63
5.2 *Swap* acumulatiu de 3 qubits 64
5.3 Combinacions pel *swap* acumulatiu de 4 qubits 67

7.1 Llista de codis del QGT 01 78

9.1 Resum de les simulacions del QGT 03 95

10.1 Simulacions per factoritzar el 15 110
10.2 Simulacions per factoritzar el 18 110
10.3 Simulacions per factoritzar el 21 111
10.4 Simulacions per factoritzar el 24 111
10.5 Simulacions per factoritzar el 25 111
10.6 Simulacions per factoritzar el 27 111
10.7 Simulacions per factoritzar el 30 112
10.8 Simulacions per factoritzar el 33 112
10.9 Simulacions per factoritzar el 35 112
10.10 Simulacions per factoritzar el 39 112
10.11 Simulacions per factoritzar el 42 113
10.12 Simulacions per factoritzar el 45 113
10.13 Simulacions per factoritzar el 51 113
10.14 Simulacions per factoritzar el 55 113
10.15 Simulacions per factoritzar el 57 114
10.16 Simulacions per factoritzar el 63 114
10.17 Resum de les simulacions del QGT 04 114
10.18 Resum de les simulacions II del QGT 04 115

Agraïments

La idea inicial d'aquest projecte va sorgir a finals del 2011, però, a causa dels exàmens, no em vaig posar a treballar fins al 25 de gener de 2012. Recordo que aquella última setmana de gener, vaig treure els apunts de computació quàntica de l'armari i em vaig tornar a llegir pàgina per pàgina les meravelles de la física quàntica i, més concretament, el potencial de la computació quàntica.

Vull donar les gràcies, en primer lloc, a en Lluís Ametller per presentar-me aquesta nova branca d'estudi i, en certa manera, per retornar-me la il·lusió per aprendre. De fet, el comportament tan estrany del món microscòpic i els avenços de la física i la tecnologia actuals em creen una curiositat més pròpia d'un nen que descobreix el món que d'un futur enginyer. Aquesta curiositat m'ha ajudat en molts moments a tirar endavant el projecte i a continuar els meus estudis de matemàtiques. Pels seus coneixements i com a agraïment a la seva tasca docent, volia que fos el director del projecte.

També m'agradaria donar les gràcies al meu company i amic Daniel Gabriel, per la paciència, el suport i els consells. Probablement és la persona que més feina m'ha estalviat durant l'elaboració del projecte i més hores ha passat al meu costat per ajudar-me en tot el que ha pogut i més.

Pel que fa a l'àmbit formatiu, m'agradaria donar les gràcies a tots els grans professors que m'han rebut sempre de bon grat i que m'han donat el suport i la formació necessaris per completar aquest projecte. L'Antoni Benseny, en Miquel Escudero, la Núria Ferrer, en Francesc Marquès, l'Artur Travessa, en Lluís Ametller, en Narciso Román, la Beatriz Otero, en Marcel Fernández, en Miquel Soriano, en Ferran Marquès i en Javier Soria, entre d'altres, han estat bons professors, sempre disposats a resoldre els meus dubtes i a potenciar el meu interès per les matèries que imparteixen. Els coneixements que m'han proporcionat aquests professors m'han donat les eines per desenvolupar aquest projecte. Pel mateix motiu, vull donar les gràcies a en Carles Noguera, un company de matemàtiques, per la seva ajuda quan començava a programar per a iOS.

Encara que hi he tingut poc contacte, també m'agradaria agrair a Juan Ignacio Cirac la seva tasca en investigació i divulgació. Ha estat una inspiració constant i els seus articles i conferències m'han proporcionat una dosi de motivació extra durant els últims dos anys.

També m'agradaria agrair de manera especial el suport moral del meus familiars. D'una banda, els meus pares i la meva germana, Ivette, que han estat pendents del meu benestar i m'han sabut distreure quan calia. D'altra banda, la Mireia, que m'ha donat suport en tots els moments difícils, m'ha ajudat en la correcció lingüística de la memòria (i les aplicacions) i ha aguantat estoicament totes les meves explicacions sobre computació quàntica i computació acumulativa.

Finalment, agrair a tots aquells que heu cregut en mi i m'heu animat a seguir endavant en el procés d'aprenentatge i maduresa personal. Gràcies a tots.

David

Resum del projecte

Una base tèorica sobre física quàntica, computació quàntica i models matemàtics constitueix el punt de partida per al desenvolupament de la col·lecció d'aplicacions *Quantum Graph Tool*. Es tracta de quatre aplicacions per al sistema operatiu iOS relacionades amb el món de la computació quàntica i la teoria de grafs. Més concretament, les dues aplicacions principals representen simuladors interactius dels algorismes quàntics de Grover —per a cerques en llistes ordenades— i de Shor —per a la factorització de nombres enters.

Aquestes aplicacions pretenen divulgar la computació quàntica mitjançant els dispositius mòbils (iPad) i, alhora, representen eines per estudiar noves maneres de computar basades en les propietats de la mecànica quàntica. Amb aquests mateixos objectius, a banda de les aplicacions, es planteja una introducció a la *computació acumulativa*, la qual pretén ser una nova manera de computar per resoldre problemes de teoria de grafs.

Resumen del proyecto

Una base teórica sobre física cuántica, computación cuántica y modelos matemáticos constituye el punto de partida para el desarrollo de la colección de aplicaciones *Quantum Graph Tool*. Se trata de cuatro aplicaciones para el sistema operativo iOS relacionadas con el mundo de la computación cuántica y la teoría de grafos. En concreto, las dos aplicaciones principales representan simuladores interactivos de los algoritmos de Grover —para búsquedas en listas ordenadas— y de Shor —para la factorización de números enteros.

Estas aplicaciones pretenden divulgar la computación cuántica mediante los dispositivos móviles (iPad) y, a su vez, representan herramientas para el estudio de nuevas maneras de computar basadas en las propiedades de la mecánica cuántica. Con la misma finalidad, además de las aplicaciones, se plantea una introducción a la *computación acumulativa*, la cual pretende ser una nueva manera de computar para resolver problemas de teoría de grafos.

Abstract

A theoretical base on quantum physics, quantum computing and mathematical models is the starting point for the development of the applications collection *Quantum Graph Tool*. It consists on four applications for the iOS operating system related to quantum computing and graph theory. To be more precise, the two main applications are interactive simulators of Grover's quantum algorithm (to search in ordered lists) and Shor's quantum algorithm (to factorize integer numbers).

These applications aim to divulge quantum computing using mobile devices (iPad) and, in addition, they are tools to study new computing methods based on quantum mechanics' properties. With the same objective, besides the applications, *cumulative computing* is introduced as a new computing method to solve problems in graph theory.

Notació i format

c	velocitat de la llum en el buit
h	constant de Planck
$\mathbb{Z}$	conjunt dels nombres enters
$\mathbb{R}$	conjunt dels nombres reals
$\mathbb{C}$	conjunt dels nombres complexos
i	$\sqrt{-1}$, unitat complexa
$\mathbb{I}$	matriu identitat, funció identitat o element identitat
$H^{\otimes n}$	producte tensorial de n portes Hadamard
$[x]$	part entera de x
$\#X$	cardinal del conjunt X, nombre d'elements de X
$\mathcal{P}(X)$	$\{Y \mid Y \subseteq X\}$, conjunt de les parts de X o conjunt potència de X
$U_{\mathbb{C}}(n)$	conjunt de les matrius quadrades unitàries de coeficients complexos

CAPÍTOL

1

INTRODUCCIÓ

1.1 Motivació i context general

La física quàntica fou un dels grans èxits de la física del segle XX. La seva naturalesa tan estranya i fascinant obre un nou món de possiblitats pels científics i els enginyers del segle XXI. La computació quàntica n'és un bon exemple, tant per la seva necessitat immediata —estimulada pel procés de miniaturització— com pel seu gran potencial —derivat de les propietats quàntiques de la matèria.

El nucli d'aquest projecte és el conjunt d'aplicacions *Quantum Graph Tool* (QGT), que representa una família d'Apps per al dispositiu iPad relacionades amb el món de la computació quàntica. La finalitat d'aquestes aplicacions és proporcionar eines d'estudi que permetin desenvolupar noves maneres de computar. En concret, els problemes oberts de la teoria de grafs es podrien resoldre utilitzant les propietats de paral·lelisme i superposició de la computació quàntica. En aquest text es presenten les quatre primeres aplicacions del projecte QGT. Les dues primeres, QGT 01 i QGT 02, corresponen respectivament a una introducció als qubits —la unitat d'informació de la computació quàntica— i als grafs. La tercera App és una simulació de l'algorisme quàntic de Grover, que implementa un buscador en una llista ordenada utilitzant les propietats de la mecànica quàntica. La quarta és també una simulació d'un algorisme quàntic molt recent que resol el problema de la factorització de nombres enters, l'algorisme de Shor.

Addicionalment, s'introdueix una nova manera de computar basada en l'acumulació d'informació que aprofita els fenòmens de la superposició i l'entrellaçament per resoldre els problemes de la teoria de grafs esmentats. Aquesta «computació acumulativa» m'obre una nova línia de recerca personal per continuar els estudis en aquesta disciplina que «entrellaça» matemàtica, física i computació.

1.2 Estructura del projecte

En primer lloc, cal remarcar que bona part de la memòria del projecte és una descripció teòrica. D'una banda, les aplicacions del projecte requereixen una base conceptual sobre la computació quàntica, els algorismes de Shor i de Grover i la teoria de grafs. De l'altra, la computació acumulativa és una estructura matemàtica basada en la computació quàntica que només es pot plantejar com un desenvolupament teòric. Tota aquesta teoria s'ha dividit en quatre blocs que es presenten en els capítols 2, 3, 4 i 5.

En el primer bloc teòric es presenta la física quàntica i algunes de les propietats del món microscòpic. Aquesta descripció és fonamental per entendre la computació quàntica que es presenta en el segon bloc. En aquest bloc, doncs, a més de la computació quàntica, es descriuen els algorismes de Shor i Grover, que corresponen al contingut de dues de les aplicacions. El tercer bloc recull la descripció dels grafs i alguns conceptes matemàtics addicionals, que són importants per complementar la descripció dels algorismes mencionats. El quart bloc està dedicat exclusivament a la idea de la computació acumulativa plantejada per l'autor. Aquest darrer bloc pot semblar una mica aïllat de la resta, però constitueix el fil conductor de les aplicacions *Quantum Graph Tool*, ja que el seu objectiu principal és desenvolupar una nova manera de computar.

Després de la teoria, es presenten cinc blocs pràctics en els quals es descriuen l'entorn de desenvolupament (capítol 6) i cada una de les aplicacions (capítols 7, 8, 9 i 10). Com que els conceptes teòrics necessaris per entendre les aplicacions estan repartits en els blocs teòrics II i III, es recomana al lector rellegir els apartats corresponents si sorgeixen dificultats per seguir algun punt del codi descrit (especialment, pels algorismes de Shor i Grover). Per no allargar el text innecessàriament, s'ha obviat la major part del codi de les aplicacions —de fet, tot el que no té a veure amb els algorismes quàntics. El lector interessat es pot dirigir a l'autor[1] si té qualsevol dubte relacionat amb les aplicacions.

En resum, l'estructura del projecte es pot dividir en dos grans blocs: teoria (física, computació i matemàtica) i pràctica (programació de les aplicacions per a iPad).

1.3 Objectius

1.3.1 Objectius principals

El primer objectiu d'aquest projecte és dominar un nou llenguatge i entorn de programació. En concret, el llenguatge Objective-C i l'entorn Xcode, els quals permeten programar aplicacions per al sistema opertiu iOS, d'Apple. Això implica aprendre les particularitats de la programació d'aplicacions per a dispositius mòbils i aprofitar els coneixements de programació adquirits al llarg de la carrera per tal de desenvolupar aplicacions completes i funcionals. Es tracta d'un reciclatge de coneixements i de l'exploració d'una nova tecnologia, de manera que representa un entrenament pel dia a dia d'un enginyer de la informació.

[1] Correu de contacte: *dave.luminosity@gmail.com*.

D'altra banda, es pretén programar simuladors de dos dels algorismes quàntics més rellevants per tal de comprendre més profundament com funcionen i proporcionar eines d'estudi de la computació quàntica. Els algorismes escollits són els de Shor (QGT 04) i de Grover (QGT 03), ja que representen una millora significativa respecte dels algorismes de computació clàssica.

1.3.2 Objectius secundaris

A banda dels simuladors, també es volen programar altres aplicacions que serveixin per introduir la teoria de grafs i la computació quàntica. Concretament, una aplicació per familiaritzar-se amb la unitat d'informació quàntica (QGT 01) i una altra per presentar sistemes de codificació de grafs (QGT 02).

Un altre objectiu secundari és ordenar els coneixements teòrics sobre física quàntica, computació quàntica i matemàtiques amb la intenció de preparar el terreny per a futures investigacions i treballs i, si és possible, per desenvolupar noves aplicacions sobre computació quàntica i teoria de grafs.

Finalment, com es comenta en la introducció, es vol plantejar una nova manera de computar basada en l'acumulació d'informació. Es pretén resoldre problemes d'anàlisi en arbre (exponencials) mitjançant les propietats d'entrellaçament i superposició pròpies de la mecància quàntica. Aquesta nova computació ha de ser el punt de partida per continuar estudiant possibles algorismes quàntics.

CAPÍTOL

2

BLOC TEÒRIC I: FÍSICA QUÀNTICA

2.1 Fonaments físics

Durant els últims anys, la teoria i tecnologia dels semiconductors ha fet possible l'existència i l'evolució constant de la tecnologia CMOS basada en el silici. El 1965 Gordon E. Moore va formular la seva famosa llei, la qual predeia que cada divuit mesos es duplicaria el nombre de transistors presents en un circuit imprès, que s'ha anat confirmant durant més de 40 anys. No obstant això, el mateix Moore va afirmar que la llei deixaria de ser vàlida al cap de poc temps i que s'hauria de substituir el silici per una tecnologia nova. Certament, els transistors actuals estan formats per pocs àtoms i es preveu que el 2020 els transistors estiguin compostos per un únic àtom.[1] En aquestes escales atòmiques, les lleis de la física clàssica no es compleixen. S'ha de recórrer a la teoria quàntica.

La teoria quàntica és una branca de la física que explica els fenòmens d'interacció, a escala atòmica, de les partícules més fonamentals. Aquesta teoria va néixer el 1900 a partir de la hipòtesi de Planck —que considera que l'energia està quantitzada— i s'ha anat completant amb, entre d'altres, l'explicació d'Einstein de l'efecte fotoelèctric, les relacions de De Broglie, el principi d'incertesa de Heisenberg i la famosa equació de Schrödinger. Totes aquestes teories i hipòtesis han estat comprovades empíricament i demostren que la comprensió que l'ésser humà té del món que l'envolta ha fet un gran salt en els últims cent anys.

A més, estudiar i entendre les propietats corpusculars de la matèria i les lleis físiques que regeixen el món microscòpic és essencial per comprendre la unitat d'informació en la computació quàntica: el qubit.[2]

[1] Actualment ja s'ha aconseguit fabricar transistors amb un sol àtom.[15]
[2] El nom del qual prové de q-bit (quantum bit).

El qubit, com es veu en el capítol 3, substituiex el concepte determinista de bit de la computació clàssica. Més endavant, es posa de manifest que la computació quàntica ja no és un problema purament matemàtic de zeros i uns. El futur ordinador quàntic serà un objecte físic constituït per uns pocs àtoms o partícules elementals i, com a tal, seguirà les lleis de la física quàntica.

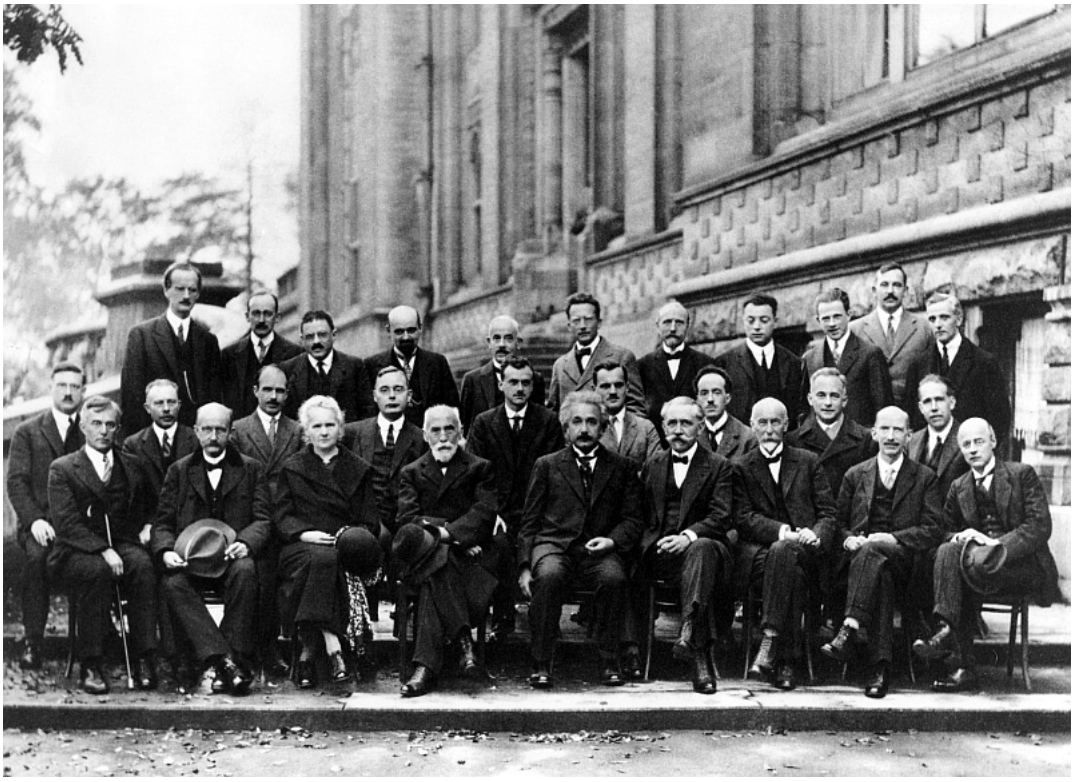

Figura 2.1: Els pares de la física quàntica durant el cinquè congrés Solvay el 1927.

A continuació, per introduir la computació quàntica, es fa un breu recorregut per alguns del resultats més importants de la revolució quàntica del segle XX iniciada per Max Planck.

2.2 Hipòtesi de Planck

«Era un acte de desesperació, havia de trobar
una interpretació teòrica de la fórmula de la distribució espectral.»

A finals del segle XIX, les observacions empíriques de l'espectre de radiació d'un cos negre[3] no tenien cap justificació teòrica. Kirchhoff ja havia demostrat l'existència d'una funció universal, que havia de descriure la intensitat de radiació d'un cos negre perfecte, que seria independent dels constituents del cos.

[3] Per *cos negre* s'entén qualsevol cos que absorbeix tota la radiació externa de manera que se'n pot observar l'espectre d'emissió. [5]

Wilhelm Wien havia proposat una llei que s'aproximava al resultat empíric per a altes freqüències, la *llei de desplaçament de Wien*. D'altra banda, la *llei de Rayleigh-Jeans* s'ajustava correctament a baixes freqüències, però presentava la denominada *catàstrofe ultravioleta* a altes freqüències. Max Planck, tal com havia fet Rayleigh, va partir d'un model basat en una cavitat que absorbia i radiava ones electromagnètiques mitjançant les lleis de Maxwell. A partir d'aquest model, va formular la *llei de Plank* (2.1) que s'ajustava correctament a tot l'espectre radioelèctric.

$$u_v = \frac{8\pi h}{c^3} \frac{\nu^3}{e^{h\nu/kT} - 1} \tag{2.1}$$

On u_v correspon a la radiància per a una freqüència ν donada; c, a la velocitat de la llum en el buit; T, a la temperatura absoluta; k, a la constant de Boltzman, i h, a la *constant de Planck* ($h = 6,62606957 \cdot 10^{-34}$ J $\cdot$ s).

A més, gràcies als grans coneixements de termodinàmica que posseïa, Planck va aconseguir trobar una justificació per a la seva llei. L'únic problema era que aquesta justificació suposava que l'energia dels oscil·ladors era discreta. Aquesta suposició es coneix amb el nom d'*hipòtesi de Planck*, la qual, per garantir la llei de desplaçament de Wien, relaciona l'energia E i la freqüència ν mitjançant la constant de Planck.

$$E = h\,\nu \tag{2.2}$$

Aquesta hipòtesi postula que l'energia està quantitzada, és la que dóna nom a la branca i va ser la precursora de la resta de teories que es presenten en aquest capítol. A finals de l'any 1900 va néixer, doncs, la física quàntica.

2.3 Efecte fotoelèctric

El 1905, cinc anys després de la presentació de la hipòtesi de Planck, Albert Einstein va publicar quatre articles de gran trascendència: sobre el moviment brownià, la relativitat especial, l'equivalència entre massa i energia i l'explicació de l'efecte fotoelèctric. Aquest últim es basava en el treball de Planck sobre la quantització de l'energia introduint-hi els fotons: els paquets d'energia radiada. Aquest treball teòric va ser comprovat empíricament per R. A. Millikan el 1916. Ambdós físics van rebre el premi Nobel per la seva aportació a la ciència el 1921 (Einstein) i el 1923 (Millikan).

L'efecte fotoelèctric verifica la validesa de l'equació (2.2) de la manera següent: quan s'il·lumina la superfície d'un metall amb radiació electromagnètica es poden arrencar electrons de la superfície. Els electrons arrencats, anomenats *fotoelectrons*, poden desplaçar-se per un circuït elèctric formant un corrent mesurable. L'energia cinètica màxima dels fotoelectrons varia linealment amb la freqüència de la radiació incident, però no amb la seva intensitat.

Les mesures de Millikan (veure figura 2.2) van demostrar que una intensitat més alta de llum fa que es generin més fotoelectrons, però la seva energia cinètica màxima no es fa més gran. A més, existeix una freqüència llindar, ν_0, que depèn del metall, a partir de

la qual es comença a produir aquest efecte. El potencial associat a aquesta freqüència mínima s'anomena *funció treball* del metall W (2.3).

$$W = h\,\nu_0 \tag{2.3}$$

L'explicació que Einstein va donar d'aquest comportament és que el metall absorbeix la radiació en quantitats discretes, quanta[4] de llum, i no d'una manera contínua com passaria en cas que l'efecte fos provocat pel camp elèctric d'una ona.[5] Així, l'energia cinètica màxima, E_{max}, que presenten els fotoelectrons es pot descriure com:

$$E_{max} = h\,\nu - W \tag{2.4}$$

On ν és la freqüència de la llum incident.

A més de l'efecte fotoelèctric, hi ha altres fenòmens que demostren la naturalesa corpuscular de la llum, per exemple l'efecte Compton. A més, actualment hi ha molts dispositius que es basen en aquest fenomen, com ara els fotoreceptors CCD de les càmeres digitals. D'altra banda, les propietats ondulatòries de la llum s'han verificat amb experiments òptics d'interferència[6] i se'n té una descripció matemàtica i física consistent. Aleshores, la llum és una ona o una partícula? Probablement, la resposta més acceptada ara per ara és: totes dues coses alhora.

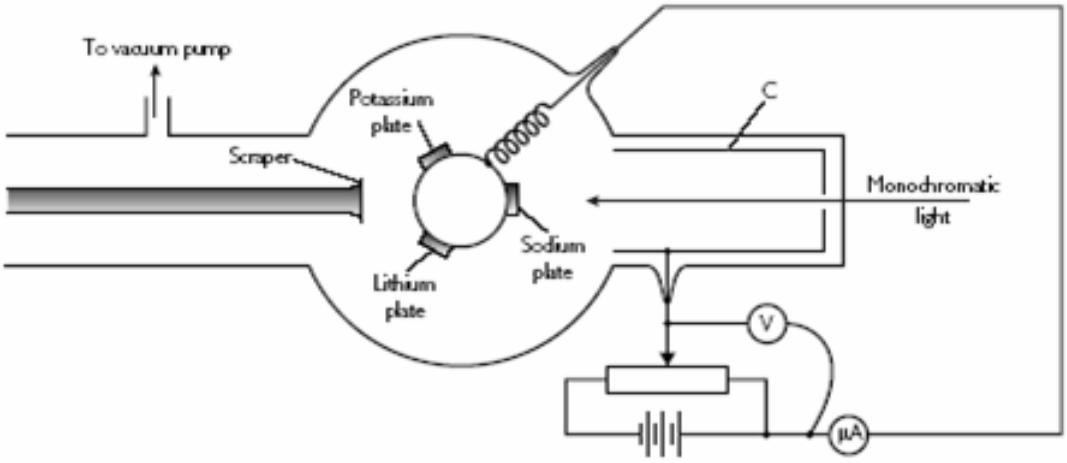

Figura 2.2: L'experiment de Millikan per verificar l'efecte fotoelèctric. [14] S'iluminen els metalls amb llum i s'introdueix un potencial de retard per frenar els fotoelectrons.

2.4 Dualitat ona-partícula

S'ha vist que la llum es pot comportar com una partícula (Millikan) i la interferència de la llum, un comportament típic de les ones, és un fenomen ben conegut. Per tal de donar

[4] Correspon al plural de la paraula *quàntum*.

[5] L'explicació descriu un efecte d'interacció entre partícules. Tal com passa en un xoc elàstic entre el fotó de la llum i l'electró del metall.

[6] N'és un exemple l'experiment de la doble escletxa de Young.

una justificació de la naturalesa dual de la llum, s'utilitza l'exemple de l'experiment de Young (veure figura 2.3) amb la hipòtesi que la llum és una partícula.

En efecte, si s'envia un feix de llum (de fotons) a través de les dues escletxes (A), el que s'observa és el patró d'interferència que es coneix de les ones. Es podria pensar que aquest patró correspon a la interacció corpuscular entre els fotons que passen per cada una de les escletxes. Per tal d'evitar caure en aquest parany, es pot reduir la cadència amb què s'envien aquests fotons fins a arribar a l'extrem d'enviar-los d'un en un. Ara, si s'observa la pantalla de mesura (B), es poden diferenciar els punts en els quals impacten els fotons. El patró pot semblar erràtic quan el nombre de fotons enviats és reduït, però, si s'allarga el temps d'exposició, apareix el mateix patró d'interferència. Com es pot acceptar, aleshores, que no es tracta d'una ona?

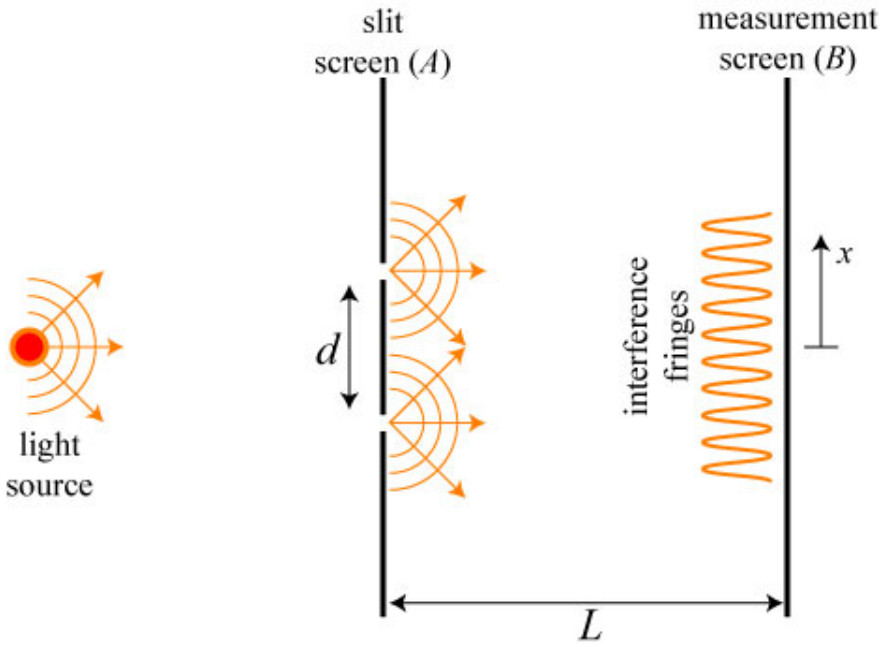

Figura 2.3: L'experiment de Young que verifica la naturalesa dual de la llum. [11]

D'altra banda, quan el nombre de fotons és reduït, l'aspecte de la pantalla de mesura no s'assembla a una versió dèbil del patró d'interferència resultant, sinó més aviat el contrari. El que es mostra en la pantalla de mesura semblen impactes de partícules. Com es pot acceptar, doncs, que no es tracta de partícules?

En resum, tenim que:

a) Si el temps d'exposició és llarg, apareix el patró d'interferència.
   $\Rightarrow$ Es descarta la descripció únicament corpuscular.

b) Si el temps d'exposició és molt curt, apareixen impactes perfectament localitzats.
   $\Rightarrow$ Es descarta la descripció únicament ondulatòria.

c) La distribució dels impactes és aleatòria, però la densitat és proporcional a la distribució del patró. En altres paraules, es va dibuixant el patró d'interferència punt a punt.

Aquest resultat es resumeix en el *principi de complementarietat*, el qual diu que un fenomen que s'explica amb un comportament ondulatori no pot ser explicat a la vegada per un comportament corpuscular. Recíprocament, un fenomen que s'explica amb un comportament corpuscular no pot ser explicat per un comportament ondulatori.

El que resulta més increïble d'aquest experiment és que no només funciona amb els fotons. Si es canvia la font de llum (*light source*) per un emissor de partícules (feix d'electrons), el resultat és el mateix!

2.5 Ones de De Broglie

Les idees que Einstein va introduir sobre la naturalesa dual de la llum es van generalitzar el 1924. Louis-Victor De Broglie va suggerir que les partícules materials,[7] en determinades condicions, també es comporten com a ones. La idea que va proposar De Broglie en la seva tesi és que les partícules materials tenen una freqüència característica associada i que, en principi, aquesta freqüència és el resultat d'algun tipus de comportament periòdic intern.

A partir de l'expressió de l'energia deduïda de la relativitat especial (2.5) i la fórmula (2.3), es pot aïllar la freqüència característica d'una partícula com es mostra en (2.6).

$$E = \frac{m_0\ c^2}{\sqrt{1 - \ (\frac{v}{c})^2}} = h\ \nu_0 \tag{2.5}$$

On m_0 correspon a la massa en repòs de la partícula; v, a la velocitat a què es mou la partícula, i ν_0, a la freqüència característica descrita per De Broglie.

$$\nu_0 = \frac{m_0\ c^2}{h\sqrt{1 - \ (\frac{v}{c})^2}} \tag{2.6}$$

A més, va establir les mateixes relacions per a la matèria que les que es tenien per la radiació electromagnètica:

$$E = h\ \nu_0, \qquad\qquad p = \frac{h}{\lambda_0} \tag{2.7}$$

On ν_0, λ_0, p i E corresponen respectivament a la freqüència, la longitud d'ona, el mòdul del moment lineal i l'energia associades a les partícules materials.

Les relacions de De Broglie s'han de prendre com a postulats i la seva validesa queda restringida als resultats dels experiments. En l'apartat anterior s'ha vist que l'experiment

[7] Les partícules materials són els corpuscles clàssics: els electrons, els protons, els àtoms o, en relació amb els últims experiments, fins i tot les molècules.

de Young[8] demostra la doble naturalesa de la llum. Aquest mateix experiment s'ha provat amb les partícules materials i el resultat ha estat el mateix: s'observa patró d'interferència. Aquest resultat, que ha estat provat amb partícules tan grans com el fluorofullerè,[9] fa evident que el postulat de De Broglie sembla anar pel bon camí.

Altres experiments que demostren la naturalesa ondulatòria de la matèria són: l'experiment de Davisson i Germer i el de George P. Thomson. El primer es basa en la reflexió de Bragg, però utilitza electrons en comptes de raigs X. El segon és un experiment semblant al de la doble escletxa de Young; en aquest cas, però, s'envia un feix d'electrons a través d'una làmina fina d'or i el que es produeix en la pantalla és un patró d'interferència circular. Aquests dos experiments van ser premiats amb el premi Nobel de física l'any 1937, precisament per contribuir a demostrar la naturalesa ondulatòria dels electrons, la qual confirma el postulat de De Broglie.

2.6 Principi d'indeterminació de Heisenberg

Aquest principi enunciat el 1927 diu que és impossible conèixer amb absoluta precisió la posició i el moment d'una partícula en un instant donat.[5] Aquesta limitació és independent del mètode de mesura i implica la impossibilitat de conèixer simultàniament dues variables conjugades. Aquesta indeterminació també apareix quan les variables que es volen conèixer són l'energia i el temps (també són conjugades).

Si s'associa Δvar a la incertesa associada a la variable var, el principi d'indeterminació es pot resumir en:

$$\Delta x\ \Delta p_x \geq \hbar, \qquad \Delta E\ \Delta t \geq \hbar \tag{2.8}$$

On Δx, Δp_x, ΔE i Δt són, respectivament, les incerteses associades a la posició, el moment, l'energia i el temps. El símbol $\hbar$ que apareix en la desigualtat és una cota,[10] s'anomena *h barra* i correspon a la constant de Planck dividida per 2π.

Com que la constant de Planck és molt petita, aquesta limitació només apareix en el món microscòpic (a escales més grans l'error és despreciable i es pot dir que es coneixen perfectament les dues magnituds relacionades).

Amb aquesta limitació present, en el món microscòpic no es pot parlar del resultat unívoc d'un experiment. En efecte, com que no es poden conèixer totes les variables inicials amb absoluta precisió, només es pot parlar de la probabilitat d'un experiment en funció dels possibles estats incials del sistema. Aquesta definició és clau per entendre el comportament de la nova unitat d'informació utilitzada en la computació quàntica.

[8] En realitat, és l'experiment de Young modificat (canviant la cadència de la llum). Young només va demostrar la naturalesa ondulatòria.

[9] El fluorofullerè és una mol·lècula massiva composta per 60 àtoms de carboni i 48 de fluor.[13]

[10] Aquesta cota a vegades apareix als llibres com $\frac{\hbar}{2}$. Tot i això, el valor exacte de la cota no és significatiu, la importància de la desigualtat rau en el fet que el producte d'incerteses és finit.

2.7 Equació de Schrödinger

A partir d'aquest punt del text es parla d'ona-partícules per referir-se a les partícules microscòpiques descrites per De Broglie. El comportament d'aquestes ona-partícules es pot descriure mitjançant un nou concepte: la funció d'ona.

La funció d'ona de les ona-partícules és complexa ($\in \mathbb{C}$) i correspon a la solució de l'equació de Schrödinger. En efecte, donada una ona-partícula sotmesa a un potencial que depèn de la posició i el temps, $V(x,t)$, el seu estat es pot caracteritzar mitjançant la funció d'ona $\psi(x,t)$ i aquesta funció compleix:

$$i\hbar \frac{\partial \psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2 \psi(x,t)}{\partial x^2} + V(x,t)\psi(x,t) \tag{2.9}$$

Schrödinger va deduir aquesta equació a partir d'una analogia formal entre òptica i mecànica. Va relacionar l'òptica geomètrica i l'equació de Hamilton-Jacobi per a ona-partícules no relativistes amb massa.

La funció d'ona solució de l'equació no té significat físic. No obstant això, el seu quadrat, com es detalla més endavant, correspon a la densitat de probabilitat del fet que la partícula es trobi en una posició i temps determinats (sempre que la funció d'ona hagi estat normalitzada correctament, de manera que la suma de probabilitat per a tot x sigui 1).

Un cas particular senzill és el potencial $V(x)$ descrit per un pou quadrat infinit. En aquest cas, el potencial és independent del temps (estacionari) i la seva representació correspon a la d'un pou amb parets infinites. Això equival a una força infinitament repulsiva en $x = 0$ i $x = L$ quan la partícula es troba en $(0, L)$.

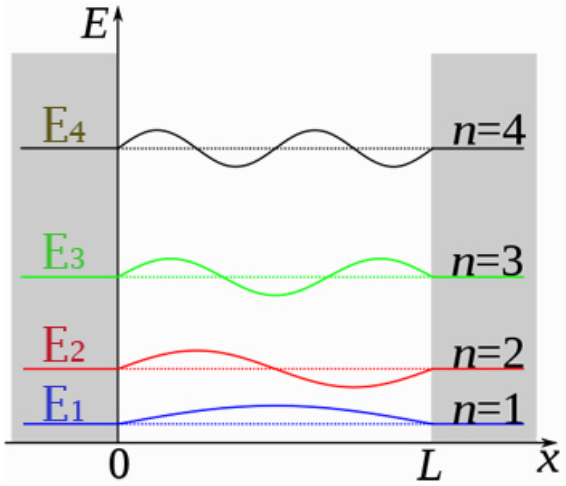

Figura 2.4: Solució de l'equació de Schrödinger: pou infinit de potencial.

En aquest cas, les solucions són de la forma descrita en (2.10) i se'n pot veure el comportament en la figura 2.4.

$$\psi_n(x,t) = \psi_n(x) = \sqrt{\frac{2}{L}} sin\left(\frac{n\pi x}{L}\right) \tag{2.10}$$

Els autovalors d'aquestes solucions, com es veu més endavant, corresponen als nivells d'energia permesos per l'ona-partícula, E_1, E_2, E_3..., i es relacionen perfectament amb la hipòtesi de Planck, amb la qual s'ha començat aquest bloc teòric. En mesurar l'energia de la partícula, el valor trobat és sempre un d'aquests autovalors; per tant, la partícula confinada presenta un cert estat, d'energia E_k, i un valor esperat en x, que és determinat per la funció d'ona $\psi_k(x)$. Més endavant, es veu la importància d'aquest fet per a la descripció de la computació quàntica.

2.8 Els postulats de la mecànica quàntica

A partir de totes les hipòtesis, descripcions, propietats i deduccions que s'han vist en el resum dels punts anteriors, i moltes altres que es van desenvolupar a principis del segle XX, va néixer la mecànica quàntica. Aquesta disciplina, que és relativament recent, estudia el comportament de les partícules microscòpiques, mitjançant les seves funcions d'ona, que han de ser solució de l'equació de Schrödinger.

A continuació es presenten els postulats de la mecànica quàntica per tal de preparar el terreny per a la computació quàntica, que es descriu en el bloc següent.

Postulat I:

Cada sistema físic té associat un espai de Hilbert (espai vectorial complex). A cada instant de temps, l'estat del sistema està completament determinat per un vector ψ normalitzat a la unitat. A més, l'evolució temporal d'aquest vector ψ és governada per l'equació de Schrödinger.

Postulat II:

A cada magnitud física se li associa un operador Hermític que la representa.

Postulat III:

Independentment de l'estat del sistema abans de la mesura d'una magnitud donada, el resultat posterior a la mesura és un dels valors propis de l'operador associat.[11]

Postulat IV:

Immediatament després de mesurar una magnitud, si el resultat ha estat l'autovalor x, l'estat del sistema col·lapsa a l'autovector associat a x.

[11] En l'exemple del pou infinit, la magnitud és l'energia i els valors propis mesurats corresponen als nivells permesos. L'operador associat a l'energia s'anomena *Hamiltonià* i apareix en l'equació de Schrödinger.

CAPÍTOL

3

BLOC TEÒRIC II: COMPUTACIÓ QUÀNTICA

3.1 Notació de Dirac

Tot sistema quàntic es pot representar mitjançant un espai de Hilbert (postulat I). Els estats del sistema són els vectors d'aquest espai i les transformacions quàntiques, les aplicacions lineals unitàries de coeficients complexos. En computació quàntica, i en física quàntica en general, és habitual utilitzar la notació de Dirac per representar els vectors de l'espai de Hilbert. Tot i així, en aquest text s'adopta, en cada cas, la notació més còmoda per a la seva correcta interpretació. A continuació, es presenta la notació de Dirac i la seva equivalència vectorial.

Com s'ha vist, en resoldre l'equació de Schrödinger s'obtenen les funcions d'ona pròpies, associades a una energia (autovalor), que en són solució. Com que la seva superposició també n'és solució, es pot interpretar que les funcions trobades són la base de l'espai de solucions. Per exemple, en el cas del pou infinit de potencial, es pot descriure el sistema mitjançant l'expressió següent:

$$\psi(x,t) = c_1(t)\cdot\phi_1(x) + \cdots + c_i(t)\cdot\phi_i(x) + c_j(t)\cdot\phi_j(x) + \cdots \tag{3.1}$$

Ja s'ha comentat que els coeficients de les funcions no tenen sentit físic, però el seu mòdul al quadrat dóna la probabilitat que el sistema es trobi en aquest estat quan es realitzi la mesura. Per tal que això sigui així, és necessari normalitzar la base, és a dir, buscar que es compleixin les condicions d'ortonormalitat següents.

$$\int \phi_i(x)\phi_j^*(x)\,dx = 0, \qquad i \neq j$$

$$\int \phi_i(x)\phi_i^*(x)\,dx = 1$$

La notació de Dirac es basa en els anomenats *kets* i *bras*. Els kets s'utilitzen pels estats del sistema i les funcions d'ona. Els bras representen la conjugació dels kets (elements del dual) i resulten especialment intuïtius per resoldre el producte escalar i descriure alguns operadors (postulat II). Es poden veure alguns exemples per entendre millor la notació:

$$|\psi\rangle = c_1\,|\phi_1\rangle + c_2\,|\phi_2\rangle + \cdots \tag{3.2}$$

$$\langle\psi| = c_1^*\,\langle\phi_1| + c_2^*\,\langle\phi_2| + \cdots \tag{3.3}$$

$$\int \phi_1^*(x)\phi_2(x)\,dx \equiv \langle\phi_1|\phi_2\rangle \tag{3.4}$$

Per ordre: funció d'ona representada mitjançant kets (3.2), funció d'ona conjugada representada mitjançant bras (3.3) i producte escalar mitjançant un bra-ket (3.4).

A més, es pot fer un paral·lelisme amb altres notacions habituals, per exemple, pel cas de dimensió 2 en (3.1) es té:

$$|\psi\rangle = c_1\,|\phi_1\rangle + c_2\,|\phi_2\rangle \equiv c_1\begin{pmatrix}1\\0\end{pmatrix} + c_2\begin{pmatrix}0\\1\end{pmatrix} \tag{3.5}$$

$$\langle\phi_1|\phi_1\rangle = 1 \qquad \langle\phi_1|\phi_2\rangle = 0 \qquad \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = 1 \qquad \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = 0$$

$$\langle\phi_2|\phi_1\rangle = 0 \qquad \langle\phi_2|\phi_2\rangle = 1 \qquad \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = 0 \qquad \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = 1$$

Els estats $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ i $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ són base i verifiquen les condicions d'ortonormalitat

$$\langle 0|0\rangle = \langle 1|1\rangle = 1 \qquad \langle 0|1\rangle = \langle 1|0\rangle = 0 \tag{3.6}$$

3.2 Qubits, qutrits i qudits

En computació clàssica, la unitat d'informació és el bit, una representació abstracta d'una variable entera que només pot prendre dos valors: 0 i 1. La notació de Dirac permet representar d'una manera compacta la unitat d'informació de la computació quàntica, el qubit, el qual representa una superposició ponderada dels valors que pot prendre el bit clàssic:

$$|a\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.7}$$

$$\alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$$

La restricció que la suma dels coeficients al quadrat sigui 1 s'anomena *condició de normalització* i la seva finalitat és donar sentit físic al quadrat dels coeficients. Més endavant, es parla del procés de mesura i de la importància d'aquests coeficients. A partir d'aquesta condició es pot reescriure el qubit (3.7) en la seva forma més general possible —llevat d'una fase global inobservable— en funció de dos angles, θ i ϕ:

$$|a\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{3.8}$$

En efecte, el qubit, també anomenat *quantum bit*, pot emmagatzemar els dos valors clàssics simultàniament. Aquest comportament tan estrany correspon a la superposició de les funcions d'ona que s'ha vist en la introducció, però en el cas particular d'una partícula que només presenti dos nivells possibles. És clar que aquest concepte es podria estendre per a sistemes amb més nivells quàntics. Per exemple, en un espai de Hilbert de dimensió 3, es tenen els anomenats *qutrits*:

$$|a_3\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle \tag{3.9}$$

$$\alpha, \beta, \gamma \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$$

Pel cas genèric de d nivells quàntics, es té el *qudit*, que es pot representar com es mostra en l'equació següent:

$$|a_d\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle \qquad \alpha_i \in \mathbb{C}, \quad \sum_{i=0}^{d-1} |\alpha_i|^2 = 1 \tag{3.10}$$

Per extensió de la computació clàssica actual, i també per aprofitar totes les propietats de l'àlgebra de Boole, se centra l'atenció en el qubit. Aquesta restricció no fa perdre generalitat, ja que, tal com passa en la computació clàssica, qualsevol sistema representat per qudits pot ser representat mitjançant qubits.

3.3 Esfera de Bloch

Una representació geomètrica del qubit molt utilitzada en el món de la computació quàntica és l'*esfera de Bloch* (veure figura 3.1). Aquesta esfera representa el qubit genèric de l'equació (3.8) en funció dels angles θ i ϕ. Cada punt d'aquesta superfície esfèrica en $\mathbb{R}^3$ representa unívocament un estat del qubit.

Per trobar els estats ortogonals a $|\psi\rangle$ s'han de buscar els punts sobre l'esfera de Bloch que es troben a distància 2. Només n'hi ha un que ho compleix i es troba a l'altre extrem de la superfície de l'esfera. En efecte, una recta qualsevol que passa per l'origen presenta dos punts d'intersecció amb l'esfera de Bloch. Aquests punts són una parella d'estats ortogonals que es poden utilitzar per computar.

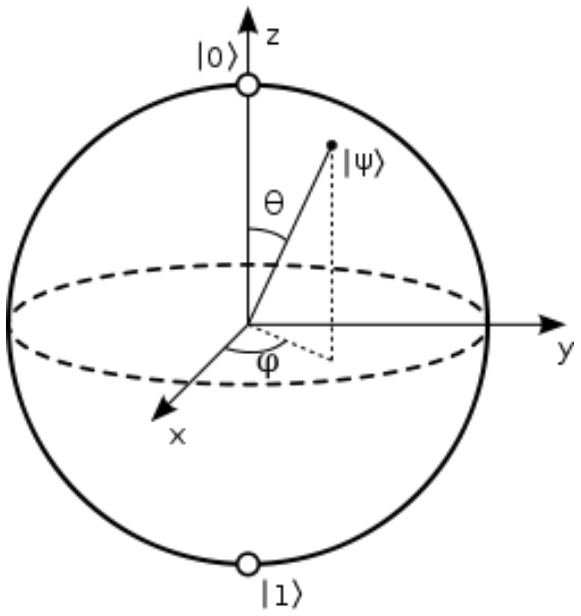

Figura 3.1: Esfera de Bloch

Es pot veure ràpidament que els estats $|0\rangle$ i $|1\rangle$ representats en la figura anterior són ortogonals, com s'ha indicat en (3.6). Habitualment s'utilitza l'eix Z per representar els estats del qubit per analogia amb l'espín,[1] que és un sistema quàntic amb dos nivells.

3.4 Sistemes amb n qubits

En els sistemes clàssics, l'extensió natural del bit resulta molt simple: cada bit és independent de la resta i pot ser llegit i modificat sense interferir en els altres. En els sistemes quàntics de més nivells, en canvi, en general es té una relació molt més complicada.

$$|x\rangle = \alpha\,|000\rangle + \beta\,|001\rangle + \cdots + \delta\,|110\rangle + \epsilon\,|111\rangle \tag{3.11}$$

La representació general d'un registre de n qubits té 2^n coeficients complexos. A més, es poden formar estats molt estranys. Per exemple, un estat com el que es mostra a continuació s'anomena *entrellaçat* (*entangled*).[2]

$$|x\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle \tag{3.12}$$

Es tracta d'un dels *estats de Bell*. En aquest estat, en llegir un dels dos qubits, es pot saber el valor de l'altre. Per exemple, si el primer qubit mesurat ha donat 0, l'únic valor possible per al següent també és 0. Aquesta propietat ha donat origen a la criptografia quàntica i s'utilitza més endavant en els algorismes basats en circuits quàntics.

[1] Nombre quàntic que descriu el moment angular intrínsec de les partícules elementals o dels nuclis.

[2] Correspon a la traducció que s'utilitza del concepte *entangled*, tot i que és més correcta la paraula *embrollat*.

3.5 Mesura

En els sistemes clàssics, llegir —o mesurar— un bit o un conjunts de bits no en modifica el valor. En computació quàntica això no és així. En realitzar la mesura d'un qubit o un conjunt de qubits, el que l'observador obté és un dels estats amb una certa probabilitat. Aquesta probabilitat, com s'ha comentat, correspon al mòdul al quadrat del coeficient complex que acompanya l'estat. Però en realitzar la mesura es *col·lapsa* l'estat al valor mesurat (postulat IV). Per exemple, si es considera el qubit $|a\rangle$ següent:

$$|a\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \tag{3.13}$$

En realitzar la mesura del qubit, l'observador només pot llegir $|0\rangle$ o $|1\rangle$ (postulat III) i el qubit queda modificat a $|\hat{a}\rangle$, com es mostra a continuació.

$$\text{Es mesura}\,|a\rangle = \begin{cases} |0\rangle & \text{amb una probabilitat del 25 \%} \implies |\hat{a}\rangle = |0\rangle \\ |1\rangle & \text{amb una probabilitat del 75 \%} \implies |\hat{a}\rangle = |1\rangle \end{cases} \tag{3.14}$$

És a dir, si se suposa que l'observador ha mesurat $|a\rangle$ i ha trobat el valor $|0\rangle$, un segon observador mesurarà $|\hat{a}\rangle = |0\rangle$ amb una probabilitat del 100 %, ja que s'ha trencat la superposició.

Això no ha de semblar estrany si es recorda que el procés de mesura d'un observable físic sempre modifica el que es vol observar. En computació quàntica l'efecte és molt més important, ja que les partícules observades tenen dimensions atòmiques ($\sim$ àngstrom).

Si en realitzar la mesura d'un qubit s'obté un únic estat i la resta es perden, com es pot computar? La resposta no és simple. En primer lloc, s'ha d'entendre la mesura del procés com la lectura final de l'experiment. Durant la cadena de computació no es pot realitzar cap lectura de l'estat parcial ni copiar-lo (*principi de no-clonació*). En segon lloc, es veurà que la computació quàntica es basa en la modificació dels coeficients, és a dir, en modificar les probabilitats de cada estat per obtenir a la sortida la solució buscada amb una probabilitat elevada i, en cas d'error, reiniciar tot el procés de càlcul.

3.6 Portes quàntiques d'un qubit

Si les variables en computació clàssica són els bits, les operacions i les funcions són les portes lògiques. La correspondència en el món de la computació quàntica d'una porta lògica és la denominada *porta quàntica*.

Les portes quàntiques han de respectar la normalització intrínseca dels qubits de manera que l'estat de sortida tingui una probabilitat total unitària. Per tal d'assegurar aquesta propietat, s'ha d'imposar que les matrius complexes associades a les portes siguin matrius unitàries:

$$\mathcal{U} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \qquad \mathcal{U}^H\mathcal{U} = \mathbb{I} \qquad \alpha, \beta, \gamma, \delta \in \mathbb{C}$$

Per a un únic bit hi ha dues portes clàssiques possibles, la Identitat i la Not.

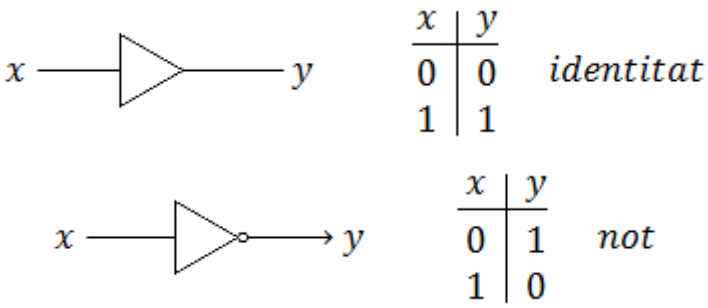

Figura 3.2: Portes lògiques d'un bit

Pel cas quàntic, en canvi, es poden trobar moltes més portes, ja que es pot realitzar una transformació dels coeficients complexos per canviar la probabilitat de cada estat. Per exemple, algunes portes importants són:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{Hadamard} \frac{\alpha}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{\beta}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Figura 3.3: Porta Hadamard

$$S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} \qquad \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{faseS} \alpha\,|0\rangle + \beta i\,|1\rangle$$

Figura 3.4: Porta fase S

$$T = \begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix} \qquad \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{faseT} \alpha\,|0\rangle + \beta e^{i\frac{\pi}{4}}\,|1\rangle$$

Figura 3.5: Porta fase T

Aquestes últimes, fase S i fase T, són casos particulars d'una porta més general que s'anomena *canvi de fase* (*Phase Shift*) i que introdueix una fase relativa entre els estats $|0\rangle$ i $|1\rangle$ d'un valor arbitrari ϕ.

$$\Phi = \begin{pmatrix}1 & 0\\ 0 & e^{i\phi}\end{pmatrix} \qquad \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{\Phi(\phi)} \alpha\,|0\rangle + \beta e^{i\phi}\,|1\rangle$$

Figura 3.6: Porta *Phase Shift*

Finalment, cal esmentar les portes de Pauli, entre les quals hi ha l'extensió de les portes clàssiques d'un bit vistes a 3.2.

$$\sigma_0 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_2 = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_3 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Figura 3.7: Portes de Pauli

Es pot comprovar fàcilment que totes les matrius anteriors són unitàries. De fet, si només es té en compte aquesta condició, es pot escriure la matriu 2×2 més general possible que representa una porta quàntica. Així doncs, en (3.15), tenim la matriu general $\mathcal{U}$ en funció dels angles θ, ϕ i les fases α i χ.

$$\mathcal{U}(\alpha, \theta, \phi, \chi) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) e^{i\phi} \\ \sin\left(\frac{\theta}{2}\right) e^{i\alpha} & \cos\left(\frac{\theta}{2}\right) e^{i\alpha} e^{i\phi} \end{pmatrix} e^{i\chi} \tag{3.15}$$

Aquesta porta general es pot construir combinant les portes Hadamard i *Phase Shift* com es mostra a continuació.

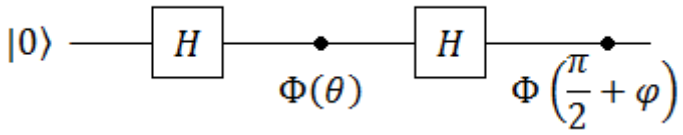

Figura 3.8: Esquema d'una porta general

Es pot comprovar que si s'aplica aquesta porta a un qubit inicialitzat a $|0\rangle$, s'obté el qubit general de (3.8), i. e. es pot desplaçar a qualsevol punt de l'esfera de Bloch.

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \xrightarrow{Hadamard} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \xrightarrow{\Phi(\theta)} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ e^{i\theta} \end{pmatrix}$$

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ e^{i\theta} \end{pmatrix} \xrightarrow{Hadamard} \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ e^{i\theta} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 + e^{i\theta} \\ 1 - e^{i\theta} \end{pmatrix}$$

$$\frac{1}{2} \begin{pmatrix} 1 + e^{i\theta} \\ 1 - e^{i\theta} \end{pmatrix} \xrightarrow{\Phi(\frac{\pi}{2}+\varphi)} \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}+\varphi} \end{pmatrix} \begin{pmatrix} 1 + e^{i\theta} \\ 1 - e^{i\theta} \end{pmatrix} = e^{i\frac{\theta}{2}} \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} e^{i\varphi} \end{pmatrix}$$

On el terme $e^{i\frac{\theta}{2}}$ és una fase global del qubit que no es pot mesurar.

3.7 Portes quàntiques de n qubits

En general, es necessiten matrius unitàries $2n \times 2n$ per representar les portes quàntiques de n qubits. L'extensió natural de les portes d'un qubit es pot aconseguir utilitzant el producte tensorial de les matrius aplicades a cada qubit. Per exemple, a continuació es mostra com generar l'extensió de la porta Hadamard en un registre de dos qubits. En aquest cas, la porta Hadamard s'ha aplicat al primer qubit, mentre que el segon es deixa invariant (porta Identitat).

$$H \otimes \mathbb{I} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \tag{3.16}$$

Tot i així, resulta molt més interessant estudiar les matrius que no es poden construir d'aquesta manera. Per exemple, la porta de dos qubits anomenada *Control-Not* (C-Not) es pot representar mitjançant una matriu 4×4 de coeficients reals.

$$\text{C-Not} = C_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.17}$$

Aquesta porta representa un inversor sobre un dels qubits controlat per l'estat de l'altre qubit. En la figura 3.9 es mostra l'esquema de la C-Not que utilitza el qubit $|x\rangle$ per controlar l'estat del qubit $|y\rangle$.

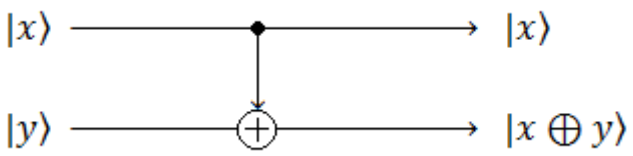

Figura 3.9: Porta Control-Not

És fàcil imaginar que el nombre i la complexitat d'aquestes portes creix ràpidament quan s'augmenta el nombre de qubits. Tot i així, és possible aproximar les portes generals mitjançant famílies finites universals.[3]

[3] Per a més informació sobre portes universals, consulteu [2], p. 118.

Per exemple, la porta *Control-Shift* (C-S), que es mostra en (3.18), juntament amb la porta Hadamard d'un qubit (veure figura 3.3) formen un conjunt finit de portes que poden aproximar qualsevol porta unitària de 2 qubits.

$$\text{C-S} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix} \tag{3.18}$$

3.8 Avaluació de funcions quàntiques

S'han vist alguns exemples de transformacions d'estats de qubits, però encara no s'ha dut a terme cap tipus de computació. Per avaluar funcions en ordinadors quàntics, es parteix de dos registres: un d'ells és l'entrada (*input*) i l'altre s'inicialitza a un estat conegut (*output*).

Mitjançant els operadors unitaris, i. e. les portes, es modifica l'estat del registre conegut utilitzant la informació del registre d'entrada.

$$\underbrace{|x\rangle}_{input}\underbrace{|0\rangle}_{output} \xrightarrow{\mathcal{U}_f} |x\rangle\,|f(x)\rangle \tag{3.19}$$

Que també es pot representar com es mostra en la figura 3.10.

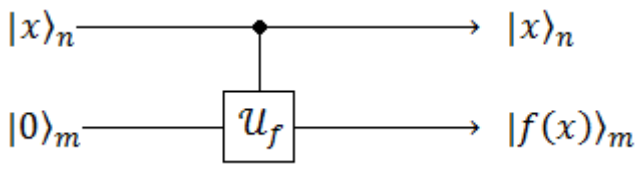

Figura 3.10: Esquema d'una funció quàntica

A continuació, es fa una breu descripció de tres algorismes importants que han demostrat la millora que pot representar la computació quàntica davant la clàssica: els algorismes de Deutsch-Jozsa, Shor i Grover.

3.9 Algorisme de Deutsch-Jozsa

Algorisme ideat per David Deutsch i Richard Jozsa el 1992. Aquest algorisme determina el comportament d'una funció binària amb una única crida.

$$\begin{aligned} f : \{0,1\} &\longrightarrow \{0,1\} \\ x &\longrightarrow f(x) \end{aligned} \tag{3.20}$$

En efecte, donada una funció binària desconeguda, es vol determinar si es tracta d'una funció constant ($f(0) = f(1)$) o balancejada ($f(0) \neq f(1)$). Les úniques funcions binàries possibles són les següents:

$$\begin{array}{lll} f_1(0) = 0 & f_1(1) = 0 & \text{constant} \\ f_2(0) = 0 & f_2(1) = 1 & \text{balancejada} \\ f_3(0) = 1 & f_3(1) = 0 & \text{balancejada} \\ f_4(0) = 1 & f_4(1) = 1 & \text{constant} \end{array} \tag{3.21}$$

La computació clàssica necessita dues crides a la funció per determinar-ne el tipus. Cal avaluar la funció en 0, després en 1 i, finalment, comprovar si els valors coincideixen.

L'algorisme de Deutsch utilitza la porta Hadamard per crear una superposició dels dos estats purs $|0\rangle$ i $|1\rangle$ i per avaluar la funció en els dos estats alhora. Mitjançant la interferència dels valors de la funció s'obté la informació desitjada. A continuació, es mostren l'esquema del circuit general (figura 3.11) i el desenvolupament matemàtic pel cas més simple.

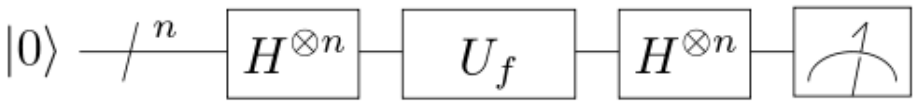

Figura 3.11: Circuit general de Deutsch

On el bloc $H^{\otimes n}$ correspon al producte tensorial de n portes Hadamard[4] i el bloc $\mathcal{U}_f$ és el que conté la informació de la funció binària en estudi.

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \xrightarrow{Hadamard} \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \xrightarrow{\mathcal{U}_f} \tfrac{1}{\sqrt{2}} \begin{pmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} (-1)^{f(0)} \\ (-1)^{f(1)} \end{pmatrix}$$

$$\tfrac{1}{\sqrt{2}} \begin{pmatrix} (-1)^{f(0)} \\ (-1)^{f(1)} \end{pmatrix} \xrightarrow{Hadamard} \tfrac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} (-1)^{f(0)} \\ (-1)^{f(1)} \end{pmatrix} = \tfrac{1}{2} \begin{pmatrix} (-1)^{f(0)} + (-1)^{f(1)} \\ (-1)^{f(0)} - (-1)^{f(1)} \end{pmatrix}$$

Si s'analitza la sortida pel cas en què $f(0) = f(1)$ (funció constant):

$$\frac{1}{2} \begin{pmatrix} (-1)^{f(0)} + (-1)^{f(1)} \\ (-1)^{f(0)} - (-1)^{f(1)} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} (-1)^{f(0)} + (-1)^{f(0)} \\ (-1)^{f(0)} - (-1)^{f(0)} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \pm 2 \\ 0 \end{pmatrix} = \begin{pmatrix} \pm 1 \\ 0 \end{pmatrix} = \pm |0\rangle \tag{3.22}$$

Mentre que si s'analitza pel cas en què $f(0) \neq f(1)$ (funció balancejada):

$$\frac{1}{2} \begin{pmatrix} (-1)^{f(0)} + (-1)^{f(1)} \\ (-1)^{f(0)} - (-1)^{f(1)} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \pm 1 \mp 1 \\ \pm(-1) - \mp(-1) \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 \\ \pm 2 \end{pmatrix} = \begin{pmatrix} 0 \\ \pm 1 \end{pmatrix} = \pm |1\rangle \tag{3.23}$$

[4] És equivalent a aplicar la porta Hadamard a cada qubit.

Per tant, de (3.22) i (3.23) es dedueix que si a la sortida es troba $|0\rangle$, implica que la funció és constant; mentre que si a la sortida es mesura $|1\rangle$, la funció és necessàriament balancejada.

Resulta interessant destacar que l'algorisme no permet conèixer el valor de la funció com passava en la crida clàssica. En aquest cas, el circuit quàntic ha implicat canviar un tipus d'informació per un altre.

3.10 Algorisme de Shor

3.10.1 Descripció

Algorisme ideat per Peter Williston Shor el 1994. Aquest algorisme representa un dels grans èxits de la computació quàntica i ha donat lloc a noves vies de recerca per a físics i enginyers d'arreu del món. A partir dels fenòmens d'entrellaçament i superposició, i juntament amb la transformada quàntica de Fourier (QFT), s'aconsegueix resoldre el problema de la factorització de nombres enters.

En efecte, calcular el producte de dos nombres enters resulta molt simple. Els processadors actuals més bàsics poden multiplicar nombres primers de centenars de bits en un temps d'execució molt baix. El pas invers —trobar els factors primers d'aquest producte— resulta un procés realment costós. De fet, només cal assenyalar que la criptografia RSA,[5] usada actualment en les comunicacions, es basa en aquesta dificultat.

El procés clàssic de factorització de nombres grans (p. e. 2048 bits) presenta un temps d'execució de l'ordre de 10^{15} anys (més elevat que l'edat estimada de l'univers), per la qual cosa resulta totalment inviable. L'eficiència de l'algorisme de Shor, en canvi, podria superar àmpliament qualsevol altre equip conegut si s'arribés a executar en un ordinador quàntic i, pel que fa al temps d'execució, es reduiria a desenes de minuts.

L'algorisme de Shor consta de dues parts ben diferenciades, una de clàssica i una altra de quàntica. La part quàntica es preocupa exclusivament de trobar el període d'una funció modular entera. La part clàssica s'encarrega de trobar algun dels factors utilitzant la informació proporcionada per la part quàntica.

A continuació, es presenta la llista de passos que cal seguir per trobar un factor no trivial d'un nombre N donat:

1) S'escull un nombre aleatori y_0 tal que $gcd(y_0, N) = 1$.[6]
(Si $gcd(y_0, N) \neq 1$ ja es té un factor no trivial).

2) S'inicialitzen dos registres de qubits $|x\rangle\,|f(x)\rangle$ a l'estat $|0\rangle\,|0\rangle$.

[5] Veure «Criptografia basada en RSA» en el bloc teòric III.
[6] En aquest treball s'utilitza *gcd* (*greatest common divisor*) com a abreviatura de *màxim comú divisor*.

3) S'aplica $H^{\otimes n}$ al registre d'entrada $|x\rangle$ per obtenir una superposició dels estats.

4) S'avalua la funció quàntica exponencial $|x\rangle\,|{y_0}^x \; mod \; N\rangle$.

5) Es mesura el registre de sortida $|f(x)\rangle$.

6) S'aplica la QFT al registre d'entrada $|x\rangle$.

7) Es mesura el registre d'entrada $|x\rangle$ i es guarda el valor.

8) Es determina r, el període de la funció $f(x) = {y_0}^x \; mod \; N$.

9) Es calculen candidats a factors a partir del període.

Cal destacar que en aquest algorisme es pressuposa que N, el nombre que volem factoritzar, és un nombre no primer. De fet, si s'utilitza per trencar RSA, N serà producte de dos nombres primers, P i Q, i factoritzar N és equivalent a trobar aquests dos factors.

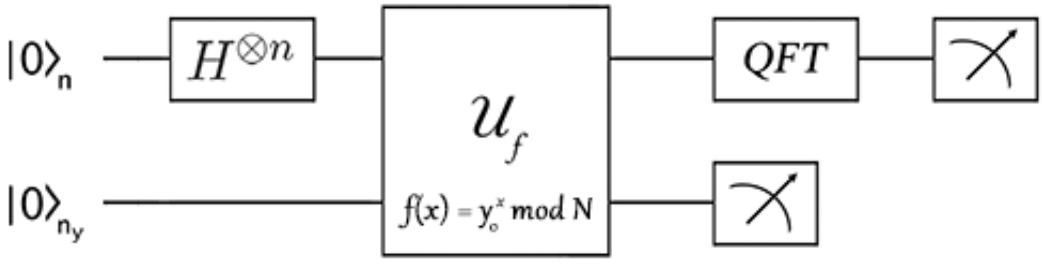

Figura 3.12: Circuit de Shor

3.10.2 L'algorisme pas a pas

A continuació, es detallen tots els passos de la llista anterior i la metodologia a seguir segons els resultats que s'obtinguin en cada un d'ells. Els passos 1, 8 i 9 corresponen a la part clàssica, la qual es pot executar en un ordinador convencional. Els passos del 2 al 7 (veure figura 3.12), en canvi, s'han d'implementar forçosament en un circuit quàntic perquè l'algorisme sigui eficient.

Pas 1: escollir y_0

S'escull un nombre enter y_0 aleatori, de manera que $1 < y_0 < N$, i es calcula, mitjançant l'algorisme d'Euclides, el màxim comú divisor entre aquest nombre i N:

$$M = gcd(y_0, N)$$

a) Si $M \neq 1$, aleshores M és un factor no trivial de N i s'ha acabat l'algorisme.
b) Si $M = 1$, es pot passar al pas següent.

Pas 2: inicialització

Es preparen els registres de qubits del circuit quàntic $|x\rangle\,|f(x)\rangle$.

Per a l'entrada, $|x\rangle$, s'utilitzen $n = 2 \cdot log_2(N)$ qubits. Aquesta n correspon al mínim nombre de qubits necessaris per codificar N^2, ja que això dóna un bon compromís de velocitat i eficiència. Per a la sortida, $|f(x)\rangle$, s'utilitzen $n_y = log_2(N)$ qubits. Per a n_y es pren el mínim nombre de qubits necessaris per codificar el període.[7]

Els registres, de $n + n_y$ qubits en total, s'inicialitzen a l'estat $|0\rangle\,|0\rangle$ i es pot passar al pas següent.

Pas 3: superposició

S'aplica $H^{\otimes n}$ al registre d'entrada (formalment s'aplica la identitat a la sortida) per obtenir una superposició dels estats possibles d'entrada. Aquest estat no és entrellaçat, ja que la mesura de cap dels qubits no pot influir sobre la probabilitat de la resta.

$$\underbrace{|0\rangle}_{entrada}\ \underbrace{|0\rangle}_{sortida} \xrightarrow{H^{\otimes n}\otimes \mathbb{I}^{n_y}} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle\,|0\rangle \tag{3.24}$$

Aquesta superposició permet fer una avaluació massiva dels *inputs* en el pas següent.

Pas 4: avaluació

S'avalua la funció quàntica exponencial $|x\rangle\,|{y_0}^x \ mod\ N\rangle$ mitjançant l'operador unitari $\mathcal{U}_f$, que genera un estat entrellaçat entre cada entrada i la sortida corresponent. Aquest operador és el més complicat d'implementar de l'algorisme, ja que ha de ser capaç d'actuar sobre l'estat superposició. El nombre de portes C-Not que s'han d'utilitzar pot arribar a ser molt elevat.

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle\,|0\rangle \xrightarrow{\mathcal{U}_f} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle\,|{y_0}^x \ mod\ N\rangle \tag{3.25}$$

Tots els valors del registre d'entrada són equiprobables i estan entrellaçats amb les sortides corresponents. Per tant, si es realitzés la mesura del registre complet, el que s'obtindria és una parella x i $f(x)$.

No és útil trencar la superposició amb una mesura de tot el registre, ja que una parella aletòria de x i $f(x)$ no serveix per obtenir el període i, a més, es pot aconseguir fàcilment amb computació clàssica. Per aquest motiu, en el pas següent s'usa una estratègia millor, una mesura parcial.

[7] Veure «Teorema d'Euler» en el bloc teòric III.

Pas 5: mesura de la sortida

Es mesura únicament el registre de sortida i això provoca que, a causa de l'entrellaçament, es produeixi un colapse dels estats de l'entrada. El registre d'entrada esdevé una superposició dels valors de x tals que la seva imatge per f coincideix amb el que s'ha mesurat.

Com que la funció $f(x) = f(x + kr) = y_0{}^x \ mod \ N$ és periòdica amb període r, tots els valors del registre d'entrada són de la forma $x_0 + kr$, amb k natural i x_0 la primera entrada.

$$|x\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle \tag{3.26}$$

On el paràmetre $m \geq \dfrac{2^n}{r}$ i la igualtat es dóna si $\dfrac{2^n}{r} \in \mathbb{Z}$.

Pas 6: QFT

S'aplica l'operador F associat a la QFT (3.27). Aquesta transformada quàntica representa una extensió de la transformada discreta de Fourier (DFT) aplicada a l'espai d'estats del registre.

$$F\,|x\rangle_n = QFT\,|x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} e^{i\frac{2\pi x l}{2^n}} |l\rangle \tag{3.27}$$

Com que la QFT es basa en una aplicació lineal sobre les amplituds, aquesta transformació també es pot representar mitjançant una matriu unitària (operador) com la que es mostra en (3.28).

$$F = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{\frac{2\pi i}{2^n}} & e^{2\frac{2\pi i}{2^n}} & e^{3\frac{2\pi i}{2^n}} & \dots & e^{(2^n-1)\frac{2\pi i}{2^n}} \\ 1 & e^{2\frac{2\pi i}{2^n}} & e^{4\frac{2\pi i}{2^n}} & e^{6\frac{2\pi i}{2^n}} & \dots & e^{2(2^n-1)\frac{2\pi i}{2^n}} \\ 1 & e^{3\frac{2\pi i}{2^n}} & e^{6\frac{2\pi i}{2^n}} & e^{9\frac{2\pi i}{2^n}} & \dots & e^{3(2^n-1)\frac{2\pi i}{2^n}} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{(2^n-1)\frac{2\pi i}{2^n}} & e^{2(2^n-1)\frac{2\pi i}{2^n}} & e^{3(2^n-1)\frac{2\pi i}{2^n}} & \dots & e^{(2^n-1)(2^n-1)\frac{2\pi i}{2^n}} \end{pmatrix} \tag{3.28}$$

Cada element a_{ij} d'aquesta matriu és una potència de $W = e^{\frac{2\pi i}{2^n}}$. Així, una manera compacta de representar l'operador de la QFT és:

$$F = \frac{1}{\sqrt{2^n}}\{a_{ij}\} = \frac{1}{\sqrt{2^n}}\{W^{(i-1)(j-1)}\} \tag{3.29}$$

El resultat d'aplicar aquest operador al registre d'entrada (3.26) es pot veure en (3.30). La informació de la periodicitat apareix explícita en l'espai dual (freqüència) i, tot i que no es veu a simple vista en l'expressió, es pot passar al pas següent per comprovar-ho.

$$|x\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle \xrightarrow{F} \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} e^{i\frac{2\pi(x_0+kr)l}{2^n}} |l\rangle \tag{3.30}$$

$$\sum_{l=0}^{2^n-1} \underbrace{\frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^n}} e^{i\frac{2\pi x_0 l}{2^n}} \sum_{k=0}^{m-1} e^{i\frac{2\pi krl}{2^n}}}_{a_l} |l\rangle = \sum_{l=0}^{2^n-1} a_l |l\rangle$$

Pas 7: mesura de l'entrada

La probabilitat d'obtenir l'estat $|l\rangle$ en mesurar el registre d'entrada és $|a_l|^2$. És important destacar que desapareix la dependència de x_0 quan es calcula el mòdul al quadrat del coeficient a_l.

$$P(l) = |a_l|^2 = \frac{1}{m}\frac{1}{2^n} \left| \sum_{k=0}^{m-1} e^{i\frac{2\pi krl}{2^n}} \right|^2 \tag{3.31}$$

Per tal d'especificar els possibles valors mesurats, es defineixen els l_j d'interès:

$$l_j = j\frac{2^n}{r} + \Delta_j, \qquad j \in \mathbb{Z},\ \ \Delta_j \in \mathbb{R} \ \text{ i } \ |\Delta_j| \leq \tfrac{1}{2}$$

A partir de (3.31) es pot veure que si $\frac{2^n}{r} \in \mathbb{Z}$ ($\Delta_j = 0$), la probabilitat és nul·la per a tots els $l \neq l_j$:

$$\sum_{k=0}^{m-1} e^{i\frac{2\pi krl}{2^n}} = 1 + e^{i\frac{2\pi rl}{2^n}} + e^{i\frac{2\pi rl}{2^n}2} + \cdots + e^{i\frac{2\pi rl}{2^n}(m-1)} = \frac{1 - e^{i\frac{2\pi rl}{2^n}m}}{1 - e^{i\frac{2\pi rl}{2^n}}} = \frac{1-1}{1 - e^{i\frac{2\pi rl}{2^n}}} = 0$$

En canvi, els l_j apareixen amb una probabilitat $P(l_j) \neq 0$:

$$\sum_{k=0}^{m-1} e^{i\frac{2\pi krl}{2^n}} = \sum_{k=0}^{m-1} 1 = m \implies P(l_j) = \frac{1}{m}\frac{1}{2^n} m^2 = \frac{m}{2^n}$$

Pel cas general en què $\Delta_j \neq 0$, el que s'obté és una distribució com la que es mostra en la figura 3.13.

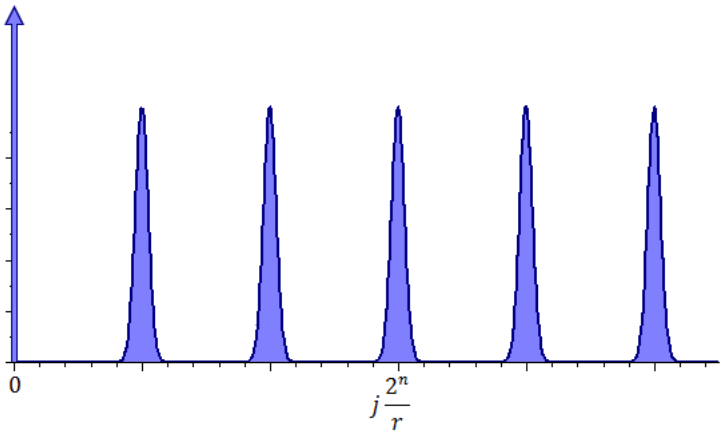

Figura 3.13: Distribució de probabilitat en $|x\rangle$ després de la QFT

De nou, a partir de (3.31) es pot trobar la probabilitat dels l_j:

$$P(l_j) = \frac{1}{m}\frac{1}{2^n}\left|\sum_{k=0}^{m-1} e^{i\frac{2\pi kr}{2^n}(j\frac{2^n}{r}+\Delta_j)}\right|^2 =$$

$$= \frac{1}{2^n m}\left|\sum_{k=0}^{m-1} e^{i\frac{2\pi kr}{2^n}\Delta_j}\right|^2 = \frac{1}{2^n m}\left|\frac{1-e^{i\frac{2\pi kr}{2^n}\Delta_j m}}{1-e^{i\frac{2\pi kr}{2^n}\Delta_j}}\right|^2 = \frac{1}{2^n m}\left|\frac{sin\left(\frac{\pi r\Delta_j m}{2^n}\right)}{sin\left(\frac{\pi r\Delta_j}{2^n}\right)}\right|^2$$

Si s'utilitza que en l'interval $0 \leq x \leq \frac{\pi}{2}$, es pot acotar el sinus per $\frac{x}{\frac{\pi}{2}} \leq sin\ x \leq x$, llavors:

$$P(l_j) > \frac{1}{2^n m}\frac{\left|\frac{\pi r\Delta_j m 2}{\pi 2^n}\right|^2}{\left|\frac{\pi r\Delta_j}{2^n}\right|^2} = \frac{4}{r\pi^2} \implies \sum_{k=1}^{r-1} P(l_j) = \frac{4(r-1)}{r\pi^2} \approx \frac{4}{\pi^2} \approx 0,4$$

De manera que, pel cas general, la probabilitat de trobar algun dels valors d'interès l_j és d'un 40 %.

Finalment, s'analitza el valor mesurat,

a) Si es mesura $|0\rangle$, l'algorisme ha fallat i s'ha de tornar al pas 2.
b) Si es mesura $|l\rangle \neq |0\rangle$, es guarda el valor i es pot passar al pas següent.

Pas 8: càlcul del període

A partir del valor obtingut en el pas anterior, es pot calcular el període amb un algorisme clàssic recursiu. Aquest algorisme es basa a aproximar successivament la fracció $\frac{l_j}{2^n}$ mitjançant els seus convergents, tal com es veu en el capítol 4.

En funció del resultat obtingut en aquest algorisme:

a) Si no es troba el període, l'algorisme ha fallat i s'ha de tornar al pas 2.
b) Si el període és imparell, l'algorisme ha fallat i s'ha de tornar al pas 1.
c) Si el període és parell, es guarda el valor i es pot passar a l'últim pas.

Pas 9: càlcul dels factors

Com que el període obtingut és parell:

$$y_0{}^r \equiv 1\ mod\ N \implies \left(y_0{}^{\frac{r}{2}}\right)^2 - 1 \equiv 0\ mod\ N \implies (y_0{}^{\frac{r}{2}}+1)(y_0{}^{\frac{r}{2}}-1) \equiv 0\ mod\ N$$

Per tant, es poden calcular els candidats a factors no trivials de N, p_a i p_b, amb l'algorisme d'Euclides.

$$\begin{aligned} p_a &= gcd(y_0^{\frac{r}{2}}+1, N) \\ p_b &= gcd(y_0^{\frac{r}{2}}-1, N) \end{aligned} \tag{3.32}$$

Si aquests candidats resulten ser els factors trivials del nombre, N i 1, el valor y_0 escollit no serveix. D'altra banda, és interessant remarcar que l'algorisme no funciona si es tracta d'una potència de nombres primers. En aquest cas, el període dels y_0 serà imparell o els candidats p_a i p_b trobats sempre seran els factors trivials.[8]

Finalment, cal dir que tots aquests passos es tornen a veure en la descripció del codi de l'aplicació QGT 04, la qual, com es comenta en la introducció, implementa una simulació de l'algorisme de Shor.

3.11 Algorisme de Grover

3.11.1 Descripció

Algorisme ideat per Lov Kumar Grover el 1998. Aquest algorisme s'ocupa de realitzar cerques en bases de dades amb una eficiència molt més gran que la clàssica. Concretament, en una llista de N entrades, la computació clàssica necessitaria, de mitjana, $\frac{N}{2}$ lectures per trobar un element en concret. L'algorisme de Grover necessita de l'ordre de $\sqrt{N}$ lectures per trobar-lo.

Per posar un exemple, si es tenen un bilió d'elements en una llista, mitjançant computació clàssica es necessitarien, de mitjana, $5 \cdot 10^{11}$ lectures. Amb l'algorisme de Grover, en canvi, es necessitarien de l'ordre de 10^6 lectures, i. e. la tasca es realitzaria mig milió de vegades més ràpid. En altres paraules, un servidor treballant a una comprovació per microsegon tardaria més de 8 minuts a trobar l'element utilitzant un ordinador clàssic; mentre que amb un ordinador quàntic, tardaria un segon.

Es pot veure fàcilment que aquesta millora és més gran com més gran és la llista d'elements. De fet, es torna especialment important quan es té en compte la quantitat de pàgines web que ha de comprovar un buscador —com ara Google— quan realitza les cerques a Internet.

L'algorisme de Grover utilitza un operador *oracle*, molt semblant al $\mathcal{U}_f$ (figura 3.11) que s'ha vist en l'algorisme de Deutsch-Jozsa, i un altre operador anomenat *inversor respecte la mitjana* (*Diffusion Operator*), que es pot descriure com es mostra en (3.33).

$$H^{\otimes n}[2\left|0\right\rangle\left\langle 0\right| - \mathbb{I}]H^{\otimes n} = [2\left|\psi\right\rangle\left\langle\psi\right| - \mathbb{I}] \tag{3.33}$$

$$\text{Essent } \left|\psi\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle \text{ una superposició de tots els valors possibles.}$$

En la figura 3.14, es pot veure el circuit complet de l'algorisme de Grover que utilitza $n = \log_2(N)$ qubits per tal de codificar els N elements de la llista.

[8] En el capítol dedicat als models matemàtics, es justifica aquesta afirmació.

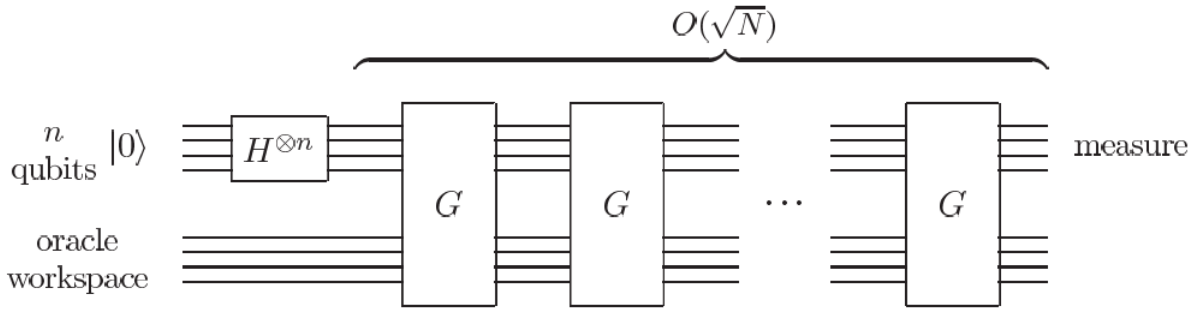

Figura 3.14: Circuit general de Grover

On cada un dels blocs G (veure figura 3.15) estan formats per un operador oracle seguit de l'inversor respecte la mitjana (3.33). Els blocs G són les diferents iteracions de l'algorisme de Grover i, per tal d'optimitzar-lo, se n'ha d'escollir correctament el nombre.

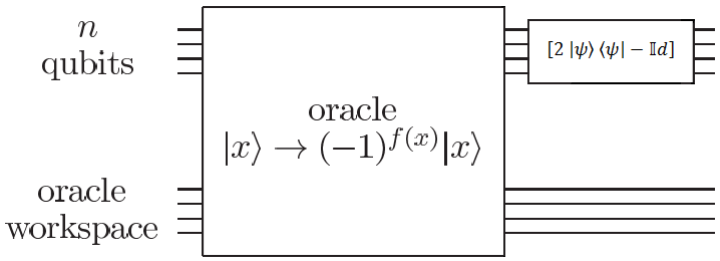

Figura 3.15: Bloc G de Grover

3.11.2 Iteració de Grover

Per analitzar el comportament complet d'una iteració de l'algorisme de Grover, s'ha d'especificar, en primer lloc, el comportament de la funció oracle. En aquest cas, l'operador $\mathcal{U}_f$ ha de ser capaç d'identificar l'element buscat $|x_0\rangle$, de manera que $f(x_0) = 1$ i $f(x) = 0$ $\forall x \neq x_0$. Més concretament, després d'aplicar $H^{\otimes n}$ sobre l'estat $|0\rangle$, es crea una superposició de tots els valors possibles i, a continuació, l'oracle s'encarrega de canviar el signe de l'element buscat.

$$|0\rangle \xrightarrow{Hadamard} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \xrightarrow{\mathcal{U}_f(oracle)} \frac{1}{\sqrt{2^n}} \left(\sum_{x=0}^{2^n-1} |x\rangle - 2\,|x_0\rangle \right) = |\psi\rangle - \frac{2}{\sqrt{2^n}}\,|x_0\rangle$$

En aplicar l'operador inversor respecte la mitjana, el pes de l'element buscat es veu modificat, de manera que, si es fes la mesura, el valor trobat més probable seria $|x_0\rangle$.

$$|\psi\rangle - \frac{2}{\sqrt{2^n}}\,|x_0\rangle \xrightarrow{[2|\psi\rangle\langle\psi|-\mathbb{I}]} 2\,|\psi\rangle\,\langle\psi|\psi\rangle - |\psi\rangle - \frac{4}{\sqrt{2^n}}\,|\psi\rangle\,\langle\psi|x_0\rangle + \frac{2}{\sqrt{2^n}}\,|x_0\rangle =$$

$$= |\psi\rangle - \frac{4}{\sqrt{2^n}} |\psi\rangle \langle\psi|x_0\rangle + \frac{2}{\sqrt{2^n}} |x_0\rangle = \frac{1}{2^n\sqrt{2^n}} \left((2^n - 4) \sum_{\substack{x=0 \\ x\neq x_0}}^{2^n-1} |x\rangle + (3 \cdot 2^n - 4) |x_0\rangle \right)$$

Noteu que: $\langle\psi|x_0\rangle = \frac{1}{\sqrt{2^n}}$ i $\langle\psi|\psi\rangle = \frac{\sqrt{2^n}}{\sqrt{2^n}} = 1.$

Així, en cada iteració, la probabilitat de trobar l'element va incrementant. En la figura 3.16 es pot veure gràficament el comportament de la iteració de Grover. Es tracta d'una representació de l'espai solució $|\beta\rangle = |x_0\rangle$ i l'hiperpla ortogonal a la solució, $|\alpha\rangle$.

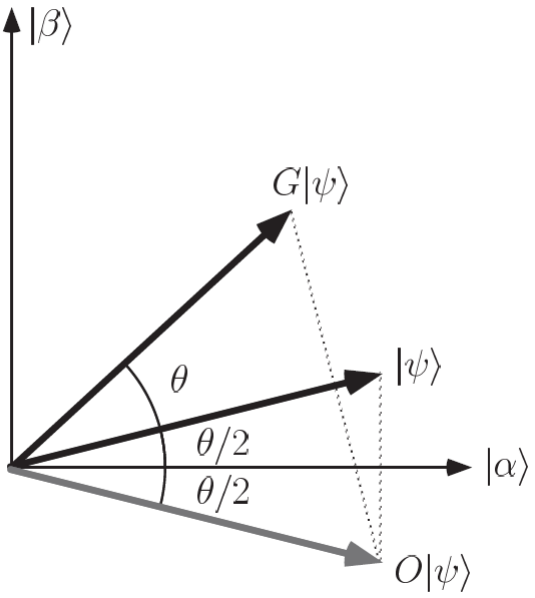

Figura 3.16: Iteració de Grover

Amb aquesta representació queda justificat el nom de l'operador inversor respecte la mitjana. Cada iteració genera una reflexió respecte l'eix $|\alpha\rangle$ mitjançant l'oracle i una segona reflexió respecte l'eix $|\psi\rangle$ deguda a l'operador $[2\,|\psi\rangle\langle\psi| - \mathbb{I}]$, de manera que l'estat del registre es va desplaçant cap a l'eix solució, $|\beta\rangle$.

Com s'ha comentat, si el nombre d'iteracions és excessiu, l'algorisme pot superar l'eix solució i allunyar-se'n amb cada nova iteració. Es pot calcular que el nombre d'iteracions òptim és el mostrat en (3.34):

$$N_{iter} = \frac{1}{4} \left(\frac{\pi}{\arcsin \sqrt{N}^{-1}} - 2 \right) \xrightarrow[N\gg 1]{} \approx \frac{\pi\sqrt{N}}{4} \tag{3.34}$$

CAPÍTOL

4

BLOC TEÒRIC III: MODELS MATEMÀTICS

4.1 Introducció

En aquest apartat s'inclou un recull de definicions i conceptes matemàtics que poden resultar molt útils per entendre correctament els algorismes anteriors (en particular l'algorisme de Shor, que presenta una forta base aritmètica). A més, es fa un breu repàs de la criptografia de clau pública basada en RSA per tal de posar de manifest la gran importància que representa l'algorisme de Shor en aquest àmbit.

D'altra banda, es fa una breu introducció a la teoria de grafs, una branca de les matemàtiques que cada vegada s'estén a més camps i que ha donat lloc al concepte de *quantum graph*. La segona de les aplicacions *Quantum Graph Tool* se centra exclusivament a presentar aquests objectes matemàtics i les bases per a la seva descripció. El seu interès en aquest treball radica en el fet que la computació quàntica pot ser la solució per resoldre alguns dels problemes oberts d'aquesta branca.

4.2 Aritmètica modular, ordres i phi d'Euler

En l'algorisme de Shor, s'ha reduït el problema de la factorització al càlcul del període de la funció exponencial. Això es correspon a l'ordre dels elements invertibles del grup $\mathbb{Z}/N$.

$$\mathbb{Z}/N = \{0, 1, 2, ..., N-2, N-1\}$$

En aquest grup, els elements invertibles, x, són aquells que compleixen precisament que $gcd(x, N) = 1$, condició que compleixen els y_0 aleatoris de l'algorisme. És a dir:

$$x \in (\mathbb{Z}/N)^* \iff gcd(x, N) = 1$$

On $x \in (\mathbb{Z}/N)^*$ denota el conjunt dels elements invertibles de $\mathbb{Z}/N$.

Com que el registre de sortida ha d'emmagatzemar els ordres, cal saber quin valor poden prendre. A partir del teorema següent, es pot donar una cota d'aquest ordre (període) per determinar els qubits necessaris del registre de sortida.

Teorema 1. (Teorema d'Euler.) Sigui $N \in \mathbb{Z}$, $N > 1$. $\forall a \in \mathbb{Z}$ tal que $gcd(a, N) = 1$, tenim:

$$a^{\varphi(N)} \equiv 1 \; mod \; N$$

On $\varphi(N) \triangleq |\{z \in \mathbb{Z} : 1 \leq z \leq N, gcd(z, N) = 1\}|$ correspon a la funció phi d'Euler.

La funció $\varphi(N)$ compta, per definició, el nombre d'elements invertibles del grup $\mathbb{Z}/N$; per tant, $\varphi(N) < N$, ja que $\mathbb{Z}/N$ té N elements (i el 0 no és invertible).

Com que l'ordre d'un element a correspon al mínim k tal que $a^k \equiv 1 \; mod \; N$, tenim que $k \leq \varphi(N) < N$. Per tant, amb $log_2(N)$ qubits a la sortida es poden codificar tots els períodes possibles de la funció exponencial. Aquest resultat justifica el nombre de qubits utilitzats en l'algorisme.

A més, la funció phi d'Euler permet estudiar el comportament de l'ordre:

Proposició 1. Sigui $N \in \mathbb{Z}$, $N > 1$. L'ordre de qualsevol element $x \in (\mathbb{Z}/N)^*$ és un divisor de $\varphi(N)$.

Perquè la proposició anterior sigui útil per estudiar el comportament de l'ordre, ha de ser senzill calcular $\varphi(N)$ a partir d'un nombre N determinat.

Proposició 2. Sigui $N \in \mathbb{Z}$, $N > 1$.

$$N \text{ primer} \iff \varphi(N) = N - 1$$

La demostració d'aquesta proposició és immediata per la definició de nombre primer, ja que no pot tenir factors en comú amb cap dels nombres menors que ell. A més, cal recordar que el 0 no és invertible, ja que $gcd(0, N) = N$ per a quasevol $N \in \mathbb{N}$.

La proposició següent demostra que calcular $\varphi(N)$ quan N és potència de nombres primers també resulta molt senzill.

Proposició 3. Per a tot $v \in \mathbb{Z}$, v positiu i per a tot P primer:

$$\varphi(P^v) = P^{v-1}(P-1)$$

Finalment, la propietat multiplicativa de l'última proposició permet calcular, a partir de la descomposició d'un enter en factors primers, la phi associada a l'enter.

Proposició 4. Si $M, N \in \mathbb{Z}$, $M, N > 1$, i $gcd(M, N) = 1$, aleshores:

$$\varphi(MN) = \varphi(M)\varphi(N)$$

En les taules 4.1 i 4.2 es mostren els valors de $\varphi(N)$ i els ordres dels elements invertibles de $\mathbb{Z}/N$ per a alguns enters N.[1]

En la taula següent es mostren nombres que poden ser factoritzats mitjançant l'algorisme de Shor. Això és, alguns enters $N > 0$ que no són potències de nombres primers.

N	$\varphi(N)$	Ordres	Elements
12	4	1, 2	1, 5, 7, 11
14	6	1, 2, 3, 6	1, 3, 5, 9, 11, 13
15	8	1, 2, 4	1, 2, 4, 7, 8, 11, 13, 14
18	6	1, 2, 3, 6	1, 5, 7, 11, 13, 17
20	8	1, 2, 4	1, 3, 7, 9, 11, 13, 17, 19
21	12	1, 2, 3, 6	1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20
22	10	1, 2, 5, 10	1, 3, 5, 7, 9, 13, 15, 17, 19, 21
24	8	1, 2	1, 5, 7, 11, 13, 17, 19, 23
26	12	1, 2, 3, 4, 6, 12	1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25
28	12	1, 2, 3, 6	1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25, 27
30	8	1, 2, 4	1, 7, 11, 13, 17, 19, 23, 29

Taula 4.1: Llista de l'ordre dels elements de $(\mathbb{Z}/N)^*$ pels $N \neq P^v$.

D'altra banda, els nombres parells no constitueixen bons candidats per l'algorisme de Shor, ja que resulta molt més simple dividir per dos que executar tot l'algorisme per trobar un possible factor.

[1] Aquesta llista correspon a la primera meitat dels nombres que es poden factoritzar utilitzant l'aplicació QGT 04.

En la taula següent es mostren alguns dels enters que no poden ser factoritzats per l'algorisme.

N	$\varphi(N)$	Ordres	Elements
13	12	1, 2, 3, 4, 6, 12	1,...,12
$16 = 2^4$	8	1, 2, 4	{1,...,15} - $\{2k, \ k \in \mathbb{N}\}$
17	16	1, 2, 4, 8, 16	1,...,16
19	18	1, 2, 3, 6, 9, 18	1,...,18
23	22	1, 2, 11, 22	1,...,22
$25 = 5^2$	20	1, 2, 4, 5, 10, 20	{1,...,24} - $\{5k, \ k \in \mathbb{N}\}$
$27 = 3^3$	18	1, 2, 3, 6, 9, 18	{1,...,26} - $\{3k, \ k \in \mathbb{N}\}$
29	28	1, 2, 4, 7, 14, 28	1,...,28
31	30	1, 2, 3, 5, 6, 10, 15, 30	1,...,30
$32 = 2^5$	16	1, 2, 4, 8	{1,...,31} - $\{2k, \ k \in \mathbb{N}\}$

Taula 4.2: Llista de l'ordre dels elements de $(\mathbb{Z}/P^v)^*$ amb P primer i $v > 0$.

És evident que per als nombres primers, l'algorisme només pot retornar factors trivials (ja que no n'hi ha més), però el que no resulta tan evident és que per a les potències de primers l'algorisme falli sempre (per períodes imparells o retornant els factors trivials). El motiu pel qual l'algorisme retorna els factors trivials és l'existència d'arrels primitives en $\mathbb{Z}/P^v$, això és, elements invertibles que tenen ordre igual a $\varphi(P^v)$.[2] Com a conseqüència que existeixin arrels primitives, els y_0 de l'algorisme compleixen:

$$y_o^{\frac{r}{2}} \equiv -1 \equiv (P^v - 1) \ mod \ P^v \tag{4.1}$$

Això es deu al fet que l'existència d'arrels primitives implica que el conjunt dels invertibles és un grup cíclic generat per l'element arrel primitiva. Com que en un grup cíclic d'ordre n l'unic element d'ordre 2 és el -1 $(mod\ n)$, l'algorisme sempre retorna aquest element; per tant, els candidats que se'n dedueixen són els trivials:

$$\begin{aligned} p_a &= gcd(y_0^{\frac{r}{2}} + 1, P^v) = gcd((P^v - 1) + 1, P^v) = P^v \\ p_b &= gcd(y_0^{\frac{r}{2}} - 1, P^v) = gcd((P^v - 1) - 1, P^v) = 1 \end{aligned} \tag{4.2}$$

A més, resulta interessant assenyalar que els enters de la forma $2P^v$, amb P primer imparell, també tenen arrels primitives. En aquest cas, però, un dels candidats retornat és el 2, ja que, com que el nombre és parell, $gcd(-2, 2P^v) = 2$. Com s'ha comentat, no té sentit aplicar l'algorisme de Shor per factoritzar un nombre parell.

Finalment, per tancar aquest apartat sobre aritmètica modular i en relació amb el problema dels elements $y_0 \equiv -1 \ (mod \ N)$, si l'element y_0 escollit és precisament $N - 1$ —pel mateix motiu que en (4.2)—, l'algorisme de Shor retorna els factors trivials. Per tal d'evitar aquest problema, es pot acotar el rang de valors de y_0, és a dir:

$$2 \leq y_o \leq N - 2 \tag{4.3}$$

[2] Per a més informació sobre arrels primitives, consulteu [12].

4.3 Algorisme de fraccions contínues

Com s'ha vist en el capítol anterior, la part clàssica de l'algorisme de Shor intenta aproximar successivament la fracció $\frac{l_j}{2^n}$ per tal de determinar el període de la funció exponencial. En aquest apartat, es fa un petit resum de la teoria de les fraccions contínues i es presenta l'algorisme utilitzat en el QGT 04 per trobar el període.

Donat un nombre racional positiu ξ, és possible expressar-lo de la manera següent:

$$\xi = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \cfrac{1}{a_k + \cfrac{1}{\cdots + \cfrac{1}{a_N}}}}}} = [a_0, a_1, a_2, \ldots, a_k, \ldots, a_N] \tag{4.4}$$

Aquesta expressió s'anomena *fracció contínua* i la notació $[a_0, \ldots, a_N]$ correspon a una manera compacta de representar-la.

Si es prenen els primers termes de la fracció contínua s'obté una aproximació del racional:

$$\xi \approx \xi_k = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \cfrac{1}{a_k}}}} = [a_0, a_1, a_2, \ldots, a_k] \tag{4.5}$$

On $0 \leq k \leq N$. Aquesta aproximació, ξ_k, s'anomena *convergent k-èssim* de la fracció contínua. A més, com més gran és k, més bé s'aproxima el racional ξ.

Teorema 2. Sigui x un nombre racional i $\frac{p}{q}$ una fracció irreductible tal que:

$$\left| x - \frac{p}{q} \right| \leq \frac{1}{2q^2}$$

aleshores, la fracció $\frac{p}{q}$ és un convergent de la fracció contínua de x.

Aquest teorema és el que demostra que es pot calcular el període de la funció exponencial a partir de la mesura del registre d'entrada a l'algorisme de Shor (veure 3.31, pas 7). Els valors mesurats en l'algorisme són els l_j de la forma:

$$l_j = j\frac{2^n}{r} + \Delta_j, \quad |\Delta_j| \leq \frac{1}{2} \implies \frac{l_j}{2^n} = \frac{j}{r} + \frac{\Delta_j}{2^n} \implies \frac{l_j}{2^n} - \frac{j}{r} = \frac{\Delta_j}{2^n} \leq \frac{1}{2^{n+1}}$$

$$\Longrightarrow \quad \left|\frac{l_j}{2^n} - \frac{j}{r}\right| \leq \frac{1}{2 \cdot 2^n} < \frac{1}{2 \cdot r^2}$$

De manera que es compleixen les hipòtesis del teorema 2 ($x = \frac{l_j}{2^n}$). Per tant, si es van trobant els convergents de $\frac{l_j}{2^n}$, algun dels denominadors coincidirà amb el període. Només queda, doncs, presentar un algorisme per calcular els convergents de x:

$$\text{Primers convergents: } \begin{cases} p_0 = a_0 = [x], & p_1 = a_1 \cdot a_0 + 1 \\ q_0 = 1, & q_1 = a_1 \end{cases} \tag{4.6}$$

$$\text{Algorisme recursiu: } \begin{cases} p_k = a_k \cdot p_{k-1} + p_{k-2} \\ q_k = a_k \cdot q_{k-1} + q_{k-2} \end{cases} \tag{4.7}$$

On els termes a_k també es poden generar recursivament a partir de l'algorisme següent:

$$\begin{cases} a_0 = [x] \\ \zeta_0 = x - a_0 \end{cases} \quad \begin{cases} a_{k+1} = [\zeta_k] \\ \zeta_{k+1} = \dfrac{1}{\zeta_k} - a_{k+1} \end{cases} \tag{4.8}$$

En el QGT 04 s'ha programat una funció en la qual es generen recursivament tots aquests valors. En cada iteració es comprova el q_k obtingut, comprovació que consisteix a determinar si $y_0^{q_k} \equiv 1 \; mod \; N$; en cas afirmatiu, q_k correspon al període buscat.

4.4 Criptografia basada en RSA

En un criptosistema de clau pública, a diferència dels de clau privada, tot i disposar de la clau xifradora, no és possible trobar la clau desxifradora sense un temps computacional prohibitiu.

La criptografia de clau pública basada en RSA (Rivest, Shamir i Adleman), com s'ha comentat, es basa en la dificultat de factoritzar els enters grans. El funcionament d'aquest mètode de xifratge segueix l'esquema següent:

Pas 1: s'escullen p i q

Cada usuari escull dos primers, p i q, molt grans i no gaire propers (p. e. de 512 bits) i en realitza el producte $n = pq$.

Pas 2: es calcula $\varphi(n)$

Com s'ha vist en l'apartat anterior, es pot calcular la funció $\varphi(n)$ d'Euler a partir de la descomposició de l'enter en factors primers:

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1) = n - p - q + 1 \tag{4.9}$$

Pas 3: s'escull e

A continuació, s'escull un enter aleatori, e, $1 < e < \varphi(n)$, tal que $gcd(e, \varphi(n)) = 1$.

Pas 4: es calcula l'invers d

Es calcula l'enter d, que correspon a la clau de desxifratge:

$$d \equiv e^{-1} (mod\ \varphi(n)) \tag{4.10}$$

Es fa pública la clau de xifratge $K_X = (n, e)$ i s'amaga la clau de desxifratge $K_D = d$.

Pas 5: funció de xifratge

L'usuari emissor aplica la funció xifradora al text en clar (s'aplica a cada unitat de missatge per generar el text xifrat):

$$\begin{aligned} f : \mathbb{Z}/n &\longrightarrow \mathbb{Z}/n \\ x &\longrightarrow x^e\ (mod\ n) \end{aligned}$$

Pas 6: funció de desxifratge

L'usuari receptor aplica la funció desxifradora a cada unitat del text xifrat per tal de recuperar el text en clar:

$$\begin{aligned} f^{-1} : \mathbb{Z}/n &\longrightarrow \mathbb{Z}/n \\ y &\longrightarrow y^d (mod\ n) \end{aligned}$$

Aquestes aplicaciones són inverses, ja que, pel teorema d'Euler (Teorema 1):

$$ed \equiv 1\ (mod\ \varphi(n)) \implies ed = 1 + k\varphi(n) \implies x^{ed} \equiv x^{1+k\cdot\varphi(n)} \equiv x\ (mod\ n) \tag{4.11}$$

A més, si p i q s'han escollit bé,[3] la factorització de n (per tal de calcular $\varphi(n)$ i d) no es pot realitzar, en un temps acceptable, sense l'ordinador quàntic i l'algorisme de Shor.

4.5 Teoria de grafs

La teoria de grafs és una branca de les matemàtiques relativament recent que es troba molt present en la informàtica i les telecomunicacions. El ventall d'aplicacions d'aquesta disciplina és molt ampli, en part, a causa de la flexibilitat que ofereix el seu element matemàtic: el graf. Tot i el ventall d'aplicacions dels grafs, la incapacitat dels ordinadors actuals de resoldre alguns dels problemes de la disciplina en limita parcialment el progrés. Per aquest motiu, es vol seguir estudiant, com a continuació d'aquest projecte, la possiblitat de modelar els grafs mitjançant la computació quàntica.

[3] Si els p i q que s'escullen són propers, es pot aplicar el mètode de Fermat per factoritzar n.

En aquest apartat s'introdueixen breument els grafs i la manera de codificar-los per tal d'entendre els conceptes que apareixen en l'aplicació QGT 02. A banda dels grafs bàsics, la majoria de conceptes són descriptius i només fan referència a la nomenclatura utilitzada.

Definició 1. Un graf $G = (V, E)$ és un parell ordenat, V (conjunt de *nodes* o *vèrtexs*) i E (conjunt d'*arestes*), de manera que els elements de V són punts i els elements de E són arestes i cada aresta de E relaciona (o incideix en) un parell de nodes de V.

A més, el concepte anterior es completa amb les definicions següents:

- L'*ordre* d'un graf correspon al cardinal de V (nombre de nodes).
- Si una de les arestes relaciona un node amb ell mateix, l'aresta s'anomena *llaç*.
- Si diverses arestes relacionen el mateix parell de nodes, s'anomenen *arestes múltiples*.
- Si un graf no té llaços ni arestes múltiples, s'anomena *graf simple*.
- El *grau* d'un node és el nombre d'arestes que hi incideixen.[4]
- La *successió de graus* de G és la successió, ordenada de manera creixent, dels graus dels nodes de G.
- La *ponderació* d'un graf consisteix a assignar un valor (o pes) als elements de E. De manera intuïtiva es pot veure com la dificultat de passar d'un node a un altre.

En la figura 4.1 es pot veure un exemple per entendre millor aquestes definicions. Es tracta d'un graf que no és simple i que té set arestes i quatre nodes.

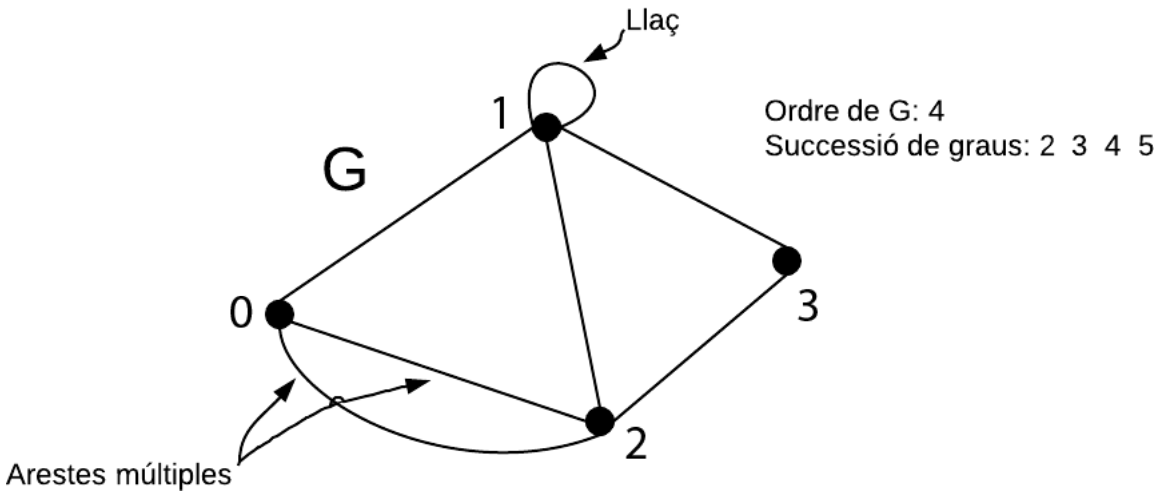

Figura 4.1: Graf d'ordre 4

[4] Si la aresta és un llaç, es compta dues vegades.

A continuació, es presenten les famílies bàsiques de grafs:

Graf complet d'ordre n, K_n

El graf complet, K_n, és un graf simple amb n nodes d'ordre $n-1$. Això implica que, per cada parell de nodes del graf, existeix una i només una aresta que els relaciona. K_n és el graf simple amb més arestes.

$$\text{nombre de nodes} = n$$
$$\text{nombre d'arestes} = \frac{n(n-1)}{2}$$

Graf elemental o nul d'ordre n, N_n

El graf elemental, N_n, és un graf simple amb n nodes aïllats. Això implica que no hi ha arestes i que tots els nodes tenen grau 0.

$$\text{nombre de nodes} = n$$
$$\text{nombre d'arestes} = 0$$

Graf cíclic d'ordre n, C_n

El graf cíclic, C_n ($n > 2$), és un graf simple amb n nodes d'ordre 2 i n arestes. Això implica que la topologia del graf es correspon amb un cicle (anell).

$$\text{nombre de nodes} = n$$
$$\text{nombre d'arestes} = n$$

Definició 2. Un graf $G = (V, E)$ és *bipartit* si existeixen dos subconjunts no buits del conjunt dels nodes $A, B \subset V$ tals que:

$$A \cup B = V \qquad A \cap B = \emptyset \qquad \text{(partició no trivial de V)}$$

de manera que, $\forall e \in E$, l'aresta e incideix en un node de A i en un node de B.

Graf bipartit complet d'ordres n i m, $K_{n,m}$

El graf bipartit complet, $K_{n,m}$, és un graf simple bipartit amb el nombre màxim d'arestes. Això implica que, si A i B conformen la partició dels nodes, $\forall a \in A$ i $\forall b \in B$ existeix una aresta que incideix en a i b. On n i m corresponen al nombre de nodes dels subconjunts A i B.

$$\text{nombre de nodes} = n + m$$
$$\text{nombre d'arestes} = n \cdot m$$

A continuació, es defineixen els dos modes de codificació dels grafs que s'han programat en l'aplicació QGT 02:

Definició 3. Donat un graf $G = (V, E)$, amb els nodes etiquetats $(0, \ldots, \#V - 1)$, es defineix la *matriu d'adjacència* com la matriu $A(G) = (a_{xy})$ on:

$$\begin{array}{ll} a_{xy} = \{\text{nombre d'arestes entre els nodes } x \text{ i } y\}, & \text{si } x \neq y \\ a_{xy} \equiv a_{xx} \equiv a_{yy} = 2 \ \cdot \{\text{nombre de llaços en el node } x\}, & \text{si } x = y \end{array}$$

De manera que $A(G)$ és una matriu quadrada i simètrica.

Per exemple, una matriu d'adjacència pel graf de la figura 4.1 és la següent:

$$A(G) = \begin{pmatrix} 0 & 1 & 2 & 0 \\ 1 & 2 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{4.12}$$

Així doncs, a partir de la matriu d'adjacència d'un graf, es pot veure fàcilment si aquest és simple. Només cal veure que els elements són zeros o uns i que la diagonal és nula.

$$G \text{ simple} \iff a_{xy} \in \{0, 1\} \ \text{ i } \ a_{xx} = 0 \tag{4.13}$$

A més, tota matriu simètrica de nombres enters no negatius amb parells a la diagonal és la matriu d'adjacència d'algun graf.

El mode de codificació següent és més compacte (eficient en espai), però més difícil de llegir i d'utilitzar:

Definició 4. Donat un graf $G = (V, E)$ amb els nodes etiquetats $(0, \ldots, \#V - 1)$, es defineix una *llista d'adjacència* com una llista per a cada node $k \in V$, en què figuren els nodes que comparteixen aresta amb k. Si hi ha n arestes múltiples entre el node k i el node l, en la llista de k el node l apareix n vegades.

Per exemple, una llista d'adjacència pel graf de la figura 4.1 és la següent:

$$\begin{array}{ll} 0: & 1\ 2\ 2 \\ 1: & 0\ 1\ 2\ 3 \\ 2: & 0\ 0\ 1\ 3 \\ 3: & 1\ 2 \end{array} \tag{4.14}$$

Tot i que s'han ordenat per facilitar-ne la lectura, les llistes de cada node no cal que estiguin ordenades. Una altra llista possible pel mateix graf és la següent:

$$\begin{array}{ll} 0: & 2\ 1\ 2 \\ 1: & 3\ 1\ 0\ 2 \\ 2: & 1\ 3\ 0\ 0 \\ 3: & 1\ 2 \end{array} \tag{4.15}$$

Les listes d'adjacència permeten estalviar molta memòria si el graf no té gaires arestes, ja que no cal emmagatzemar els zeros com en el cas de les matrius d'adjacència. Per exemple, un graf G_{10} amb deu nodes i una sola aresta entre els nodes 1 i 2 presenta una llista d'adjacència amb 100 elements (200 bytes):

$$A(G_{10}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.16}$$

Mentre que en la llista d'adjacència només hi ha 2 elements (4 bytes):

$$\begin{array}{ll} 0: & \\ 1: & 2 \\ 2: & 1 \\ 3: & \\ \vdots & \\ 9: & \end{array} \tag{4.17}$$

Més endavant, en el capítol dedicat al QGT 02, es veuen exemples d'aquestes codificacions que ajuden a entendre aquests modes de codificació.

CAPÍTOL

5

BLOC TEÒRIC IV: COMPUTACIÓ ACUMULATIVA

5.1 Funcions acumulatives

En el capítol 3 s'han vist alguns algorismes quàntics que exploten les propietats de la mecànica quàntica: la superposició, el paral·lelisme, la interferència... Aquestes noves propietats representen un marc idoni per realitzar nous plantejaments dels problemes i desenvolupar noves maneres de computar.

Amb aquesta idea, es defineixen les *funcions acumulatives* que permeten un nou tipus de tractament de les variables quàntiques $|x\rangle$.

Definició 5. Siguin X un conjunt finit, $\mathcal{P}(X)$ el conjunt de les parts de X i f una funció entre conjunts tal que:

$$\begin{aligned} f : \mathcal{P}(X) &\longrightarrow \mathcal{P}(X) \\ X \supseteq A &\longrightarrow f(A) \end{aligned}$$

Direm que f és una *funció acumulativa* si $\forall x \in X$, $x \in f(\{x\})$.

Aquesta definició pot semblar molt abstracta si no es relaciona correctament amb els conceptes d'estat quàntic i superposició. D'una banda, s'ha de fer una correspondència

unívoca entre els elements de X i els estats purs del registre de qubits (codificació); de l'altra, s'ha d'entendre tot estat (5.1) com un element de $\mathcal{P}(X)$ (interpretació).

$$|x\rangle = \sum_i \alpha_i |x_i\rangle \tag{5.1}$$

La interpretació més natural consisteix a suposar que un element $|x_i\rangle$ pertany al conjunt $|x\rangle$ si el coeficient $\alpha_i \neq 0$. Aquestes relacions permeten definir les funcions acumulatives mitjançant el llenguatge de la computació quàntica.

Definició 6. Siguin X un conjunt finit amb cardinal $|X| = n \in \mathbb{N}$ i f una funció acumulativa en $\mathcal{P}(X)$. Direm que f és *quànticament realitzable* si existeixen una codificació mitjançant qubits dels elements del conjunt X, $(|x_1\rangle, |x_2\rangle, \ldots, |x_n\rangle)$, i una matriu $\mathbb{A}_f \in U_{\mathbb{C}}(n)$, de manera que $\mathbb{A}_f |x_i\rangle = f(|x_i\rangle)$, $\forall i < n$.

Amb aquesta definició es pot fer una descripció més pràctica del concepte de funció acumulativa, i. e. a partir dels fenòmens de la superposició i l'entrellaçament, es desenvoluparan famílies de portes quàntiques, associades a endomorfismes,[1] que a la sortida presentaran una superposició ponderada dels estats $|x\rangle$ i imatge de $|x\rangle$ per una funció. En la figura 5.1 es pot veure una possible representació d'una funció acumulativa. S'adoptaran els símbols: $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, $\mathbb{D}$... amb subíndexs f, g, h... per designar les matrius associades a les funcions acumulatives.

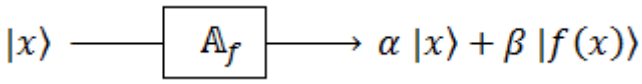

Figura 5.1: Representació del circuit associat a una funció acumulativa. Els coeficients α i β han de ser no nuls.

S'utiltzaran aquestes funcions acumulatives per convertir problemes de complexitat exponencial en problemes de complexitat lineal mitjançant el paral·lelisme quàntic.

A continuació, es veuen alguns exemples de funcions acumulatives quànticament realitzables amb la descripció de les matrius unitàries que les implementen. L'interès d'aquest tipus de computació és plantejar maneres d'aprofitar les propietats de la mecànica quàntica per resoldre problemes modelats com a grups o com a grafs.

5.2 *Swap* acumulatiu de dos qubits

El primer exemple correspon a la funció acumulativa *swap* de dos qubits. Es tracta d'una extensió del *swap* clàssic de dos bits que es pot modelar com es mostra en la taula següent.

[1] És necessari que les funcions siguin endomorfismes perquè la superposició de x i $f(x)$ tingui sentit.

Bits entrada	Bits sortida	Estat entrada	Estat sortida
00	00	$\|00\rangle$	$\|00\rangle$
01	10	$\|01\rangle$	$\alpha_1 \|01\rangle + \beta_1 \|10\rangle$
10	01	$\|10\rangle$	$\alpha_2 \|10\rangle + \beta_2 \|01\rangle$
11	11	$\|11\rangle$	$\|11\rangle$

Taula 5.1: *Swap* clàssic i *swap* acumulatiu de dos qubits.

Es pot veure que el comportament buscat és el d'acumular a la sortida la superposició de l'entrada i la sortida del *swap* clàssic. És a dir:

$$\begin{cases} |0\rangle \equiv |00\rangle \longrightarrow \alpha_0 |00\rangle + \beta_0 |00\rangle \equiv |0\rangle \\ |1\rangle \equiv |01\rangle \longrightarrow \alpha_1 |01\rangle + \beta_1 |10\rangle \equiv \alpha_1 |1\rangle + \beta_1 |2\rangle \\ |2\rangle \equiv |10\rangle \longrightarrow \alpha_2 |10\rangle + \beta_2 |01\rangle \equiv \alpha_2 |2\rangle + \beta_2 |1\rangle \\ |3\rangle \equiv |11\rangle \longrightarrow \alpha_3 |11\rangle + \beta_3 |11\rangle \equiv |3\rangle \end{cases}$$

Per tant, en mesurar l'estat de sortida, es pot trobar l'entrada o la sortida de la funció *swap* clàssica. Per tal de realitzar aquesta funció es pot partir de la porta Hadamard d'un qubit i aplicar la identitat als estats que no es modifiquen. Així, en (5.2), es pot veure la matriu $\mathbb{A}_S$ de la funció acumulativa *swap*.

$$\mathbb{A}_S = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix} \tag{5.2}$$

És immediat comprovar que la matriu és unitària i que el comportament correspon al desitjat. S'ha obtingut, doncs, la primera funció acumulativa, el *swap* de dos qubits, i és quànticament realitzable.

5.3 *Swap* acumulatiu de tres qubits

En aquest apartat s'intenta estendre el *swap* acumulatiu pel cas en què el registre d'entrada estigui constituït per tres qubits. Aquest *swap* es pot entendre com el resultat de quasevol permutació dels bits, per exemple:[2]

$$6 \equiv 110 \xrightarrow{swap} \begin{cases} 101 \longrightarrow \text{si s'ha fet un } swap \text{ dels dos bits menys significatius} \\ 011 \longrightarrow \text{si s'ha fet un } swap \text{ dels bits dels extrems} \\ 110 \longrightarrow \text{si s'ha fet un } swap \text{ dels dos bits més significatius} \end{cases}$$

Així, el *swap* objectiu ha de contemplar totes les possibles sortides que tenen el mateix nombre de zeros i uns que l'entrada.

[2] Això correspon a totes les permutacions possibles si només es té en compte la sortida lògica i, a més, l'últim *swap* és equivalent a $\mathbb{I}$ (identitat).

Per tant, el que es vol modelar és una funció acumulativa com la que es mostra en la taula següent:

Bits entrada	Bits sortida	Estat entrada	Estat sortida
000	000	$\|000\rangle$	$\|000\rangle$
001	001, 010, 100	$\|001\rangle$	$\alpha_1 \|001\rangle + \beta_1 \|010\rangle + \gamma_1 \|100\rangle$
010	010, 001, 100	$\|010\rangle$	$\alpha_2 \|010\rangle + \beta_2 \|100\rangle + \gamma_2 \|001\rangle$
011	011, 110, 101	$\|011\rangle$	$\alpha_3 \|011\rangle + \beta_3 \|110\rangle + \gamma_3 \|101\rangle$
100	100, 001, 010	$\|100\rangle$	$\alpha_4 \|100\rangle + \beta_4 \|001\rangle + \gamma_4 \|010\rangle$
101	101, 011, 110	$\|101\rangle$	$\alpha_5 \|101\rangle + \beta_5 \|011\rangle + \gamma_5 \|110\rangle$
110	110, 101, 011	$\|110\rangle$	$\alpha_6 \|110\rangle + \beta_6 \|101\rangle + \gamma_6 \|011\rangle$
111	111	$\|111\rangle$	$\|111\rangle$

Taula 5.2: *Swap* clàssic i *swap* acumulatiu de tres qubits.

De manera que la matriu unitària $\mathbb{A}_{S3}$ buscada és de la forma següent:[3]

$$\mathbb{A}_{S3} = \begin{pmatrix} \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{X} & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 \\ 0 & \mathrm{X} & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{X} & 0 & \mathrm{X} & \mathrm{X} & 0 \\ 0 & \mathrm{X} & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{X} & 0 & \mathrm{X} & \mathrm{X} & 0 \\ 0 & 0 & 0 & \mathrm{X} & 0 & \mathrm{X} & \mathrm{X} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{X} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & 0 & c & 0 & 0 & 0 \\ 0 & d & e & 0 & f & 0 & 0 & 0 \\ 0 & 0 & 0 & a & 0 & b & c & 0 \\ 0 & g & h & 0 & k & 0 & 0 & 0 \\ 0 & 0 & 0 & d & 0 & e & f & 0 \\ 0 & 0 & 0 & g & 0 & h & k & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

On els coeficients a, b, c, d, e, f, g, h i $k \in \mathbb{C}\backslash\{0\}$, de manera que es necessita una matriu unitària 3×3 amb tots els elements no nuls per substituir els menors de (5.3), marcats en verd i blau. No es pot estendre el resultat del *swap* acumulatiu de dos qubits per construir aquests menors, ja que apareix un obstacle important: no es pot utilitzar la matriu de Hadamard per generar la superposició de tres estats (de fet, no es pot utilitzar si el nombre d'estats no és de la forma 2^k). Una solució parcial és utilitzar matrius que aproximin el comportament de la hipotètica Hadamard 3×3 com, per exemple:

$$H_3 = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & k \end{pmatrix} \approx \widetilde{H_3} = \frac{1}{\sqrt{3}} \begin{pmatrix} 1+i & \frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}} & 0 \\ -\frac{i}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} + i \\ \frac{1}{\sqrt{2}} & i & 1 - \frac{i}{\sqrt{2}} \end{pmatrix} \tag{5.4}$$

Que es pot demostrar fàcilment que és unitària:

$$\widetilde{H_3}^H = \frac{1}{\sqrt{3}} \begin{pmatrix} 1+i & \frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}} & 0 \\ -\frac{i}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} + i \\ \frac{1}{\sqrt{2}} & i & 1 - \frac{i}{\sqrt{2}} \end{pmatrix}^H = \frac{1}{\sqrt{3}} \begin{pmatrix} 1-i & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}} & 1 & -i \\ 0 & \frac{1}{\sqrt{2}} - i & 1 + \frac{i}{\sqrt{2}} \end{pmatrix} \tag{5.5}$$

[3] El terme X correspon a un element indeterminat no nul.

$$\widetilde{H_3}\ \widetilde{H_3}^H = \widetilde{H_3}^H \widetilde{H_3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbb{I} \tag{5.6}$$

Per tant, en substituir H_3 en la matriu (5.3), s'obté una matriu unitària[4] que s'aproxima a la funció desitjada descrita per $\mathbb{A}_{S3}$.

Un altre mètode per obtenir la funció acumulativa desitjada consisteix a ampliar el nombre de qubits de l'entrada afegint redundància al sistema. Per exemple, la matriu H_{3-ext} correspon a una Hadamard 3×3 pels qubits menys significatius (primer bloc 4×4).

$$H_{3-ext} = \begin{pmatrix} a & b & c & 0 & \mathrm{X} & 0 & 0 & 0 \\ d & e & f & 0 & \mathrm{X} & 0 & 0 & 0 \\ g & h & k & 0 & \mathrm{X} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathrm{X} & \mathrm{X} & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{X} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{X} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{X} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \tag{5.7}$$

On els elements marcats en blau corresponen als coeficients de la matriu de Hadamard de dos qubits ($H^{\otimes 2}$) i els valors en vermell, als elements que no s'utilitzen. Per tant, es pot generar la matriu del *swap* si s'utilitzen quatre qubits en lloc de tres:

$$\mathbb{A}_{S3-extended} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & 0 & c & 0 & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d & e & 0 & f & 0 & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a & 0 & b & c & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g & h & 0 & k & 0 & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d & 0 & e & f & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & h & k & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{X} & \mathrm{X} & 0 & \mathrm{X} & 0 & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{X} & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.8}$$

Amb aquesta funció la codificació dels estats s'ha complicat. Els estats començats per zero corresponen als estats desitjats, mentre que els estats començats per 1 corresponen als estats redundants de la matriu estesa.

[4] Es tracta d'una matriu unitària, ja que s'ha obtingut a partir de permutacions de files i columnes de matrius unitàries.

Per exemple, si en l'entrada s'introdueix l'element $6 \equiv 0110$, apareix a la sortida l'element $9 \equiv 1001$, que correspon a un estat redundant (comença per 1). Més concretament:

$$|0110\rangle \xrightarrow{\mathbb{A}_{S3-extended}} \frac{1}{2}(|0011\rangle - |0101\rangle - |0110\rangle + |1001\rangle) \tag{5.9}$$

En aquest punt, si es vol eliminar la redundància, es pot realitzar una mesura del qubit més significatiu. En cas de mesurar un $|0\rangle$, l'estat dels altres qubits col·lapsa a l'estat desitjat i la funció acumulativa ha funcionat satisfactòriament. En cas de mesurar un $|1\rangle$, s'obté l'estat redundant, el qual es pot utilitzar per recuperar la superposició amb la mateixa matriu $\mathbb{A}_{S3-extended}$, ja que es torna a obtenir la mateixa sortida que a (5.9).

Aquest mètode de les matrius esteses (*extended*) és arriscat per dos motius: en primer lloc, s'incrementa la complexitat del sistema i el nombre de qubits i, en segon lloc, apareixen els estats redundants, que poden complicar la connexió en cascada de funcions acumulatives. Per contra, permet una descripció simple del sistema mitjançant les matrius Hadamard habituals i introdueix un mètode per generar la matriu d'un *swap* acumulatiu general.

5.4 *Swap* acumulatiu general

En l'apartat anterior s'ha vist que l'extensió del *swap* no és trivial. En el cas general, el *swap* acumulatiu de n qubits ha de ser capaç de generar a la sortida un estat que sigui superposició de totes les combinacions amb el mateix nombre d'uns que a l'entrada (o equivalentment de zeros).

El primer problema, doncs, consisteix a calcular el nombre de combinacions que hi ha per a cada quantitat d'uns. Una manera elegant de calcular-ho és fer una anàlisi en arbre: començar pel dígit menys significatiu i plantejar les dues situacions; si és un 0, ens desplacem a la dreta i, si és un 1, a l'esquerra. Per a cada nou digit se sumen el nombre de combinacions anteriors que van complint la condició de tenir k uns a la paraula, de manera que es va generant el triangle de Pascal (o triangle de Tartaglia). És a dir:

1 qubit:									1				1										
2 qubits:								1				2				1							
3 qubits:						1				3				3				1					
4 qubits:				1				4				6				4				1			
5 qubits:			1			5				10				10				5				1	
6 qubits:		1			6			15				20				15			6				1
⋮			↘			↘			↘			↘			↘			↘			↘		
nombre d'uns:				6			5				4				3				2			1		0

De manera que, si es compara amb els casos anteriors, es pot comprovar que realment es requeria la superposició de dos estats pel cas de dos qubits, i dues superposicions de

tres estats pel cas dels tres qubits. Es pot comprovar de manera senzilla que pel cas de 4 qubits el resultat del triangle també és correcte:

Núm. d'uns	Núm. d'estats	Llista d'estats que s'han de superposar a la sortida
0	1	$\|0000\rangle$
1	4	$\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$
2	6	$\|0011\rangle$ $\|0101\rangle$ $\|0110\rangle$ $\|1001\rangle$ $\|1010\rangle$ $\|1100\rangle$
3	4	$\|1110\rangle$ $\|1101\rangle$ $\|1011\rangle$ $\|0111\rangle$
4	1	$\|1111\rangle$

Taula 5.3: Combinacions possibles pel *swap* acumulatiu de 4 qubits segons el nombre d'uns del registre d'entrada.

Un cop s'ha determinat el nombre d'estats que s'han de superposar pel cas general, es pot procedir a construir la matriu unitària encarregada d'implementar la funció acumulativa. Ja s'ha vist que per a cada superposició es necessita una matriu que es comporti com la Hadamard. Per exemple, en el cas de quatre qubits es necessiten una matriu amb tots els elements no nuls de 6×6 i dues de 4×4. No hi ha cap problema en generar les dues de 4×4, ja que es corresponen amb la matriu de Hadamard de dos qubits ($H^{\otimes 2}$). En canvi, per generar la matriu de 6×6 es torna a tenir el mateix problema que en l'apartat anterior, i les solucions proposades també són les mateixes: aproximar o treballar amb les matrius esteses. En aquest últim cas, la matriu necessària és:

$$H_{6-ext} = \begin{pmatrix}
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & 0 & 0 & \mathrm{X} & \mathrm{X} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{5.10}$$

És a dir, una matriu 16×16 que genera la superposició dels 6 elements (H) i dos estats redundants (X) per a cada superposició. Ordenant adequadament les matrius H_{6-ext} i $H^{\otimes 2}$ es pot completar la matriu unitària estesa completa. A continuació, es mostra, doncs, una matriu estesa que implementa el *swap* acumulatiu de 4 qubits de la manera indicada.[5]

[5] S'ha fet un abús de notació per optimitzar l'espai, $\mathbb{A}_{S4} \equiv \mathbb{A}_{S4-extended}$, però es tracta d'una matriu estesa que utilitza més qubits dels necessaris.

$$\mathbb{A}_{S4} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & H & H & 0 & H & 0 & 0 & 0 & H & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & H & H & 0 & H & 0 & 0 & 0 & H & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & H & H & 0 & H & 0 & 0 & 0 & H & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & H & 0 & 0 & 0 & H & 0 & H & H & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & H & H & 0 & H & 0 & 0 & 0 & H & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & H & 0 & 0 & 0 & H & 0 & H & H & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & H & 0 & H & H & 0 & 0 & H & H & 0 & H & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & H & 0 & 0 & 0 & H & 0 & H & H & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & H & 0 & 0 & 0 & H & 0 & H & H & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & X & 0 & X & X & 0 & 0 & X & X & 0 & X & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & X & 0 & X & X & 0 & 0 & X & X & 0 & X & 0 & 0 & 0 & X & X & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1
\end{pmatrix}$$

On apareixen les matrius $H^{\otimes 2}$ en verd i blau i la H_{6-ext} en taronja. $\mathbb{A}_{S4}$ és una matriu 32×32 (es necessiten 5 qubits) i els coeficients representats amb X són els estats redundants, els quals estan codificats amb un 1 en el qubit més significatiu.

Es pot procedir de manera equivalent per a qualsevol cas d'ordre superior. A més, cal recordar que en el mètode de les matrius esteses s'ha de realitzar una mesura del qubit més significatiu per eliminar la redundància.

CAPÍTOL

6

BLOC PRÀCTIC I: OBJECTIVE-C I XCODE

6.1 Introducció

En aquest bloc es presenten les eines, tecnologies i recursos utilitzats per dissenyar i programar el contingut pràctic d'aquest projecte.

El projecte *Quantum Graph Tool* és un conjunt d'aplicacions per a iPad que s'han desenvolupat sobre la plataforma per a dispositius mòbils iOS. El sistema operatiu iOS es troba en tots els dispositius mòbils Apple; en concret, en els diferents models d'iPad, iPhone i iPod disponibles en el mercat.

Per elaborar les aplicacions d'aquest projecte s'ha utilitzat l'entorn de desenvolupament Xcode versió 4.5 executat sobre un portàtil MacBook Pro (4 GB de RAM i processador de 2,3 GHz) amb el sistema operatiu Mac OS X versió 10.7.5. L'entorn de desenvolupament Xcode permet, gràcies a la incorporació dels compiladors gcc,[1] compilar molts llenguatges estàndard que poden ser executats en qualsevol sistema operatiu, per exemple, C, C++, Objective-C i Java. A més, Xcode també incorpora l'eina *Interface Builder* que facilita molt el disseny de la interfície d'usuari.

Encara que Xcode i *Interface Builder* es poden utilitzar indistintament per desenvolupar aplicacions per a iOS i OS X, com que les llibreries que s'utilitzen són diferents (sobretot les relacionades amb la interfície d'usuari), les aplicacions *Quantum Graph Tool* només es poden executar en un iPad o en un simulador d'iPad.

[1] GNU Compiler Collection, el conjunt de compiladors de lliure distribució del projecte GNU.

En aquest projecte, el llenguatge que s'ha utilitzat per programar les aplicacions és Objective-C, el qual correspon a una evolució orientada a objectes del llenguatge C, i permet que la majoria dels algorismes que s'han implementat es puguin convertir trivialment en funcions de C. De fet, es pot dir que Objective-C és un petit embolcall de C i, per aquest motiu, tots els programes escrits en C es poden compilar amb un compilador d'Objective C.

6.2 Estructura del llenguatge Objective-C

Tal com es comenta en la introducció, el llenguatge Objective-C prové del llenguatge C i està dissenyat per permetre una programació orientada a objectes. Aquest tipus de programació facilita el disseny de programes modulars i jeràrquics representats per objectes. Els objectes, al seu torn, estan constituïts per atributs, és a dir, variables que en guarden la informació, i mètodes, que són funcions o accions realitzades generalment sobre els atributs.

En Objective-C es diu que aquests objectes s'envien i reben *missatges* per executar els mètodes que tenen implementats. Aquests missatges es resolen en temps d'execució pel compilador i un d'ells pot ser ignorat si l'objecte que el rep no té implementat el mètode solicitat (en aquest cas, retorna un punter nul).

Els objectes declarats del programa són *instàncies* del seu tipus d'objecte (*classe*). Cada objecte pot tenir altres objectes com a atributs i, en aquest cas, també calen instàncies de les seves respectives classes per comunicar-se amb ells. Si un d'aquests objectes atribut només es vol utilitzar temporalment, es pot realitzar la instància en el mateix mètode que el vol fer servir. Això pot ser útil per optimitzar els recursos de memòria.[2]

Els objectes en Objective-C sempre es declaren com a punters i no consumeixen recursos fins que no se sol·licita explícitament per mitjà d'una instància. L'acció de sol·licitar memòria es realitza amb la instrucció *alloc*, la qual va acompanyada d'una crida al mètode constructor. El mètode constructor (anomenat *init* en el llenguatge Objective-C) s'encarrega d'inicialitzar els atributs dels objectes per tal de posar-los en condicions per rebre els missatges.

Finalment, per acabar aquesta breu introducció a l'estructura del llenguatge Objective-C, cal explicar el patró de disseny *Model-View-Controller* (MVC), ja que s'ha utilitzat en totes les aplicacions del projecte. El MVC és una arquitectura de software basada en la divisió del programa en tres subsistemes: el model, la vista i el controlador.

El *model* representa la implementació abstracta del programa. Dit d'una altra manera, és el conjunt de dades i variables amb què opera el sistema per tal d'executar els algorismes i les accions pertinents.

[2] Aquest recurs s'ha utilitzat habitualment en les aplicacions del projecte.

La *vista* presenta el model del programa o l'aplicació de manera adequada a l'usuari que l'ha d'utilitzar. Aquest subsistema està fortament lligat al dispositiu amb què s'executarà el codi i és el responsable de la retrocompatibilitat de les aplicacions (en relació amb aquest projecte, el fet que les aplicacions QGT siguin compatibles únicament amb els iPads).

Per últim, el subsistema *controlador* és l'encarregat de relacionar els dos subsistemes anteriors; és a dir, el controlador ha de ser capaç de generar els canvis sobre el model quan l'usuari interactua amb la vista. I, alhora, ha d'enviar constantment les dades que ha de mostrar la vista i que han estat generades pel model.

6.3 Llibreries i recursos externs

En aquest apartat es fa un resum breu de les llibreries (*frameworks*) que s'han utilitzat per desenvolupar les aplicacions QGT. La majoria d'aquestes col·leccions de classes ja estan incorporades al mateix Xcode i només s'ha d'especificar al compilador que les hi afegeixi.

UIKit

La llibreria UIKit proporciona les classes necessàries per desenvolupar i gestionar la interfície d'usuari de les aplicacions per a iOS. Incorpora la majoria de les classes dels elements bàsics (UIView, UIViewController, UILabel, UITextView...) que componen les vistes i els controladors de vista que s'han utilitzat en les aplicacions del projecte.

Aquesta llibreria, que és de lliure distribució, la proporciona Apple en la seva pàgina web per a desenvolupadors.[3]

CoreGraphics i Quartz Core

Aquestes dues llibreries es basen en el motor Quartz i permeten gestionar renderitzacions gràfiques en 2D i animacions. S'utilitzen, per exemple, en el QGT 02 per dibuixar els grafs i en els QGT 01 i QGT 04 per gestionar la creació dels arxius PDF.

Aquestes també són bàsiques, de lliure distribució i es troben en la mateixa pàgina web.

MessageUI

Amb aquesta llibreria es pot modificar un controlador de vista per generar correus electrònics des de la mateixa aplicació. Aquesta funcionalitat s'ha utilitzat en les aplicacions que generen documents PDF per enviar-los per correu sense sortir de l'aplicació.

Aquesta també és bàsica, de lliure distribució i es troba en la mateixa pàgina web que les tres anteriors.

[3] Accessible a *http://developer.apple.com/library/ios/navigation/*.

libCorePlot

Es tracta d'un conjunt de llibreries per generar gràfics per OS X i dispositius iOS. Una d'aquestes llibreries és la que s'ha utilitzat en les aplicacions QGT 03 i QGT 04 per generar les gràfiques de probabilitat dels estats.

Aquesta llibreria ha estat desenvolupada per Google i es pot descarregar gratuïtament d'una pàgina web de la companyia.[4]

A banda d'aquestes llibreries, s'han utilitzat dos simuladors d'iPad (un amb iOS 5.0 i un altre amb iOS 6.0) per testejar les aplicacions abans de carregar-les al dispositu físic de prova. Aquest dispositiu de prova ha estat un iPad de segona generació amb iOS 6.0.

[4] Accessible a *http://code.google.com/p/core-plot/*.

CAPÍTOL

7

BLOC PRÀCTIC II: QGT 01

7.1 Introducció

La primera de les aplicacions *Quantum Graph Tool* és una eina per introduir els qubits i manipular-los individualment. Cada un dels 100 qubits disponibles es pot editar per tal de modificar-ne l'estat inicial i assignar-li un nom.

Un cop inicialitzat el qubit, s'hi poden aplicar les portes Hadamard i Phase Shift general per comprovar que l'estat inicial es pot transformar en qualsevol altre. A més, mitjançant un sistema de programació basat en instruccions, es poden aplicar més transformacions específiques als qubits i configurar altres paràmetres de l'aplicació. Tota la informació de les tasques realitzades i les instruccions executades es mostra en una vista de tipus terminal. L'aplicació inclou una llista de les portes bàsiques d'un qubit i un petit manual de programació de les instruccions.

Finalment, cal remarcar que la manipulació individual dels qubits no permet estudiar el fenomen de l'entrellaçament i, en aquest sentit, l'aplicació no serveix per estudiar els algorismes quàntics que es recullen en els capítols anteriors. Es tracta únicament d'una eina introductòria per familiaritzar-se amb la nova unitat d'informació.

7.2 Descripció

Quan s'obre l'aplicació es mostren directament la pantalla principal i un menú desplegable situat a la banda esquerra del dispostiu (veure figura 7.1). En aquesta pantalla es pot veure la informació del qubit actual, un selector que permet escollir entre els 10 primers qubits, un terminal en què van apareixent les instruccions realitzades i dos botons situats

a la part inferior, *Gates* i *Codes*, els quals permeten accedir a informació addicional, tal com es detalla més endavant.

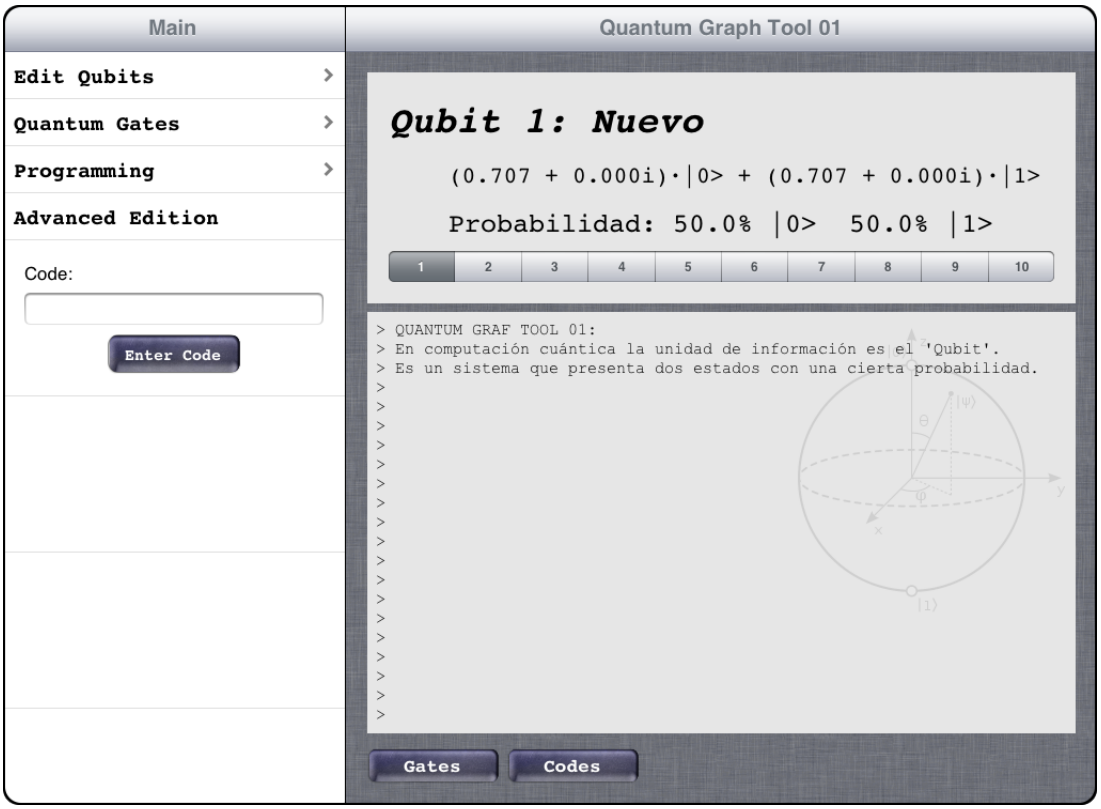

Figura 7.1: Pantalla principal del QGT 01

Per defecte, cada qubit s'anomena *Nuevo* i s'inicialitza en una superposició dels estats $|0\rangle$ i $|1\rangle$. Cada qubit mostra la probabilitat de cada estat en temps real, de manera que, en obrir l'aplicació, tots els qubits mostren un 50 % de probabilitat per a cada estat.

En el menú desplegable hi ha tres submenús i una casella per introduir-hi codi. Cada un dels tres submenús (veure figura 7.2) s'utilitza bàsicament per modificar l'estat dels qubits i canvia completament l'aspecte del menú desplegable. A continuació, es fa una breu descripció de cada un d'aquests submenús.

Edit Qubits

El submenú *Edit Qubits* s'utilitza per aplicar canvis generals al qubit actual: modificar el nom, modificar directament el mòdul de $|0\rangle$ i la fase de $|1\rangle$, modificar la fase global i simular la mesura. Totes aquestes modificacions es realitzen mitjançant botons i selectors molt simples i intuïtius.

Quantum Gates

El submenú *Quantum Gates* s'utilitza únicament per aplicar les portes Hadamard i Phase

Shift general. Tal com es descriu en el Bloc teòric II (veure figura 3.8), aquestes dues portes, combinades adequadament, permeten canviar l'estat del qubit per qualsevol dels estats possibles. De nou, la presentació d'aquest submenú és simple i intuïtiva.

Programming

El tercer submenú, *Programming*, consisteix en una llista de nou caselles, com la del menú principal, que s'utilitzen per introduir-hi petits codis de programació. El sistema de programació que s'ha desenvolupat consisteix en un joc d'instruccions de tres caràcters (o tres caràcters seguits de tres xifres) que es poden encadenar per manipular els qubits de manera automàtica. Amb això es pot simular una porta universal per inicialitzar els qubits a qualsevol estat desitjat i per tenir un control de les portes que s'han d'aplicar i l'ordre amb què s'ha de fer.

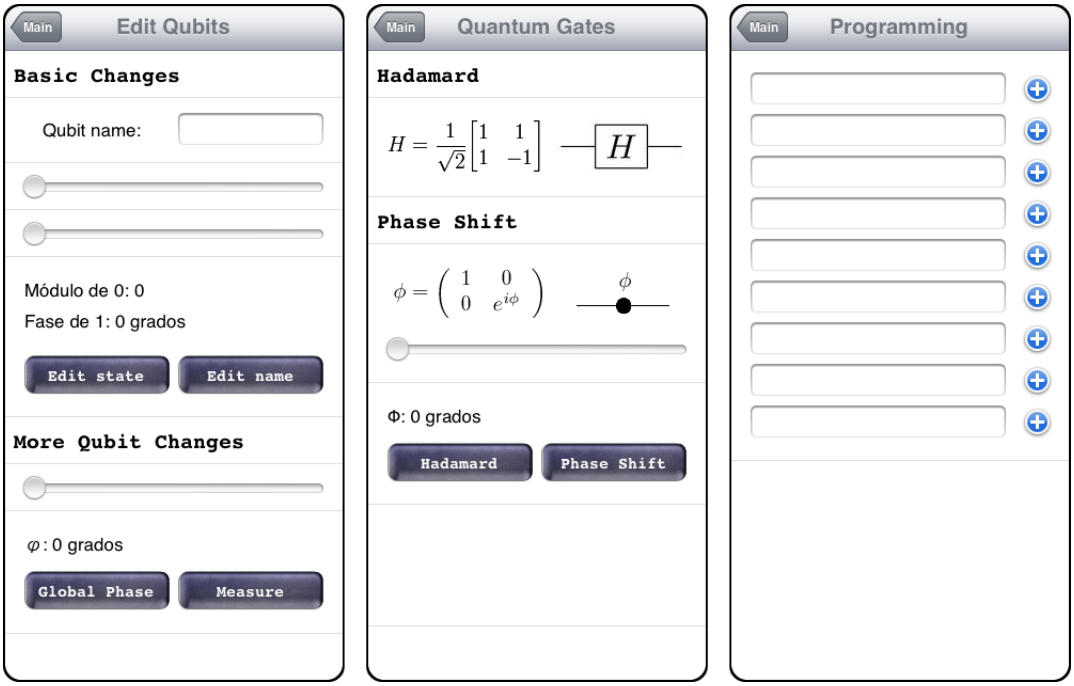

Figura 7.2: Submenús del QGT 01

Finalment, els botons *Gates* i *Codes* de la pantalla principal representen, respectivament, accessos directes a una llista de portes i a un manual d'instruccions dels codis de programació. Aquests manuals estan redactats en anglès, ja que són bàsics i fonamentalment descriptius. A més, es poden consultar simultàniament amb els submenús per tal de poder copiar les instruccions de programació i evitar errors.

7.3 Estructura

A banda de les llibreries bàsiques, el QGT 01 presenta sis classes d'objectes que segueixen l'esquema MVC. A aquest objectes s'ha d'afegir un *storyboard* (un conjunt d'objectes de

classe *ViewController* relacionats entre si) i l'objecte *main*, que s'encarrega exclusivament de crear l'objecte *AppDelegate*, que representa l'aplicació (totes les aplicacions tenen un *AppDelegate* com a nucli). Les sis classes del QGT 01 es detallen en la llista següent:

- **AppDelegate:** és el nucli de l'aplicació. Conté un objecte de classe *MasterViewController*, que es crea automàticament.
- **MasterViewController:** és el controlador del menú desplegable i dels submenús. Conté un *DetailViewController* com a atribut i tots els objectes de les vistes del menú i dels tres submenús.
- **DetailViewController:** és el controlador de la pantalla principal. Entre els seus atributs hi ha 100 objectes de classe *Qubit*, un objecte de classe *ManualViewController* i un objecte de classe *SavePDF*.
- **ManualViewController:** és un objecte encarregat de descodificar la informació dels arxius html i mostrar-la en una *ScrollView*. Concretament, descodifica la llista de portes i el manual d'instruccions que estan guardats a l'aplicació en format html.
- **Qubit:** és l'objecte que representa el qubit i les portes quàntiques.
- **SavePDF:** és l'objecte encarregat de crear, editar i enviar per correu els arxius en format PDF que resumeixen la informació del qubits.

En la figura 7.3 es pot veure una representació de l'esquema jeràrquic d'aquests objectes.

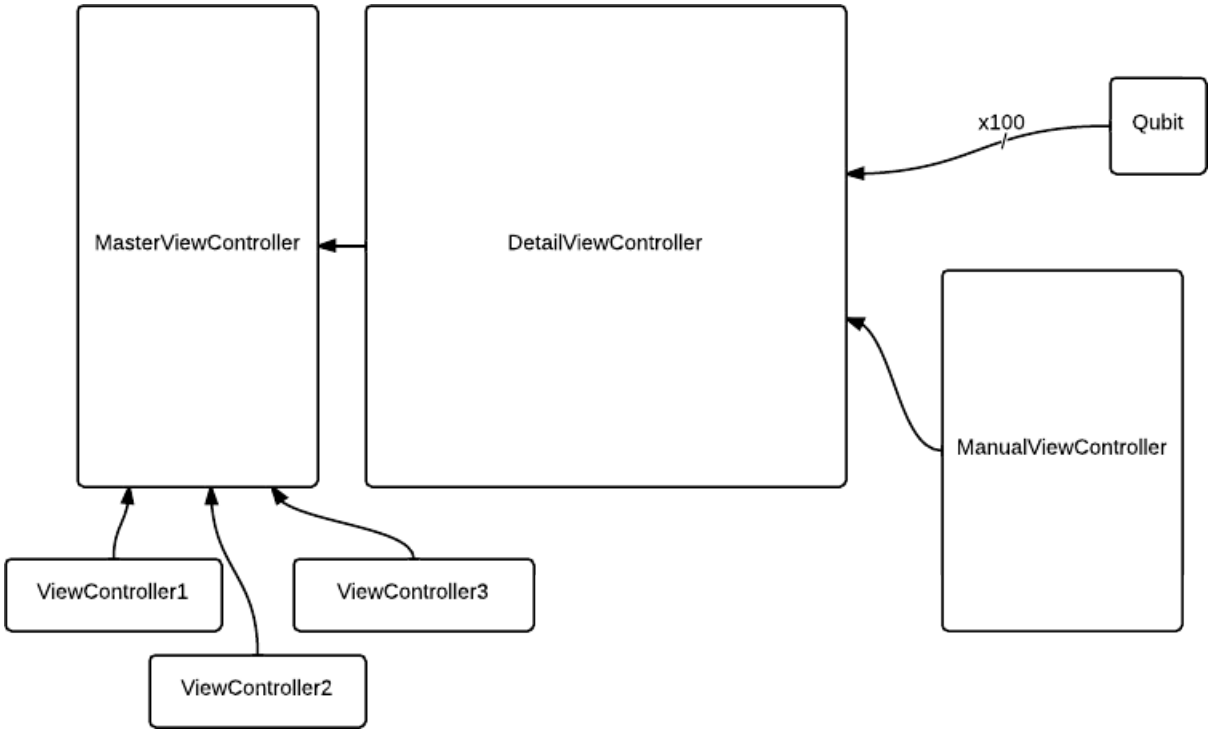

Figura 7.3: Esquema dels objectes del QGT 01

Pel que fa a l'objecte *storyboard*, en la figura 7.3 es pot veure un esquema de les vistes i els controladors de vista que relaciona. Els *storyboard* són uns objectes molt útils per agilitzar

el procés de disseny. No obstant això, no són tan flexibles com la programació en text pla i consumeixen recursos innecessàriament. Per aquest motiu, en la resta d'aplicacions QGT s'ha obviat aquest objecte i s'han programat tots els controladors de vista mitjançant codi.

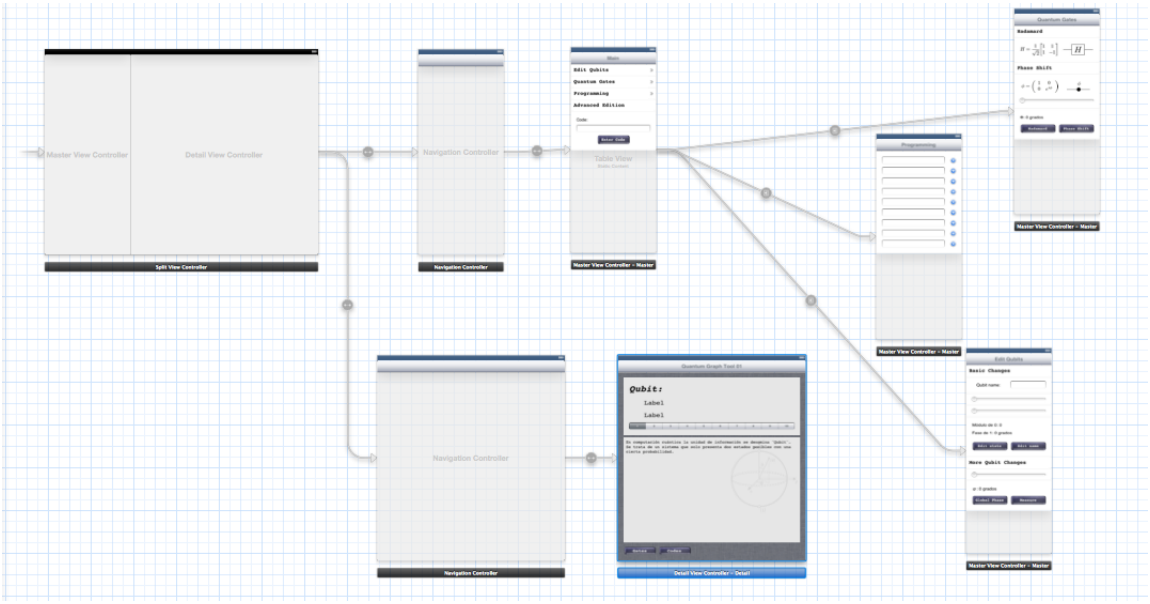

Figura 7.4: Objecte *storyboard* del QGT 01

7.4 Implementació

Tenint present l'estructura de l'apartat anterior, es comenten alguns aspectes més concrets relacionats amb la implementació de l'aplicació. Per no allargar el text innecessàriament, només es descriuen els elements relacionats amb la computació quàntica i els mètodes i classes més interessants.

Pel que fa a la classe *Qubit*, tenint en compte que cada qubit es manipula individualment, només cal emmagatzemar la parella de coeficients complexos que n'acompanyen els estats purs $|0\rangle$ i $|1\rangle$. Aquests coeficients es guarden en variables de tipus *float*, de manera que els qubits tenen un error intrínsec a causa de la limitació per codificar els nombres complexos. Tot i així, per les transformacions que s'usen en aquesta aplicació, l'error es pot menysprear. També es guarden les probabilitats de cada estat, les quals s'actualitzen després d'aplicar cada operador, per evitar realitzar el càlcul cada vegada que es vol accedir a la informació del qubit. D'altra banda, entre els mètodes d'aquesta mateixa classe hi ha les portes Hadamard i Phase Shift general; per tant, si es vol aplicar algun d'aquest operadors, només cal enviar el missatge corresponent al qubit.

Un aspecte interessant del QGT 01 és la possibilitat de programar petites instruccions per modificar els qubits. El «compilador» d'aquestes instruccions consisteix en un bucle que s'encarrega de llegir caràcters d'un *array*,[1] en grups de tres, que introdueix l'usuari.

[1] Les cadenes de caràcters en Objective-C són de classe *NSString*.

Cada terna es compara successivament amb les instruccions que estan programades per saltar a la rutina d'atenció corresponent (en cas de no trobar coincidència per a alguna instrucció, la ignora). Si la instrucció necessita un paràmetre numèric, llegeix la terna de caràcters següent i la converteix en un enter.

En la taula 7.1 es mostren els codis que s'han implementat en el compilador d'instruccions. El compilador no diferencia entre majúscules i minúscules per tal d'evitar errors per part de l'usuari.

INSTRUCCIÓ	**DESCRIPCIÓ**
had o **hdm**	Aplica l'operador Hadamard al qubit actual.
phs ###	Aplica l'operador Phase Shift amb un angle especificat per ### (en graus).
gol ###	Aplica una fase global amb un angle especificat per ### (en graus).
not o **pxg**	Aplica l'operador X de Pauli al qubit actual. Correspon a la porta Not clàssica.
pyg	Aplica l'operador Y de Pauli al qubit actual.
pzg	Aplica l'operador Z de Pauli al qubit actual.
tqg	Aplica l'operador T al qubit actual.
mea	Simula una mesura del qubit actual sense modificar-ne l'estat.
inf ###	Mostra el missatge d'informació del programa especificat per ###.
off ###	Canvia els deu qubits mostrats pel selector. El primer qubit mostrat correspon al qubit ### + 1 del registre.
lin ###	Canvia el nombre màxim de línies que es mostren en el terminal a ### línies.
pdf	Envia per correu un PDF que conté l'estat actual dels 100 qubits.
rst	Inicialitza els 100 qubits.
clr	Neteja la pantalla del terminal.

Taula 7.1: Llista d'instruccions per programar en el QGT 01.

El símbol ### correspon a un nombre de tres dígits. Per exemple, per escriure 15 s'ha d'afegir el primer zero, i. e. 015. Aquest «compilador» és un mètode del *DetailViewController*, però tant el botó per cridar el mètode com l'objecte de classe *UITextView* d'on extreu la informació de l'*array*, es troben instanciats en la vista del *MasterViewController*. Per aquest motiu, les accions del menú capturades pel *MasterViewController* (peticions de l'usuari) són reenviades al *DetailViewController* per ser ateses. De fet, això es pot extrapolar a tots els objectes del menú; per tant, tot i que el *MasterViewController* és l'objecte amb la jerarquia més alta, no té implementat cap mètode de càlcul.

Per acabar, un últim punt interessant és el generador de PDF, *SavePDF*, que s'encarrega de generar un document en aquest format estàndard i enviar-lo per correu des de la mateixa aplicació. Aquest objecte correspon a una versió simplificada de l'objecte inclòs

en el QGT 04, que porta el mateix nom. Per aquest motiu, la descripció d'aquest objecte es deixa per més endavant.

7.5 Presentació

Aquesta aplicació és la primera del projecte QGT que es va penjar a l'App Store d'Apple. El procés per penjar les aplicacions és llarg. En primer lloc, cal crear un identificador per a l'aplicació i un certificat de codificació amb les dades del desenvolupador. Aquest certificat de codificació, anomenat *Distribution Profile*, permet generar un arxiu compacte —anomenat *binari*— que representa l'executable de l'aplicació.[2] A continuació, s'ha d'omplir tota la informació obligatòria de l'aplicació per tal que es generi una entrada més al portal web *iTunes Connect*,[3] que és el portal dels desenvolupadors.

Si tota la informació és correcta, es pot penjar l'arxiu binari degudament signat. En aquest moment, l'aplicació apareix en l'estat *Waiting for Review*. Passada una setmana, l'aplicació és avaluada per Apple i, si no hi ha problemes de cap tipus,[4] s'actualitza la informació del web, l'aplicació canvia el seu estat per *Ready for Sale* i, al cap de poques hores, l'aplicació es troba disponible a l'App Store.

En la figura següent es motra la informació general de la aplicació que apareix al web *iTunes Connect* quan ha passat el procés d'avaluació:

Figura 7.5: Informació de la versió penjada del QGT 01

S'hi pot veure que el QGT 01 s'ha posat dins les categories d'educació (primària) i utilitats (secundària). A la part dreta apareixen els enllaços a les dades de la versió, els informes d'estabilitat, la informació de l'arxiu binari i la informació de l'historial. A més, es poden generar fins a un màxim 100 codis promocionals (*Promo Codes*), que permeten descarregar l'aplicació gratuïtament.

[2] A més del *Distribution Profile* també es poden generar els *Developer Profile* per testejar les aplicacions en un dispositiu físic.

[3] Accessible a *https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa*.

[4] El procés d'avaluació d'Apple és molt rigorós i l'aplicació pot ser rebutjada si hi ha problemes d'estabilitat, de consum de recursos, de contingut inapropiat...

Pel que fa a la informació detallada de la versió actual, es pot veure en la figura següent.

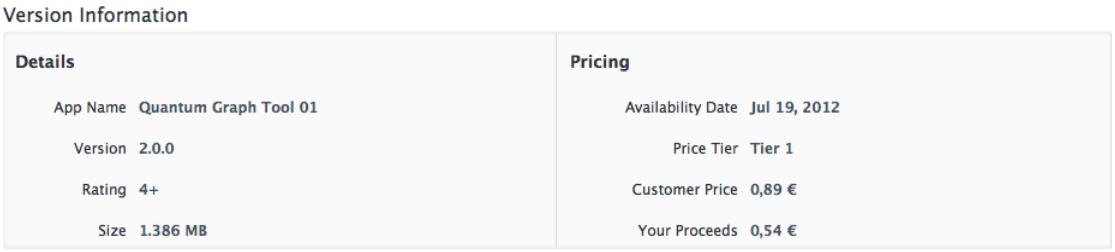

Figura 7.6: Detalls de la versió penjada del QGT 01

Per ordre, conté: el nom complet de l'App, la versió actual, la categoria per contingut (estipulada per Apple), la mida del fitxer binari en MB, la data de publicació de la primera versió, la categoria de preus, el preu de venta i el que cobra el desenvolupador per unitat venuda.

Finalment, es presenta l'aspecte de l'aplicació a l'App Store.

Figura 7.7: Dades del QGT 01 a l'App Store

CAPÍTOL

8

BLOC PRÀCTIC III: QGT 02

8.1 Introducció

La segona aplicació *Quantum Graph Tool* també és una eina introductòria que, en aquest cas, pretén facilitar l'estudi dels grafs. En l'aplicació es presenten els grafs bàsics i la manera de codificar-los mitjançant els dos mètodes presentats en el capítol 4: utilitzant matrius d'adjacència i llistes d'adjacència. En cada mode de codificació, es poden manipular deu grafs en una vista de tipus terminal, que inclou la representació gràfica del graf en temps real.

L'aplicació incorpora un conjunt de botons intuïtius que permeten controlar les instruccions que es produeixen en el terminal com si es tractés d'una calculadora. El botó més interessant de l'aplicació correspon al d'informació de graf, «i», el qual mostra pel terminal les dades de la representació codificada i ordenada del graf (segons el mode), de la successió de graus dels nodes i del nombre de nodes i arestes del graf.

A més, el mode de codificació mitjançant llistes d'adjacència permet manipular grafs ponderats (amb pesos a les arestes), els quals resulten molt útils en algorismes d'enrutat i d'optimització de processos (correspon als botons blaus de l'App).

8.2 Descripció

Quan s'obre l'aplicació, es mostra un menú de selecció per escollir el mode de codificació dels grafs: *Adjacency matrix* o *Adjacency list*. Un cop escollit el mode, s'accedeix a la vista terminal (veure figura 8.1). En la vista terminal hi ha un selector com el del QGT 01 per escollir entre els deu grafs diferents. Cada graf està inicialitzat al graf elemental que

conté un únic node (i cap aresta). A més, els deu grafs tenen els nodes de color diferent per tal d'identificar-los gràficament, per ordre: vermell, verd, blau, groc, cian, rosa, caqui, magenta, lila i negre.

A la dreta del terminal hi ha tres columnes de botons que permeten modificar els grafs. La primera columna correspon als botons per introduir els nombres decimals; la segona, als botons de generació de grafs (els quals necessiten paràmetres addicionals per definir els grafs), i la tercera, als botons de manipulació. En la figura següent, es poden veure dos exemples de manipulació de grafs representats mitjançant matrius d'adjacència (esquerra) i llistes d'adjacència (dreta).

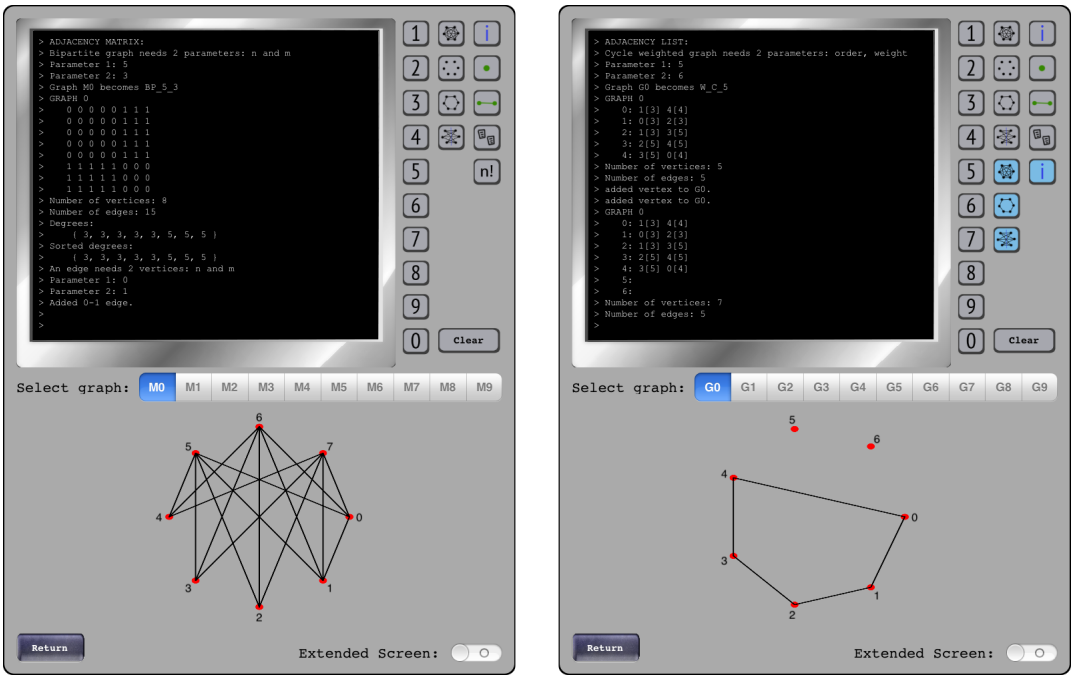

Figura 8.1: Vistes d'edició de grafs del QGT 02

8.3 Estructura

A banda les llibreries bàsiques, el QGT 02 presenta nou classes d'objectes que segueixen l'esquema MVC. Com es comenta anteriorment, aquesta aplicació no disposa de *storyboard*; per aquest motiu, les instàncies de tots els controladors de vista s'han realitzat manualment. A més, per tal d'estalviar memòria, els objectes es creen en el moment en què es volen utilitzar i es destrueix tota la informació quan es torna al menú inical. A continuació, es descriuen amb detall les nou classes del QGT 02:

- **AppDelegate:** és el nucli de l'aplicació. Conté un objecte de classe *ViewController* que es crea automàticament i s'executa quan s'obre l'aplicació.
- **ViewController:** és el controlador del menú inicial i conté tres objectes controladors com a atributs —un *GraphMatrixViewController*, un *GraphListViewController* i un *AdvancedMatrixViewController*. A més, implementa un mètode automàtic per generar un polígon regular de N costats, que es mostra en el menú inicial.

- **GraphMatrixViewController:** és el controlador de la vista i el terminal del mode matrius d'adjacència. Entre els seus atributs hi ha els deu grafs de classe *Matrix* i una *array* d'objectes *NSString* per emmagatzemar les instruccions mostrades en el terminal.
- **GraphListViewController:** és l'equivalent a l'anterior pel mode llistes d'adjacència. En comptes d'objectes de classe *Matrix*, té deu objectes de classe *List*.
- **AdvancedMatrixViewController:** és un controlador d'un terminal addicional que permet executar algorismes que resolen problemes de coloració minimal de mapes. Utilitza llistes d'adjacència per codificar els grafs utilitzats. Per qüestions de rendiment i complexitat d'ús, encara no s'ha inclòs en la versió penjada de l'aplicació.
- **Matrix:** és l'objecte que representa el graf codificat amb una matriu d'adjacència. Conté els mètodes que generen els tipus de graf bàsics i la seva informació associada.
- **List:** és l'objecte que representa el graf codificat amb una llista d'adjacència. Conté els mètodes per generar els tipus de graf bàsics i la seva informació associada.
- **Tablero:** és un objecte encarregat de dibuixar mapes modelats amb grafs per presentar la informació dels algorismes de coloració.
- **DrawLine:** és l'objecte encarregat de dibuixar els grafs dels controladors i els polígons regulars del menú principal.

En la figura 8.2 es pot veure representat l'esquema jeràrquic d'aquests objectes.

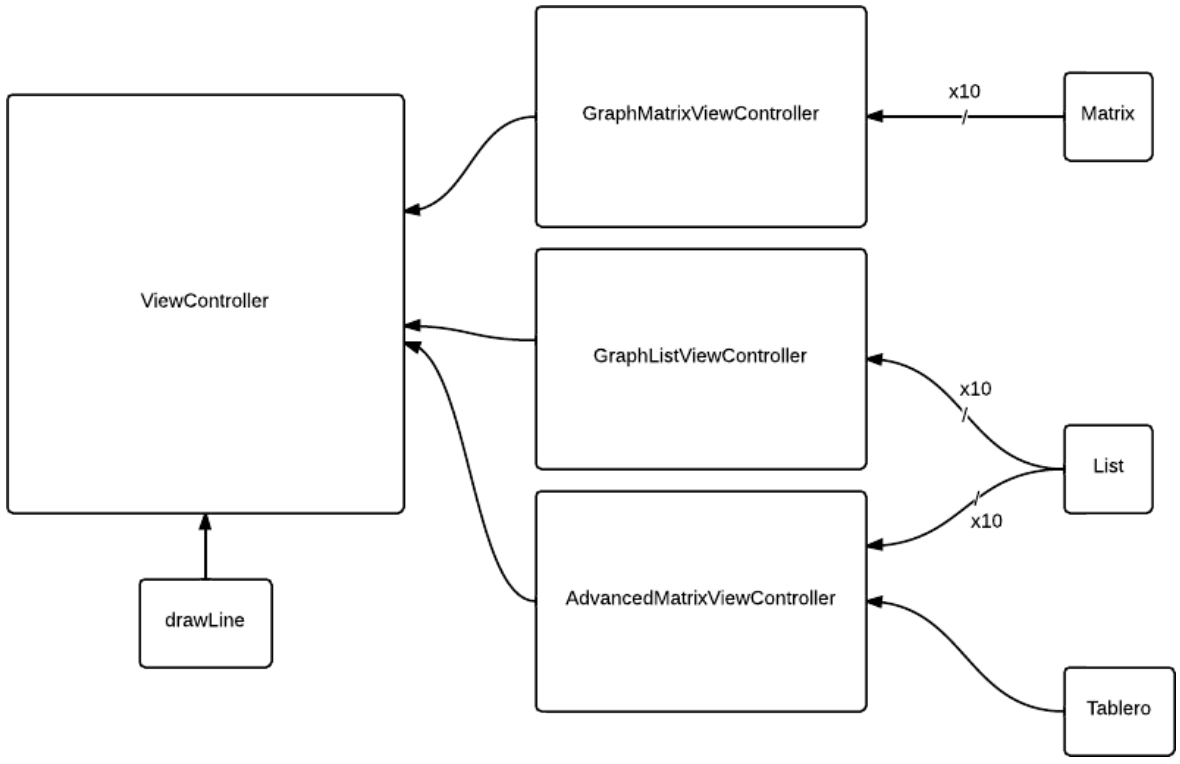

Figura 8.2: Esquema dels objectes del QGT 02

8.4 Implementació

En aquest apartat es fa una breu descripció de la implementació d'algunes de les classes anteriors. Com que les classes *List* i *Matrix* s'encarreguen de codificar el mateix element, els mètodes que tenen definits són pràcticament els mateixos. Tot i això, com que la manera de codificar els grafs és molt diferent segons si s'utilitzen llistes o matrius, la implementació dels mètodes també ho és. Per exemple, els mètodes per afegir nodes i arestes a un graf de la classe *Matrix* són:

```
- (void) addVertex{
    if (order < MAX_ORDER) {
        int auxiliar[MAX_ELEMENTS];
        int pos = 0;
        for (int fil = 0; fil<order; fil++) {
            for(int col = 0; col < order; col++){
                auxiliar[pos] = elements[fil*order+col];
                pos++;
            }
            auxiliar[pos] = 0;
            pos++;
        }
        for (int i = 0; i <= order; i++) {
            auxiliar[pos] = 0;
            pos++;
        }
        for (int j = 0; j<= pos; j++) {
            elements[j] = auxiliar[j];
        }
        order++;
    }
}

- (void) addEdge:(int)a:(int)b{
    if (a < order && b < order) {
        elements[a*order+b]++;
        elements[b*order+a]++;
    }
}
```

On es pot veure que l'atribut que emmagatzema el graf és un vector,[1] anomenat *elements*, i l'element x_{ab} de la matriu es troba en la posició $a \cdot order + b$. D'altra banda, els mateixos mètodes en la classe *List*:

```
- (void) addVertex{
    if (order < MAX_VERT) {
        edges[order] = 0;
        order++;
    }
}

- (void) novaArestaAfegida_A_B_Weight:(int)a:(int)b:(int)weight{
    if ((a != b) && (a<order) && (b<order)) {//No pot ser llaç i han d'existir els nodes.
        elements[a][edges[a]][0] = b;
        elements[b][edges[b]][0] = a;
        elements[a][edges[a]][1] = weight;
        elements[b][edges[b]][1] = weight;
        edges[a]++;
        edges[b]++;
    }
}
```

Aquests mètodes mostren l'estructura vectorial (de tres paràmetres) per guardar les llistes dels nodes, així com un vector auxiliar, *edges*, que determina el nombre d'arestes que incideixen a cada node, i. e. el nombre d'elements de la seva llista. A més, es pot veure

[1] El vector *elements* conté $order \times order$ elements, de manera que és equivalent a la matriu d'adjacència.

que la classe *List* permet treballar amb grafs ponderats introduint-hi un paràmetre addicional, *weight*. Si es treballa amb grafs no ponderats, el paràmetre *weight* de cada aresta s'inicialitza per defecte a 1; de manera que, a efectes pràctics, és com si no s'hi afegís ponderació i tots els algorismes el tracten com si no fos ponderat.

Les classes *GraphListViewController* i *GraphMatrixViewController* s'encarreguen de gestionar la vista d'usuari i de traduir les peticions dels botons a mètodes de les classes *List* i *Matrix*, respectivament. Cada botó genera una interrupció que crida un mètode del controlador, el mètode *teclaPulsada*, el qual utilitza l'identificador (*tag*) del botó que ha enviat el missatge per determinar la instrucció seleccionada per l'usuari. Cada instrucció s'estructura de la manera següent: es determina si la instrucció necessita més paràmetres; en cas afirmatiu, s'entra en mode espera de paràmetres; es modifica la informació relacionada amb la instrucció per presentar-la per pantalla; es determina si ja es tenen tots els paràmetres necessaris de la instrucció i si són correctes, i, finalment, s'envia un missatge a l'objecte *List* o *Matrix* per executar la instrucció. Si s'ha realitzat amb èxit, es refresquen el terminal i el dibuix del graf i se surt del mode espera de paràmetres (en cas que s'hagués activat). Aquesta implementació resulta molt interessant, ja que simula un «processador de grafs» i s'espera reaprofitar aquesta mateixa estructura per a properes aplicacions del projecte QGT.

Finalment, tot i que les classes *Tablero* i *AdvancedMatrixViewController* resulten molt interessants, tant computacionalment com conceptualment, no se n'explica el contingut ni el funcionament, ja que es basen en *coloració de grafs* i el *mètode de Brelaz*, que són conceptes que no es presenten en el bloc tèoric corresponent. A més a més, encara no són accessibles per als usuaris que es descarreguin el QGT 02 de l'App Store.

8.5 Presentació

Aquesta aplicació es va penjar a l'App Store quatre mesos més tard que l'anterior, en part, a causa de les dificultats que suposava implementar el processador d'instruccions i l'algorisme que dibuixa els grafs. El procés va ser el mateix que el del QGT 01, però en aquest cas es van utilitzar dos iPads físics diferents per provar el rendiment (mitjançants els *Developer Profiles*). En la figura següent es mostra la informació general que apareix al web *iTunes Connect* després de passar el procés d'avaluació.

Figura 8.3: Informació de la versió penjada del QGT 02

En aquest cas, s'ha posat únicament dins la categoria d'educació. Pel que fa a la informació detallada de la versió actual, es mostra en la figura següent.

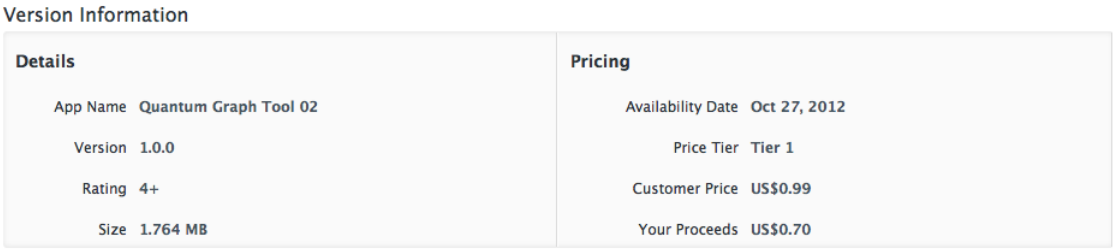

Figura 8.4: Detalls de la versió penjada del QGT 02

Es pot veure que aquesta aplicació ocupa una mica més que l'anterior a causa de les imatges dels botons i els fons.

Finalment, es presenta l'aspecte de l'aplicació a l'App Store:

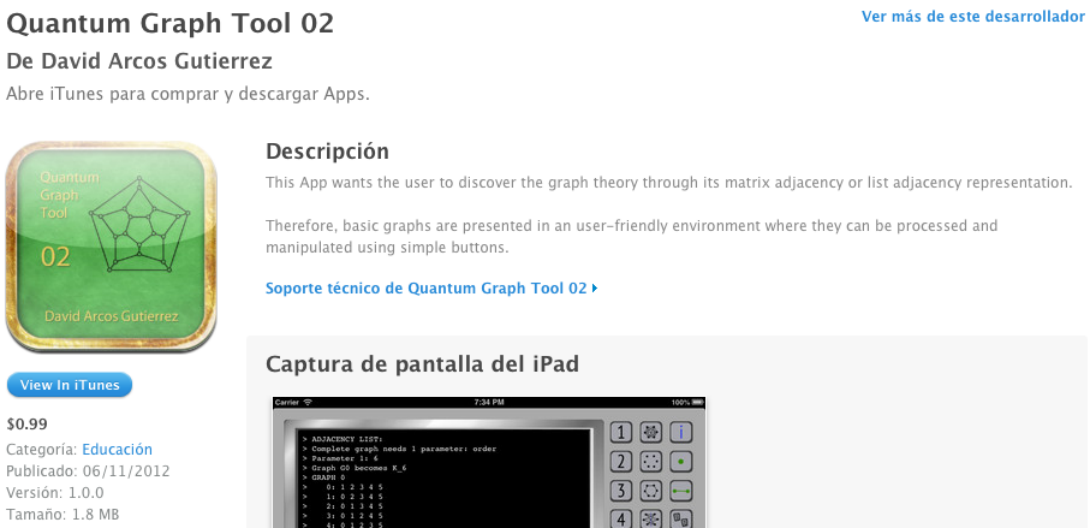

Figura 8.5: Dades del QGT 02 a l'App Store

CAPÍTOL

9

BLOC PRÀCTIC IV: QGT 03

9.1 Introducció

La tercera aplicació *Quantum Graph Tool* representa un simulador de l'algorisme quàntic de Grover, el qual es descrit en el bloc teòric II. L'aplicació permet crear llistes ordenades d'entre 4 i 127 elements (2-7 qubits) i comprovar el comportament del circuit a cada iteració G de l'algorisme. Donat el seu caràcter descriptiu, s'ha desenvolupat un entorn gràfic intuïtiu per tal que l'usuari mitjà sigui capaç de seguir tot l'algorisme. L'usuari pot escollir entre veure l'efecte de cada bloc mitjançant animacions gràfiques o una petita descripció del comportament. De la mateixa manera, pot escollir com visualitzar els estats del registre en cada punt del circuit.

L'interès de programar un simulador d'un algorisme quàntic en un iPad pot ser discutible, ja que no és eficient ni presenta cap utilitat aparent. No obstant això, donada la filosofia de les aplicacions QGT, pot ser molt interessant disposar d'una eina per estudiar les noves maneres de computar a partir dels avenços que ja han realitzat altres (en aquest cas, Lov Grover). D'altra banda, disposar d'una aplicació intuïtiva i animada del circuit, pot ajudar a difondre el potencial de la computació quàntica per motivar joves científics i enginyers.

9.2 Descripció

En la pantalla inicial de l'aplicació QGT 03, apareixen les opcions de configuració inicial del circuit (veure figura 9.1). A l'esquerra hi ha dos selectors que permeten esollir el nombre de qubits que codifiquen la llista i el nombre total d'iteracions de Grover (blocs G) que es volen fer en el circuit (per defecte hi ha l'opció òptima, veure (3.34)).

A la dreta apareix un selector numèric de classe *PickerView*, de tres columnes, per seleccionar l'element buscat. Si s'escull un element que no es troba a la llista, el selector numèric es reordena automàticament per presentar l'element més gran.

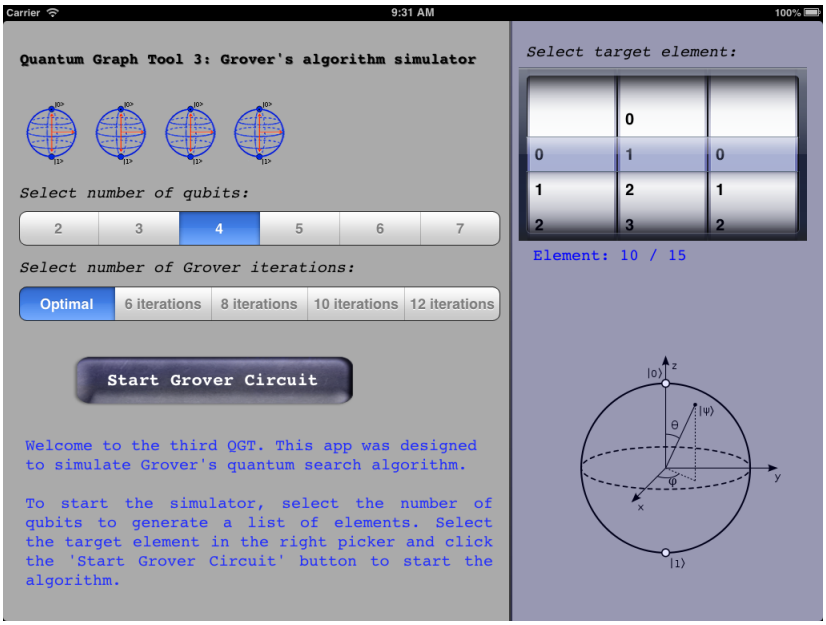

Figura 9.1: Pantalla inicial del QGT 03

Un cop configurat el circuit, es pot iniciar l'algorisme mitjançant el botó *Start Grover Circuit* per passar a la pantalla de circuit (figura 9.2).

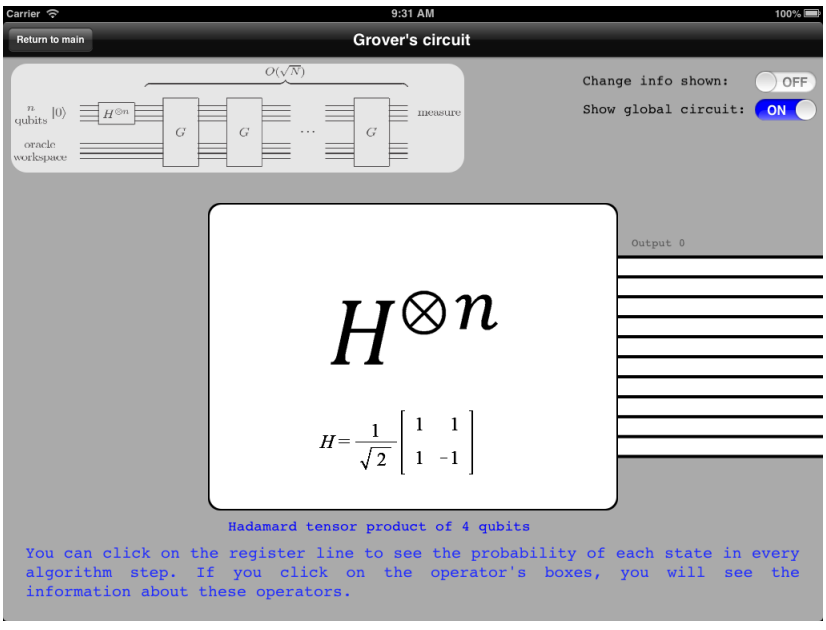

Figura 9.2: Pantalla del circuit del QGT 03

En aquesta vista apareixen els operadors en caixes al centre i les línies del registre que uneixen els operadors.

En fer clic sobre un operador apareix l'animació que en mostra el comportament sobre l'espai de Hilbert tal com es veu en la captura següent:

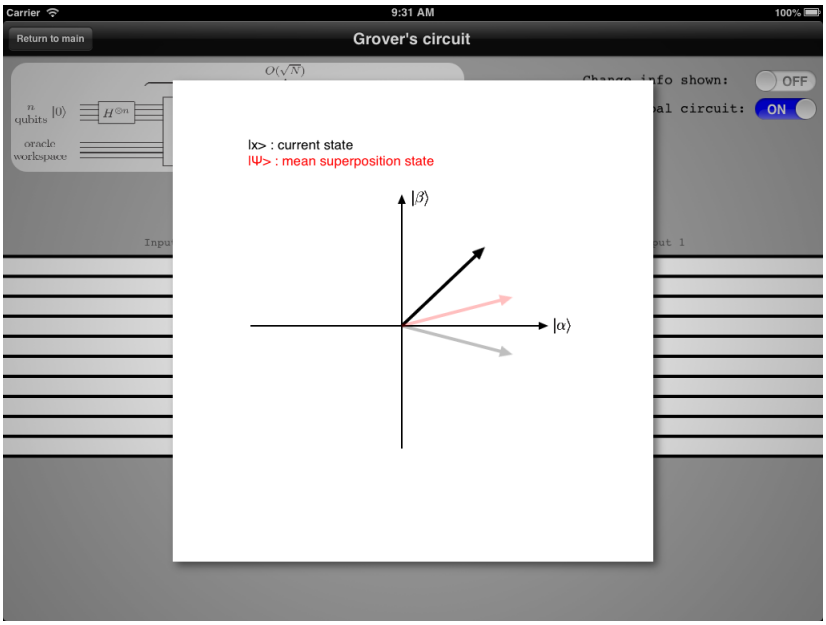

Figura 9.3: Animació d'una iteració al QGT 03

D'altra banda, si es fa clic sobre les línies del registre, apareix una gràfica de probabilitat de cada estat.

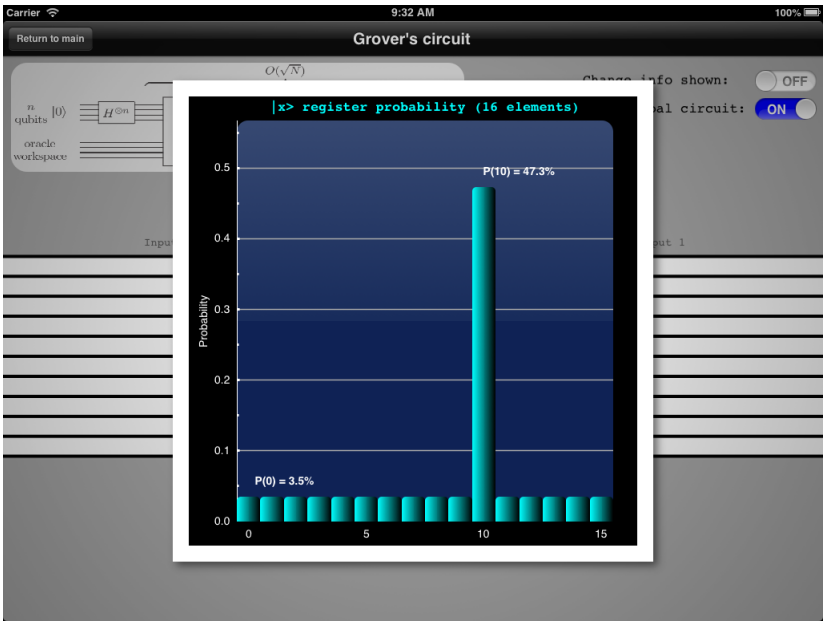

Figura 9.4: Gràfic de probabilitat del registre al QGT 03

NOTA: en l'exemple de la figura 9.4 es mostra que la probabilitat de l'element buscat és pràcticament d'un 50 % després de la primera iteració de Grover. Un resultat molt millor que el de qualsevol buscador clàssic de 16 elements.

L'aplicació també té dos selectors de classe *Switch*. El primer és per mostrar o amagar l'esquema complet del circuit que apareix a la part superior. El segon canvia la informació que es mostra quan es fa clic sobre els operadors i les línies de registre (es canvien les animacions i els gràfics de barres per informació escrita sobre els operadors i els estats).

Finalment, cal comentar que, com que l'aplicació està totalment en anglès, els noms dels operadors s'han deixat amb la nomenclatura anglesa; per tant, l'inversor respecte la mitjana apareix com a *diffusion operator*.

9.3 Estructura

A banda de les llibreries bàsiques i la llibreria libCorePlot de Google, el QGT 03 presenta únicament 4 classes d'objectes que segueixen l'esquema MVC. Com que la càrrega d'aquesta aplicació és bàsicament computacional, per tal d'optimitzar-ne l'eficiència, s'utilitza una única classe per executar tota la simulació de l'algorisme. Pel mateix motiu, per tal d'estalviar recursos i, com que l'algorisme de Grover només utilitza nombres reals, no s'ha utilitzat la classe *Qubit* del QGT 01 per codificar la informació. A continuació es detallen les quatre classes que constitueixen el QGT 03.

- **AppDelegate:** és el nucli de l'aplicació. Conté l'objecte *ViewController* que es crea automàticament com en la resta d'aplicacions.

- **ViewController:** és el controlador de la vista inicial. Conté l'objecte *CircuitViewController* que representa el circuit de Grover. Aquest controlador es comporta, a més, com un delegat del selector *PickerView* per capturar les dades de les columnes de dígits a temps real.

- **CircuitViewController:** és el controlador de la vista i el model del circuit de Grover. Es tracta d'un objecte molt gran que conté vint atributs i disset mètodes, entre els quals hi ha les matrius dels operadors, els estats de totes les iteracions i els objectes que dibuixen les animacions i les gràfiques. Aquest objecte és tan gran que l'iPad no pot emmagatzemar a la memòria dinàmica més d'un d'aquests objectes quan té tots els atributs actius (instanciats); per aquest motiu, conté mètodes que alliberen la memòria que no s'utilitza.

- **DrawMatrixViewController:** és el controlador de les vistes que generen la informació dels operadors i els estats. Conté els mètodes que generen les animacions dels operadors, així com les gràfiques de probabilitat dels estats.

A més d'aquests objectes, s'han utilitzat *NSMutableArray* per emmagatzemar els nombres que es capturen en el delegat de la vista inicial (per a cada columna del selector *PickerView*).

En la figura 9.5 es pot veure representat l'esquema jeràrquic dels objectes descrits (llevat de l'objecte principal *AppDelegate*, ja que només s'encarrega de crear el controlador de la vista).

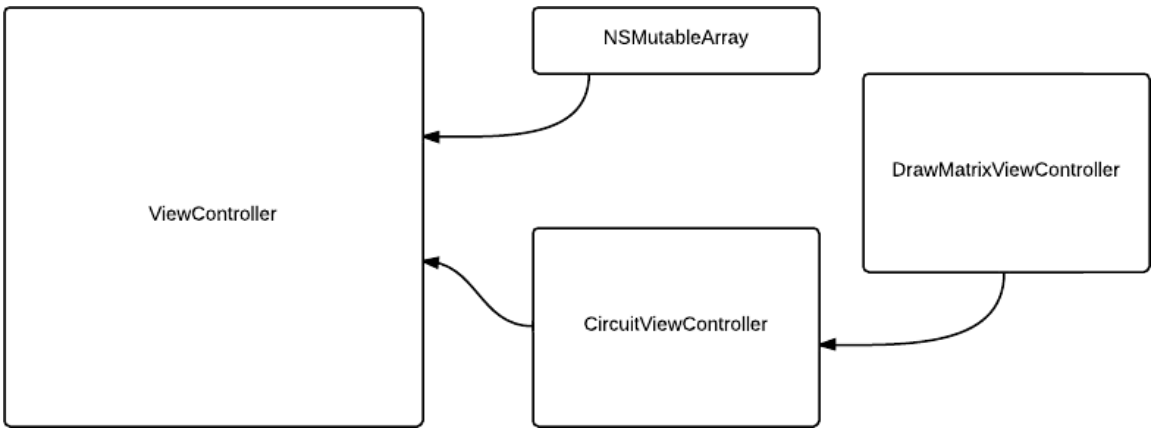

Figura 9.5: Esquema dels objectes del QGT 03

Per tal de millorar l'eficiència de l'aplicació, el *DrawMatrixViewController* només s'instancia quan és necessari i es destrueix quan finalitza la visualització. Per tant, cada cop que es fa clic sobre les línies del registre o els blocs dels operadors, es creen dinàmicament les gràfiques o les animacions corresponents i s'eliminen quan l'usuari surt de la vista.

9.4 Implementació

Com que el codi és molt llarg i hi ha molts mètodes dedicats a la gestió de les vistes, en aquest apartat només es comenten els punts més rellevants de la implementació que tenen a veure pròpiament amb l'algorisme.

En primer lloc, és important destacar l'estratègia que s'ha utilitzat per emmagatzemar la informació dels registres i dels operadors.

```
double state[Nmax];
double kstate;

double SavedStates[3][Imax];

int Grover[Nmax][Nmax];
int kGrover;
int Hadamard[Nmax][Nmax];
int kHadamard;
```

On es pot veure que, llevat de la matriu *SavedStates*, les dades estan dividides en dues parts: una matriu (o vector) i una variable començada per k. A més, totes són de tipus *int* o *double*. Les variables k- corresponen als denominadors al quadrat, per això sempre tenen valor enter. Pel que fa a les matrius, com que en l'algorisme de Grover no hi ha fases diferents de 180º, es corresponen amb els numeradors de cada coeficient. Per exemple, pel cas de la matriu Hadamard:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \xrightarrow{es\ codifica} \begin{cases} \text{kHadamard} = (\sqrt{2})^2 = 2 \\ \text{Hadamard [0] [Nmax]} = \begin{pmatrix} 1 & 1 \end{pmatrix} \\ \text{Hadamard [1] [Nmax]} = \begin{pmatrix} 1 & -1 \end{pmatrix} \end{cases}$$

Amb aquesta codificació, per aplicar els operadora només cal realitzar un producte matricial i multiplicar les respectives variables k. Pel que fa a la variable *SavedStates*, la qual guarda tots els registres del circuit, encara s'ha utilitzat una estratègia més compacte. Per a cada estat del regsitre només es guarden tres enters: el denominador al quadrat (la variable k de l'estat) i dos numeradors, el de l'element buscat i qualsevol dels altres (ja que tots són idèntics).

Un altre aspecte interessant és la construcció de la matriu Hadamard que s'ha implementat. Es tracta d'un mètode recursiu que consisteix a copiar l'estructura de la matriu Hadamard tres vegades i una d'elles amb inversió de signe.

```
- (void)Hadamard_Tensor{

    int i, j, dim;

    //Inicialització
    for(i = 0; i<Nmax; i++)
        for(j = 0; j<Nmax; j++)
            Hadamard[i][j] = 0;

    //Hadamard d'un qubit
    Hadamard[0][0] = 1;
    Hadamard[0][1] = 1;
    Hadamard[1][0] = 1;
    Hadamard[1][1] = -1;
    kHadamard = 2;

    for (dim = 2; dim < n; dim = dim*2) {

        //Producte de denominadors
        kHadamard = 2*kHadamard;

        //Duplicació de la Hadamard anterior
        for (i = 0; i < dim; i++) {
            for (j = 0; j < dim; j++) {
                Hadamard[i][j+dim] = Hadamard[i][j];
                Hadamard[i+dim][j] = Hadamard[i][j];
                Hadamard[i+dim][j+dim] = -1*Hadamard[i][j];
            }
        }
    }
}
```

Tot i això, en la versió actual de l'aplicació aquest mètode no s'utilitza, ja que la matriu Hadamard només s'usa per convertir l'estat $|0\rangle$ en l'estat superposició $|\psi\rangle$, de manera que amb el producte de les k n'hi ha prou.[1]

```
- (void)applyHadamard_FAST{

    kstate = kHadamard*kstate;

    for (int j = 0; j < n; j++) {
        state[j] = 1;
    }
}
```

En efecte, el mètode *applyHadamardFAST* únicament posa uns a tots els elements del vector d'estat, ja que l'algorisme suposa que es troba inicialment a l'estat $|0\rangle$.

[1] En una primera versió del QGT 03 es mostrava la informació dels coeficients de tots els operadors de l'algorisme. Això resultava útil per comprovar el bon funcionament dels algorismes, però no és interessant per l'usuari.

Quan l'usuari fa clic en el botó *Start Grover Circuit*, es crea l'objecte *CircuitViewController* i aquest executa inmediatament totes les iteracions de Grover executant el codi que es mostra a continuació.

```
- (void) Grover_algorithm{

    //Per generar un mínim de 4 estats
    if (qubits == 0)
        qubits = 2;

    n = 1;
    for (int k = 0; k<qubits; k++) {
        n = n*2;
        if(n > Nmax){
            n = Nmax;
            k = qubits;
        }
    }

    //Inicialitzar vector d'estats
    for (int l = 0; l < Nmax; l++) {
        state[l] = 0;
    }
    state[0] = 1;
    kstate = 1;

    //Iteracions que calen?
    switch (iterationSelected) {
        case 1:
            iterationSelected = 6 + 1;
            break;
        case 2:
            iterationSelected = 8 + 1;
            break;
        case 3:
            iterationSelected = 10 + 1;
            break;
        case 4:
            iterationSelected = 12 + 1;
            break;
        default:
            iterationSelected = M_PI*sqrt(n)/4 + 1;
            break;
    }

    other = (element+1)%n;
    iter = 0;

    [self Hadamard_Tensor];
    [self Fast_Make_Gro_Ora];
    [self applyHadamard_FAST];

    for (int l = 0; l<iterationSelected; l++) {
        [self G_iteration];
    }
}
```

```
- (void)G_iteration{

    //Es guarda l'estat anterior
    SavedStates[0][iter] = kstate;
    SavedStates[1][iter] = state[element];
    SavedStates[2][iter] = state[other];

    iter++;

    NSLog(@"Grover iteration: %d",iter);

    //S'aplica l'operador de Grover
    [self applyOracle];
    [self applyGrover_FAST];
}



- (void)applyOracle{

    //Més eficient computacionalment.
    state[element] = -1*state[element];
}



- (void)applyGrover_FAST{

    int i, j;
    double auxState[n];

    kstate = kGrover*kstate;

    for (i = 0; i < n; i++) {
       auxState[i] = 0;
       for (j = 0; j < n; j++) {
           if (i != j) {
               auxState[i] = auxState[i] + state[j];
           }else{
               auxState[i] = auxState[i] + state[j]*(1-(n/2));
           }
       }
    }

    for (j = 0; j < n; j++) {
        state[j] = auxState[j];
    }
}
```

Cal notar que l'operador Oracle és molt simple d'implementar en aquesta aplicació de simulació,[2] però constitueix un pas molt complicat de l'algorisme quàntic tal com s'ha vist en el bloc teòric II.

A més de l'algorisme i els atributs dels operadors i els estats, el *CircuitViewController* presenta molts més objectes i mètodes destinats a manipular la informació que es mostra per pantalla. Aquests atributs es mantenen en memòria dinàmica per tal de presentar les dades sense retards quan es vol generar una animació o un gràfic; per aquest motiu, l'objecte és molt pesat. En cada punt de l'aplicació s'intenta mantenir un equilibri entre consum de memòria i velocitat de resposta; de fet, en les primeres implementacions de l'algorisme, es van aconseguir programar fins a 13 qubits (8192 elements a la llista) a costa de mètodes de compressió dels estats i d'un temps d'espera aproximat de 30 segons en el pitjor cas. Com que per a l'objectiu de l'aplicació, això no era gaire útil, la versió final només admet fins a 128 elements, amb 7 qubits, i així s'obtenen temps d'execució de l'ordre de les dècimes de segon.

[2] De fet, s'utilitza el fet que ja es coneix l'element buscat i simplement se'n canvia el signe.

9.5 Presentació

Aquesta aplicació es va penjar a l'App Store a finals de maig del 2013. El procés va ser el mateix que en les aplicacions anteriors, tot i que, a causa d'una actualització del servidor d'Apple, es va allargar una mica la gestió dels *Profiles*.

De la mateixa manera que en el QGT 02, es van utilitzar dos iPads físics diferents per provar-ne el rendiment. A més, es va utilitzar un simulador virtual d'iPad 3 per provar l'aspecte i el rendiment de les animacions i les gràfiques en un dispositiu amb pantalla *Retina Display*.[3]

En la figura següent es mostra la informació general que apareix al web *iTunes Connect* després de passar el procés d'avaluació:

Figura 9.6: Informació de la versió penjada del QGT 03

En aquesta App s'ha tornat a posar la segona categoria: utilitats. Pel que fa a la informació detallada de la versió actual, es recull en la figura següent:

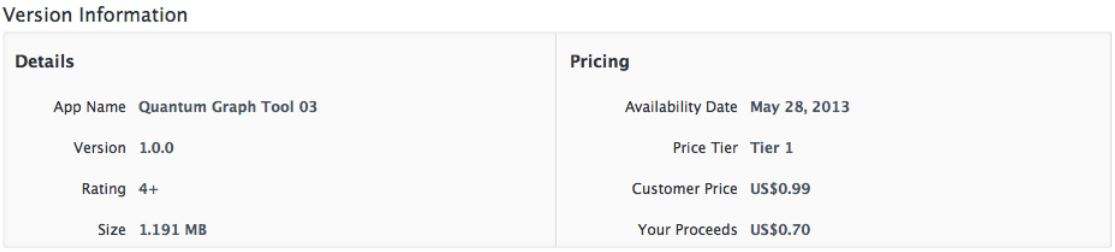

Figura 9.7: Detalls de la versió penjada del QGT 03

On es veu clarament que el consum de memòria estàtica (mida del fitxer) és també molt baix (inferior a 2 MB com en les aplicacions anteriors), ja que totes les grafiques i animacions es creen en temps d'execució. El preu i les categories de preus i continguts també són iguals que en les anteriors aplicacions.

[3] A partir de la versió 3 de l'iPad, s'utilitzen més punts en la pantalla: 2048×1536 píxels en comptes dels 1024×768 píxels dels models anteriors.

Finalment, es presenta l'aspecte de l'aplicació a l'App Store:

Figura 9.8: Dades del QGT 03 a l'App Store

9.6 Simulacions

A continuació, es presenten els resultats de la simulació de l'algorisme en realitzar les vuit primeres iteracions. Es tracta de les probabilitats de trobar l'element buscat i qualsevol altre per a cada iteració de l'algorisme (començant per aplicar la porta Hadamard).

qubits	estat	H %	G1 %	G2 %	G3 %	G4 %	G5 %	G6 %	G7 %	G8 %
2	buscat	25,0	100,0	25,0	25,0	100,0	25,0	25,0	100,0	25,0
2	altre	25,0	0,0	25,0	25,0	0,0	25,0	25,0	0,0	25,0
3	buscat	12,5	78,1	94,5	33,0	1,2	54,7	99,9	57,7	1,9
3	altre	12,5	3,1	0,8	9,6	14,1	6,5	0,0	6,0	14,0
4	buscat	6,2	47,3	90,8	96,1	58,2	12,5	2,0	36,5	83,6
4	altre	6,2	3,5	0,6	0,3	2,8	5,8	6,5	4,2	1,1
5	buscat	3,1	25,8	60,2	89,7	99,9	86,0	54,6	21,0	1,4
5	altre	3,1	2,4	1,3	0,3	0,0	0,5	1,5	2,5	3,2
6	buscat	1,6	13,5	34,4	59,1	81,6	96,4	99,7	90,7	71,8
6	altre	1,6	1,4	1,0	0,6	0,3	0,1	0,0	0,1	0,4
7	buscat	0,8	6,9	18,3	33,7	51,1	68,4	83,4	94,2	99,6
7	altre	0,8	0,7	0,6	0,5	0,4	0,2	0,1	0,0	0,0

Taula 9.1: Resum de les simulacions realitzades amb el QGT 03

S'han marcat en blau els percentatges de les iteracions òptimes.

En la taula també es pot veure l'efecte de realitzar més iteracions. En general, si es realitzen més iteracions, la probabilitat de trobar l'element objectiu empitjora, de manera que es pot arribar al punt en què l'element buscat és menys probable que la resta d'elements. No obstant això, també es pot aconseguir una millora respecte de l'òptim si s'incrementa el nombre d'iteracions.[4] Per exemple, si s'utilitzen 3 qubits per codificar els elements, l'òptim correspon a realitzar dues iteracions, de manera que s'obté l'element buscat amb una probabilitat del 94,5 %, però, si en comptes de dues iteracions se'n fan sis, la probabilitat de l'element buscat passa a ser del 99,9 %. Cal dir, però, que, en general, això és molt poc eficient i surt més a compte quedar-se amb el 94,5 % i repetir el procés en cas d'error, és a dir:

$P(element)$ sense reiniciar:	94,5 %	2 intents (mitjana)
$P(element)$ reiniciant 1 cop:	$1-(1-0,945)^2 = 99,7$ %	2,104 intents
$P(element)$ reiniciant 2 cops:	$1-(1-0,945)^3 = 99,98$ %	2,111 intents
$P(element)$ reiniciant ∞ cops:	$1-(1-0,945)^\infty = 100,0$ %	2,116 intents

De manera que sempre s'han de realitzar les iteracions corresponents al primer màxim (òptim) per obtenir el millor resultat de la cerca.

[4] El nombre òptim d'iteracions correspon al millor compromís entre nombre d'iteracions i probabilitat de trobar l'element buscat.

CAPÍTOL

10

BLOC PRÀCTIC V: QGT 04

10.1 Introducció

La quarta aplicació *Quantum Graph Tool* és un simulador de l'algorisme quàntic de Shor. Aquesta és amb diferència l'aplicació més complexa i delicada d'aquest projecte, en part, a causa de la dificultat intrínseca de l'algorisme. L'aplicació permet executar l'algorisme quàntic descrit en el bloc teòric II per factoritzar els nombres enters menors que 63. A més, s'ha inclòs una vista per estudiar l'aritmètica entera, en què l'usuari pot factoritzar nombres enters molt més grans (de fins a nou xifres decimals) i calcular-ne la funció φ d'Euler.

De la mateixa manera que en el QGT 03, s'ha desenvolupat un entorn senzill, basat en botons i gràfiques, perquè l'usuari executi l'algorisme pas a pas i entengui les diferents etapes del procés. Addicionalment, la informació d'una iteració de l'algorisme (incloent-hi la gràfica de probabilitat després de la QFT) es pot registrar en un fitxer PDF i enviar-lo per correu.

De nou, l'objectiu d'aquesta aplicació és difondre el potencial de la computació quàntica i motivar els joves científics i enginyers. En particular, aquest algorisme és un bon exemple de la necessitat de dominar diferents branques del coneixement (educació interdisciplinària) per arribar a desenvolupar algorismes innovadors.

Com que el QGT 04 tanca la col·lecció d'aplicacions d'aquest projecte, s'hi ha afegit una petita presentació de l'autor en una de les vistes a tall d'introducció al projecte QGT. S'espera que aquestes quatre siguin les precursores de futures aplicacions sobre computació quàntica i codificació de grafs.

10.2 Descripció

En aquesta aplicació s'ha programat l'entorn d'usuari mitjançant una estructura de tipus *TabBarController* amb quatre vistes. Es tracta d'una classe que permet mantenir diferents controladors de vista actius i passar de l'un a l'altre mitjançant botons de selecció situats a la part inferior de la pantalla. Es va seleccionar aquesta estructura perquè permet a l'usuari deixar l'algorisme a mitges i consultar la informació de la vista d'aritmètica. A més, en futures actualitzacions, està previst incloure una cinquena vista amb més informació teòrica per fer més complet el simulador.

La primera vista, *Intro*, és una presentació de l'aplicació i del circuit, que té com a única funció introduir-los (veure figura 10.1). L'última, *About*, com es comenta en l'apartat anterior, correspon a una presentació de l'autor i del projecte QGT.

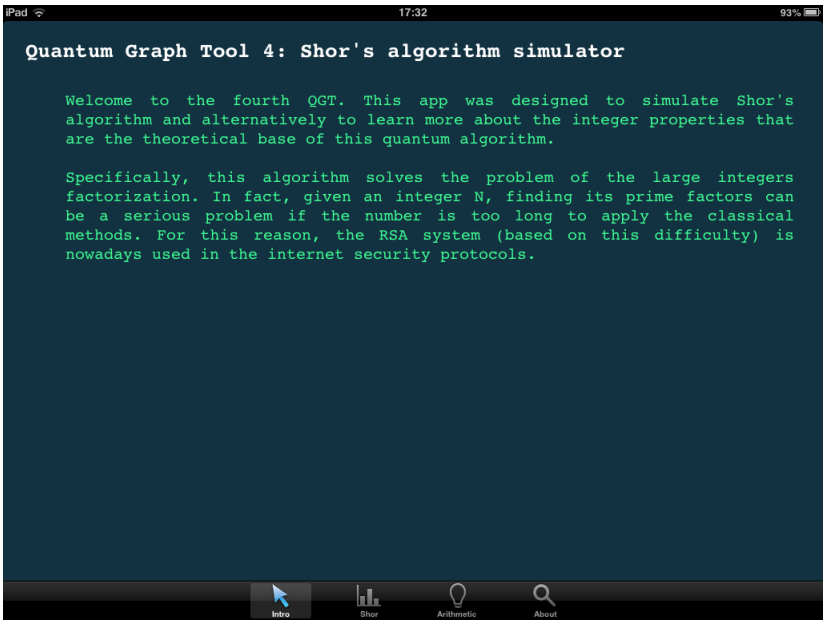

Figura 10.1: Vista *Intro* del QGT 04

La segona vista, *Shor*, és la que correspon al simulador de l'algorisme de Shor (veure figura 10.2). Es pot escollir l'element buscat utilitzant un selector de classe *PickerView* com el del QGT 03. Entre els nombres que apareixen en aquest selector hi ha nombres primers i nombres parells (per comprovar el comportament del sistema quan s'introdueix un nombre inadequat). Tot i això, aquesta llista es pot modificar mitjançant l'interruptor *Extended Array*, perquè només mostri els nombres compostos imparells. L'usuari pot escollir executar tot l'algorisme directament si desactiva l'interruptor *step by step mode*; de la mateixa manera, es pot millorar la velocitat de processament si es desactiva l'interruptor de les gràfiques de probabilitat, les quals es mostren al final de l'algorisme. Les gràfiques de probabilitat mostren la informació més rellevant de l'algorisme; això és, l'estat del registre d'entrada després d'aplicar la QFT. Les deltes que apareixen corresponen, tal com es veu en el bloc teòric II, als valors més probables del registre d'entrada, els quals es poden utilitzar en l'algorisme de les fraccions contínues.

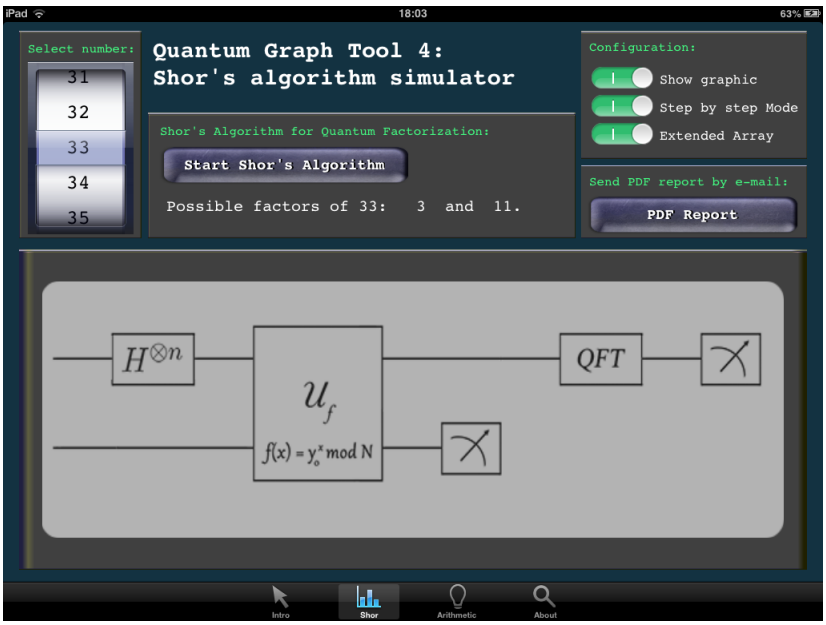

Figura 10.2: Vista *Shor* del QGT 04

Si s'executa l'algorisme en el mode pas a pas (*step by step mode*), apareix una vista amb el circuit complet que guia l'usuari fins al final de l'algorisme. En la figura següent, es mostra el pas 5 de l'algorisme amb les opcions que té l'usuari.

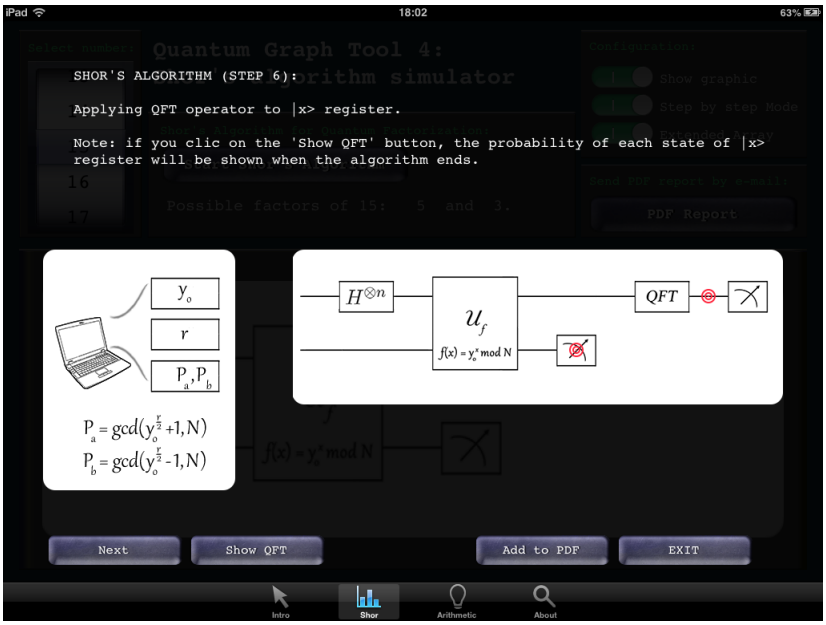

Figura 10.3: Algorisme de Shor en procés en el QGT 04

Quan l'algorisme finalitza, l'aplicació torna a la vista de l'algorisme de Shor i mostra els resultats per pantalla. En la figura següent es pot veure el resultat d'aplicar l'algorisme per factoritzar el nombre 15, de manera que s'obtenen els dos factors: 3 i 5, juntament amb les gràfiques de probabilitat.

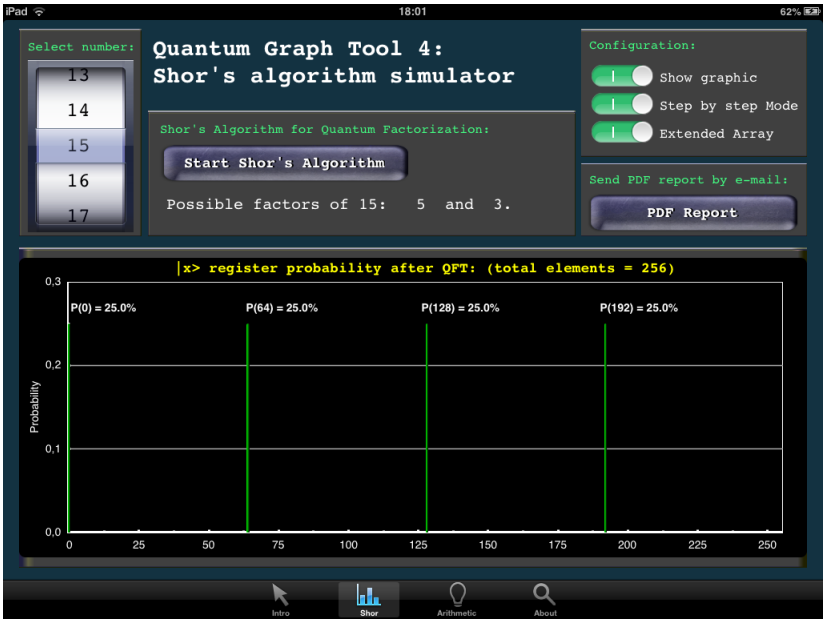

Figura 10.4: Resultat d'aplicar l'algorisme per factoritzar el 15 en el QGT 04.

La tercera vista, *Arithmetics*, és la que correspon a la factorització clàssica. En aquesta pantalla hi ha un selector gran de classe *PickerView* amb nou columnes que permeten escollir un nombre enter positiu menor que 10^9. Mitjançant el botó *Factorize Number* s'executa un mètode clàssic per factoritzar el nombre i calcular la φ d'Euler, tal com es mostra en la figura següent:

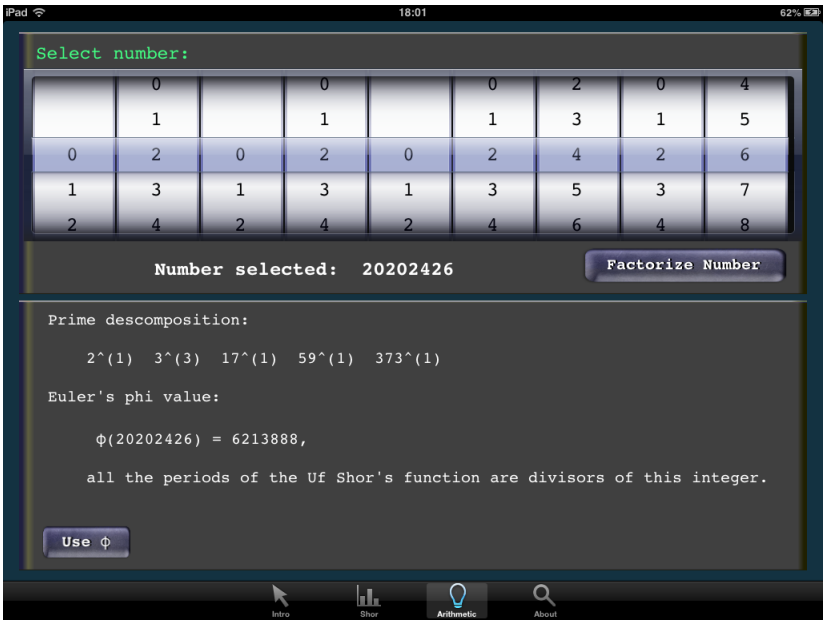

Figura 10.5: Vista *Arithmetic* del QGT 04

A més, apareix un nou botó, *Use* ϕ, que permet modificar el selector automàticament i així factoritzar el valor de la φ d'Euler del nombre anterior.

10.3 Estructura

A banda de les llibreries bàsiques i de la llibreria libCorePlot de Google per a les gràfiques, el QGT 04 presenta vuit classes d'objectes que segueixen l'esquema MVC. Com en el cas del simulador de Grover, com que la càrrega del circuit de Shor és bàsicament computacional, per tal d'optimitzar-ne l'eficiència, s'utilitza una única classe que executa tota la simulació.

A continuació, es detallen les vuit classes que s'han programat en el QGT 04.

- **AppDelegate:** és el nucli de l'aplicació i, a més, delegat d'un *UITabBarController*. Per aquest motiu, a diferència dels *AppDelegate* anteriors, a més de contenir i crear l'objecte *ViewController*, en crea tres més: *ShorViewController*, *ArithmeticViewController* i *AboutViewController*. Afageix les quatre vistes instanciades a l'atribut *UITabBarController* i s'activa (es mostra) la del *ViewController* per defecte.

- **ViewController:** és el controlador de la vista de presentació. Només conté dos atributs de text, un per al títol i l'altre per a la informació inicial.

- **ShorViewController:** és el controlador de la vista del simulador. Es tracta d'un objecte que implementa molts protocols de dades i que és delegat de sis classes addicionals per poder gestionar tota la informació: alertes, composició de correus, PDF, gràfics, selectors... Conté, entre d'altres: l'objecte *ShorCircuit*, el *SavePDF* i les *arrays* d'informació associades, la gràfica *CorePlot* i els vectors de resultats, i les vistes del mode pas a pas.

- **ArithmeticViewController:** és l'equivalent del *ShorViewController* per a la vista d'aritmètica. En aquest cas, però, només és delegat de dues classes i el nombre d'atributs i mètodes és molt més reduït. Conté l'objecte de classe *ArithmeticTest* per factoritzar clàssicament.

- **AboutViewController:** és el controlador de la vista de *About*. Només conté dos atributs de text i una imatge de classe *UIImageView*.

- **ShorCircuit:** és l'objecte que representa el model del circuit de Shor. És, com en el cas del *CircuitViewController* del QGT 03, un objecte molt gran que conté quinze atributs i disset mètodes. Implementa tots els passos clàssics i quàntics (simulats) de l'algorisme de Shor i un conjunt de mètodes de test per verificar el comportament correcte. Aquest objecte també és limitant a causa de la dificultat de codificar els estats i el gran nombre d'operacions que ha d'executar (temps computacional).

- **ArithmeticTests:** és un objecte que conté mètodes clàssics de factorització i càlcul de la φ d'Euler. També conté un objecte *PickerView* de nou columnes per seleccionar l'enter a factoritzar.

- **SavePDF:** és l'objecte encarregat de crear, editar i enviar per correu els arxius en format PDF que inclouen la informació de la simulació i la gràfica de probabilitat. Es tracta d'una versió més completa de l'objecte *SavePDF* del QGT 01.

En la figura 10.6 es pot veure representat l'esquema jeràrquic d'aquests objectes.

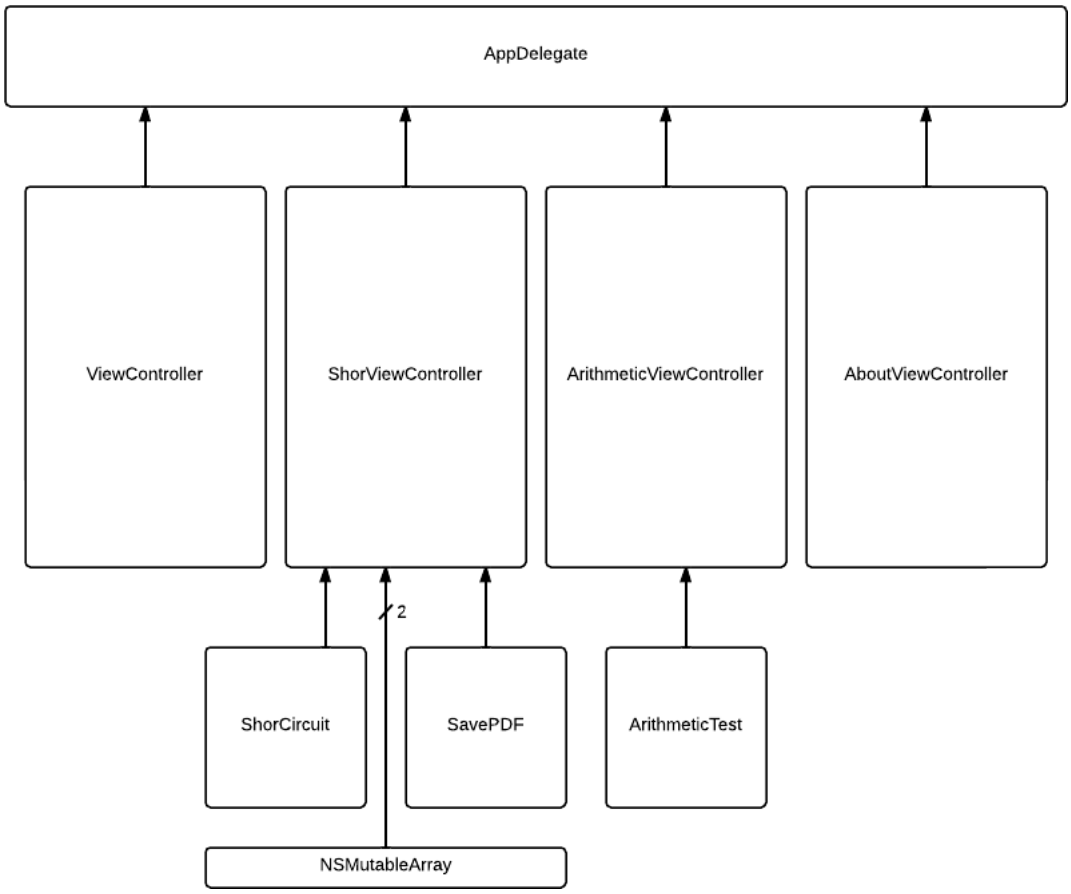

Figura 10.6: Esquema dels objectes del QGT 04

A més, per tal de preparar l'aplicació per a futures actualitzacions, s'han afegit alguns atributs de l'*AppDelegate* per actuar com a pont (*bridge*) entre els diferents controladors de vistes.

10.4 Implementació

De la mateixa manera que en els QGT anteriors, en aquest apartat només es comenta el codi relacionat amb els algorismes (la gestió de les vistes i els objectes, els protocols d'intercanvi de dades i la delegació de classes no són tractats per la naturalesa del projecte).

Els mètodes *algorithm* i *algorithm_step* de la classe *ShorCircuit* són els més importants de l'aplicació. De fet, aquests mètodes són conceptualment el mateix, però el primer optimitza la velocitat de l'algorisme (sense accedir a la vista del controlador), mentre que el segon intenta mostrar tota la informació possible a l'usuari (mode pas a pas). Com que l'estructura del mètode és molt semblant a la descripció que s'ha fet en el bloc teòric II, a continuació s'expliquen les funcions que va executant.

En primer lloc, es busca el valor de y_0 aleatori. El bucle *do-while* descarta aquells valors que farien que l'algorisme acabés, ja que interessa que l'usuari obtingui una simulació de l'algorisme. Això no es faria en una aplicació real de l'algorisme, tot i que, de fet, seria una gran sort obtenir un dels factors a l'atzar abans de començar.

```
//Es busca un valor aleatòri per a y.
do{
    y_0 = rand() % (number-2);
    y_0 = y_0 + 2;
}while([self euclides_mcd:number :y_0] != 1);
```

El mètode *euclides_mcd* retorna el *gcd* dels paràmetres: *number* —el nombre que es vol factoritzar— i y_0. A continuació, es calcula el nombre de qubits necessaris del registre d'entrada i sortida utilitzant el mètode *getDimension*.

```
dim_in = [self getDimension:number*number];
dim_out = [self getDimension:number];

//Es crea i s'inicialitza el registre de dim_in + dim_out qubits a 0.
N_in = 1;
N_out = 1;

for (int i = 0; i < dim_in; i++) {
    N_in = N_in*2;
}
for (int i = 0; i < dim_out; i++) {
    N_out = N_out*2;
}
```

A N_in i N_out es guarden els elements dels registres i s'utilitzen per iniciar les dades dels coeficients complexos.

```
for (int j = 0; j < N_in*N_out; j++) {
    quantum_register[j][real] = 0;
    quantum_register[j][imag] = 0;
}
quantum_register[0][real] = 1;
[self Test_Register];
```

El mètode *Test_Register* s'utilitza per comprovar que el registre complet s'ha creat correctament. Un cop inicialitzat, toca aplicar la porta Hadamard per crear la superposició. El mètode *Hadamard_Tensor_2* només calcula la variable k de la matriu Hadamard —com en el QGT 03—, ja que també és parteix del supòsit que el registre es troba a l'estat $|0\rangle$ i, posteriorment, es propaga aquest coeficient.

```
//S'aplica hadamard al registre d'entrada.
 S_factors.text = @"Applying Hadamard operator to |x> register.";
if (N_in <= Nmax) {
    [self Hadamard_Tensor_2:N_in];//Només ens cal el factor
}else{
    NSLog(@"ERROR: es necessiten %d qubits.",dim_in);
    return 0;
}
[self Test_Hadamard];

f_aux = 1/sqrt(kHadamard);
for (int j = 0; j < N_in; j++) {
    quantum_register[j*N_out][real] = f_aux;
}
```

Es pot veure que per crear la superposició s'ha anat propagant el valor del coeficient obtingut de la porta Hadamard als valors del registre que toquen (cada N_out elements). Així s'obté una superposició de tots els valors possibles d'entrada entrellaçats amb un zero a la sortida. El pas següent és calcular la funció exponencial.

```
//S'aplica la funció exponencial: y^x mod number. (Actuació sobre cada estat del
    registre d'entrada).
S_info.text = @"Applying Uf operator to |x>|f(x)> register.";
for (int j=0; j<N_in; j++) {
    //Es calcula f(x).
    aux = [self exp_entera:y_0 exponent:j modul:number];
    //S'entrellaça |x> amb |f(x)> (encara no hi ha part imaginaria).
    quantum_register[j*N_out + aux][real] = quantum_register[j*N_out][real];
    quantum_register[j*N_out][real] = 0;
}
[self Test_Register];
```

Aquest bucle *for* genera l'estat $|x\rangle\,|y_0^x \; mod \; number\rangle$. Pel que fa a les variables S_info i $S_factors$ que van apareixent, són atributs de la classe *ShorCircuit*, els quals apunten a objectes del controlador de vista per mostrar la informació a l'usuari. A continuació s'ha de simular la mesura del registre de sortida; per tant, és necessari calcular la probabilitat de cada estat.

```
//Es calcula la probabilitat de cada valor de sortida
for (int i = 0; i<N_out; i++) {
    prob_sortida[i] = 0;
}
for (int i = 0; i < N_in*N_out; i++) {
    prob_sortida[i%N_out] = prob_sortida[i%N_out] + (quantum_register[i][0]*
        quantum_register[i][0]+quantum_register[i][1]*quantum_register[i][1]);
}
for (int i = 0; i<N_out; i++) {
    if (prob_sortida[i] != 0) {
        NSLog(@"sortida = %d amb prob. = %f",i,prob_sortida[i]);
        prob_sortida[i] = prob_sortida[i]*1000000;
    }
}
```

En el vector $prob_sortida$ hi ha la probabilitat de cada estat multiplicada per un milió.[1] Per implementar la mesura es genera un nombre aleatori, aux, entre 1 i un milió, de manera que, en cada iteració i, es compara amb la probabilitat de l'estat $|i\rangle$; si és inferior, s'obté l'estat mesurat i es guarda a q_out; si no ho és, es resta la probabilitat de l'estat al valor actual de la variable aux.

```
//Es mesura la sortida i es guarda el valor en q_out.
aux = rand()%1000000 + 1;
NSLog(@"Mesura de la sortida: aleatori entre 1 i 1.000.000 = %d",aux);
S_info.text = @"Measuring |f(x)> register.";
for (int i = 0; i < N_out; i++) {
    if (prob_sortida[i]!=0) {
        aux = aux - prob_sortida[i];
        if (aux <= 0) {
            q_out = i;
            NSLog(@"La mesura de la sortida és %d",q_out);
            S_info.text = [NSString stringWithFormat:@"%@\n|%d> measured.",
                S_info.text, q_out];
            break;
        }
    }
}
```

[1] L'elecció del valor d'un milió s'ha fet a partir dels valors que pren la probabilitat en el pitjor cas. De fet, amb 10.000 ja s'obté un error molt petit pels valors de l'aplicació.

Un cop mesurada la sortida, s'ha de simular la nova distribució de probabilitat dels registres d'entrada i sortida pel fet que la sortida ha col·lapsat a *q_out*.

```
//Es reparteix la probabilitat entre els estats d'entrada que poden donar aquest
    valor de sortida
f_aux = 0;
for (int i = 0; i < N_in*N_out; i++) {
    if (i%N_out == q_out) {//Entrades possibles
        f_aux = f_aux + (quantum_register[i][0]*quantum_register[i][0]+
            quantum_register[i][1]*quantum_register[i][1]);
    }else{
        quantum_register[i][real] = 0;
        quantum_register[i][imag] = 0;
    }
}
f_aux = 1/sqrtf(f_aux);//La probabilitat correspon al quadrat dels coeficients.
for (int i = 0; i < N_in*N_out; i++) {
    if (quantum_register[i][real] != 0 || quantum_register[i][imag] != 0) {
        quantum_register[i][real] = quantum_register[i][real]*f_aux;
        quantum_register[i][imag] = quantum_register[i][imag]*f_aux;
    }
}
[self Test_Register];
```

A continuació, s'aplica la transformada quàntica de Fourier al registre d'entrada mitjançant el mètode *quantum_fourier_transform_2* que aplica els productes matricials de manera recursiva.

```
//S'aplica la QFT sobre el registre d'entrada.
NSLog(@"%d",N_in);
float aux_a = 1/sqrtf(N_in);//constant
NSLog(@"%f",aux_a);
S_info.text = @"Applying QFT operator to |x> register.";
[self quantum_fourier_transform_2: N_in: N_out:q_out];
[self Test_Register];
```

El mètode *quantum_fourier_transform_2* es correspon amb la versió iterada que es mostra a continuació.[2]

```
- (void) quantum_fourier_transform_2:(int)i_size:(int)o_size:(int)y{

    NSString *line;

    float aux_vect[i_size][2];
    float aux_b = 1/sqrtf(i_size);//constant

    //Es copia el registre d'entrada
    for (int i = 0; i < i_size; i++) {
        aux_vect[i][real] = quantum_register[i*o_size+y][real]*aux_b;
        aux_vect[i][imag] = quantum_register[i*o_size+y][imag]*aux_b;
    }

    aux_b = aux_b*aux_b;

    //Es realitza el producte matricial dels coeficients complexes
    for (int fil = 0; fil < i_size; fil++){
        line = @"";
        quantum_register[fil*o_size+y][real] = 0;
        quantum_register[fil*o_size+y][imag] = 0;
        for (int col = 0; col < i_size; col++) {
            //Part real dels coeficients
            quantum_register[fil*o_size+y][real] += cosf(2*M_PI*aux_b*(fil*col))*aux_vect[col][real];
            quantum_register[fil*o_size+y][real] += -1*sinf(2*M_PI*aux_b*(fil*col))*aux_vect[col][imag];
            //Part imag dels coeficients
            quantum_register[fil*o_size+y][imag] += cosf(2*M_PI*aux_b*(fil*col))*aux_vect[col][imag];
            quantum_register[fil*o_size+y][imag] += sinf(2*M_PI*aux_b*(fil*col))*aux_vect[col][real];
        }
    }
}
```

[2] El mètode *quantum_fourier_transform*, en canvi, és una versió que genera tota la matriu a l'inici.

A continació es mesura el registre d'entrada aplicant la mateixa estratègia que en el cas anterior (sortida).

```
//Es calcula la probabilitat de cada valor d'entrada
for (int i = 0; i<N_in; i++) {
    prob_entrada[i] = 0;
}
for (int i = 0; i < N_in*N_out; i++) {
    prob_entrada[i/N_out] = prob_entrada[i/N_out] + (quantum_register[i][0]*
        quantum_register[i][0]+quantum_register[i][1]*quantum_register[i][1]);
}
for (int i = 0; i<N_in; i++) {
    if (prob_entrada[i] != 0) {
        prob_entrada[i] = prob_entrada[i]*1000000;
    }
}

//Es mesura l'entrada.
S_info.text = @"Measuring |x> register.";
aux = rand()%1000000 + 1;
NSLog(@"Mesura de l'entrada: aleatori entre 1 i 1.000.000 = %d",aux);
for (int i = 0; i < N_in*N_out; i++) {
    if (prob_entrada[i]!=0) {
        aux = aux - prob_entrada[i];
        if (aux <= 0) {
            q_in = i;

            NSLog(@"La mesura de l'entrada és %d",q_in);
            S_info.text = [NSString stringWithFormat:@"%@\n|%d> measured.",
                S_info.text ,q_in];
            break;
        }
    }
}
//Si ha fallat acava.
if (q_in == 0) {
    S_factors.text = @"Algorithm failed. |0> measured.";
    return 1;
}
```

Si la mesura de l'entrada no ha estat $|0\rangle$ (codi de finalització 1), s'aplica l'algorisme de les fraccions contínues.

```
//Si no ha fallat, algorisme de fraccions continues.
r = [self algoritme_fraccions_continues:q_in :N_in: y_0: number];
period = r;

S_info.text = @"Searching the period with classical continued fractions
    algorithm.";
S_info.text = [NSString stringWithFormat: @"%@\nPeriod found: r = %d.", S_info.
    text, r];
//Si el periode r és zero --> Ha fallat l'algorisme de fraccions continues
if(r == 0){
    NSLog(@"No s'ha trobat el període. Torna-ho a intentar.");
    S_factors.text = @"Period not found. Please, try again.";
    return 2;
}

//Si el periode r és imparell --> falla.
if(r%2 == 1){
    NSLog(@"Període imparell. Torna-ho a intentar.");
    S_factors.text = [NSString stringWithFormat: @"r = %d, Odd period. Please,
        try again.", r];
    return 3;
}
```

En cas que el mètode *algoritme_fraccions_contiues* no trobi el període i retorni un zero (codi de finalització 2) o el període retornat sigui imparell (codi de finalització 3), s'acaba la simulació i s'escriu l'error per pantalla.

Si s'ha trobat un període parell, s'aplica el mètode clàssic per trobar els dos factors i finalitza la simulació.

```
//Si el periode r és parell --> algorisme clàssic.
r = period/2;

n_0 = [self exp_entera:y_0 exponent:r modul:number];

n_0 = (n_0 + 1) % number;
n_1 = (n_0 + number - 2) % number;

n_0 = [self euclides_mcd:number :n_0];
n_1 = [self euclides_mcd:number :n_1];
```

Com s'ha vist, tots aquests passos es corresponen perfectament amb la descripció teòrica de l'algorisme. Cal dir, però, que el fet que se simuli en un ordinador clàssic, emmascara la potència de l'algorisme. Per exemple, el repartiment de probabilitats després de la mesura que sembla tan costós, és un pas immediat si s'executa en un ordinador quàntic. De la mateixa manera, l'operador que calcula totes les sortides y_o^x *mod number* s'aplicaria automàticament a tot el registre amb una sola crida. El lector ha de ser conscient de la limitació que suposa simular-ho mitjançant la computació clàssica i comprendre'n les conseqüències.

Un altre bloc interessant d'aquesta aplicació és el de la factorització clàssica. De nou, per no allargar el text innecessàriament, s'explica únicament el mètode principal. El mètode més interessant de la classe *ArithmeticTests* és el que s'encarrega de factoritzar i calcular la φ d'Euler (*prime_factors_ descomposition_ of*), el qual es detalla a continuació.

En primer lloc, es crea una llista de nombres primers que es van guardant en un vector. Per determinar si un nombre és primer, s'utilitza precisament la llista que es va generant: si un nombre no és divisible per cap dels primers menors que ell, aleshores és primer.

```
//Es busquen els nombres primers
prime_descom[pos][0] = 2;
prime_descom[pos][1] = 0;
pos++;

for (int i = 3; i < maxArithPrimeRoot; i++) {
    isPrime = YES;
    for (int j = 0 ; j < pos; j++) {
        if (i%prime_descom[j][0] == 0) {
            isPrime = FALSE;
            break;
        }
    }
    if (isPrime) {
        prime_descom[pos][0] = i;
        prime_descom[pos][1] = 0;
        pos++;
        if (prime_descom[pos][0] > sqrt(num)) {
            break;
        }
    }
}
```

Per cada primer trobat, es posa un comptador a zero que serveix per indicar que encara no és un dels factors del nombre objectiu, *num*. Per acabar, si s'arriba a un nombre més gran que l'arrel de *num*, ja s'han trobat prous primers per descomposar-lo i se surt del bucle.

A continuació, es factoritza el nombre mitjançant el mètode *descomposition_ of* que s'encarrega de calcular la *valoració p-àdica*[3] de *num* i es guarda en el comptador del primer, *prime_descom*[*p*][1]:

```
//Es factoritza el nombre
for (int k = 0; k < pos; k++) {
    prime_descom[k][1] = [self descomposition_of:num prime:prime_descom[k][0]];

    if (prime_descom[k][1] != 0) {
        for (int n = 0; n < prime_descom[k][1]; n++) {
            FactorsTrobats = FactorsTrobats*prime_descom[k][0];
        }
    }
}
```

A més, es van acumulant els factors que es van trobant en la variable $FactorsTrobats$. Si en acabar la llista aquesta variable no coincideix amb el nombre, falta un factor primer que correspon al quocient d'aquests dos (en particular si el nombre buscat és un nombre primer).

```
if (FactorsTrobats != num) {
    FactorsTrobats = num/FactorsTrobats;
    S_info.text = [NSString stringWithFormat:@"%@  %d^(1)",S_info.text, FactorsTrobats];
    NSLog(@"%d", FactorsTrobats);
    pos++;
}

numFactors = pos;//Es guarda el nombre de factors
```

Finalment, es calcula la φ d'Euler a partir de la descomposició que s'ha anat guardant en els comptadors. La variable *phi* s'ha inicialitzat a 1 i, en cas que el nombre objectiu sigui primer, se salta el primer bucle i entra directament als condicionals.

```
//Es calcula phi
for (int k = 0; k < pos; k++) {
    if (prime_descom[k][1] == 1) {
        phi = phi*(prime_descom[k][0] - 1);
    }else if (prime_descom[k][1] > 1){
        phi = phi*(prime_descom[k][0] - 1);
        for (int m = 1; m < prime_descom[k][1]; m++) {
            phi = phi*prime_descom[k][0];
        }
    }
}
if (FactorsTrobats != num) {
    phi = phi*(FactorsTrobats-1);
}

//Si és primer
if (phi == 1 && num != 2) {
    phi = num-1;
}
```

Aquest mètode funciona bé si el nombre no té gaires xifres, però el nombre d'iteracions dels bucles es torna prohibitiu per a nombres grans com els que s'utilitzen en RSA.

[3] Correspon a l'exponent del primer p, en la descomposició en factors primers d'un nombre enter.

10.5 Presentació

Aquesta aplicació es va penjar a principis de juny del 2013. Com que va coincidir amb la setmana de la WWDC[4] d'Apple i la corresponent presentació del nou sistema operatiu iOS 7,[5] es va allargar el temps que l'aplicació va estar en l'estat *In Review*. Tot i això, va passar el test sense problemes i ja es troba activa a l'App Store.

En aquest cas, es va utilitzar un únic simulador físic i el simulador virtual d'iPad 3 per provar el rendiment en les diferents plataformes.

En la figura següent, es mostra la informació general que apareix en el web *iTunes Connect* després de passar el procés d'avaluació:

Figura 10.7: Informació de la versió penjada del QGT 04

Pel que fa a la informació detallada de la versió actual, és recollida en la figura següent:

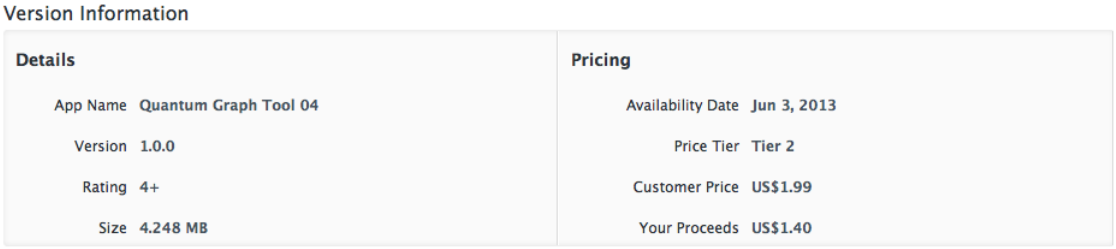

Figura 10.8: Detalls de la versió penjada del QGT 04

Es pot veure que, en aquest cas, el preu pertany a la segona categoria de preus, ja que el temps i l'esforç dedicats a aquesta aplicació han estat molt superiors a la resta (tot i disposar de més experiència). Tot i això, segueix essent un preu simbòlic (inferior a dos euros) per tal que l'usuari valori l'aplicació. D'altra banda, s'aprecia un increment important de la mida del binari (entre el doble i el triple que les anteriors) a causa de les imatges i de les classes addicionals que s'hi han inclòs.

[4] Worldwide Developers Conference, una conferència mundial per a desenvolupadors que se celebra anualment.

[5] L'aplicació també serà compatible amb aquest nou sistema, tot i que encara no es troba disponible per a iPad. Actualment, només hi ha una versió beta de iOS 7 per iPhone.

Finalment, es presenta l'aspecte de l'última aplicació del projecte a l'App Store:

Figura 10.9: Dades del QGT 04 a l'App Store

10.6 Simulacions

A continuació, es recullen alguns dels resultats de les simulacions de l'algorisme amb l'aplicació QGT 04. En les taules apareixen, per ordre: els y_0 aleatoris, els l_j mesurats després de la QFT, el període de la funció exponencial i els factors retornats.

15

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
7	$\|192\rangle$	4	5 i 3	2	$\|64\rangle$	4	5 i 3
4	$\|0\rangle$	-	-	4	$\|128\rangle$	2	5 i 3
4	$\|128\rangle$	2	5 i 3	11	$\|0\rangle$	-	-
2	$\|192\rangle$	4	5 i 3	8	$\|0\rangle$	-	-
7	$\|64\rangle$	4	5 i 3	4	$\|128\rangle$	2	5 i 3
8	$\|192\rangle$	4	5 i 3	14	$\|128\rangle$	2	15 i 1

Taula 10.1: Simulacions del QGT 04 per factoritzar el 15

18

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
7	$\|0\rangle$	-	-	17	$\|256\rangle$	2	18 i 2
13	$\|341\rangle$	3 (odd)	-	7	$\|0\rangle$	-	-
11	$\|341\rangle$	*not found*	-	7	$\|341\rangle$	3 (odd)	-

Taula 10.2: Simulacions del QGT 04 per factoritzar el 18

21

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
13	$\|256\rangle$	2	7 i 3	4	$\|0\rangle$	-	-
17	$\|342\rangle$	*not found*	-	20	$\|256\rangle$	2	21 i 1
11	$\|341\rangle$	*not found*	-	16	$\|341\rangle$	3 (*odd*)	-
5	$\|84\rangle$	6	21 i 1	4	$\|342\rangle$	3 (*odd*)	-
10	$\|0\rangle$	-	-	11	$\|83\rangle$	6	3 i 7
8	$\|256\rangle$	2	3 i 7	5	$\|85\rangle$	6	21 i 1
2	$\|171\rangle$	*not found*	-	13	$\|0\rangle$	-	-
16	$\|0\rangle$	-	-	10	$\|427\rangle$	6	7 i 3
2	$\|427\rangle$	6	3 i 7	8	$\|0\rangle$	-	-

Taula 10.3: Simulacions del QGT 04 per factoritzar el 21

24

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
19	$\|512\rangle$	2	4 i 6	5	$\|512\rangle$	2	6 i 4
17	$\|0\rangle$	-	-	11	$\|0\rangle$	-	-
7	$\|0\rangle$	-	-	23	$\|512\rangle$	2	24 i 2
17	$\|512\rangle$	2	6 i 8	19	$\|512\rangle$	2	4 i 6
23	$\|0\rangle$	-	-	11	$\|512\rangle$	2	12 i 2
19	$\|0\rangle$	-	-	19	$\|512\rangle$	2	4 i 6

Taula 10.4: Simulacions del QGT 04 per factoritzar el 24

25

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
11	$\|410\rangle$	5 (*odd*)	-	12	$\|461\rangle$	20	25 i 1
14	$\|717\rangle$	10	25 i 1	14	$\|205\rangle$	*not found*	-
12	$\|717\rangle$	*not found*	-	4	$\|307\rangle$	10	25 i 1
18	$\|768\rangle$	4	25 i 1	8	$\|563\rangle$	20	25 i 1
7	$\|256\rangle$	4	25 i 1	3	$\|768\rangle$	*not found*	-

Taula 10.5: Simulacions del QGT 04 per factoritzar el 25

27

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
16	$\|569\rangle$	9 (*odd*)	-	13	$\|455\rangle$	9 (*odd*)	-
8	$\|512\rangle$	*not found*	-	22	$\|796\rangle$	9 (*odd*)	-
2	$\|0\rangle$	-	-	4	$\|336\rangle$	*not found*	-
11	$\|57\rangle$	18	27 i 1	8	$\|855\rangle$	6	27 i 1
13	$\|567\rangle$	9 (*odd*)	-	13	$\|114\rangle$	9 (*odd*)	-
10	$\|683\rangle$	3 (*odd*)	-	2	$\|512\rangle$	*not found*	-

Taula 10.6: Simulacions del QGT 04 per factoritzar el 27

	y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
30	29	$\|512\rangle$	2	30 i 2	19	$\|0\rangle$	-	-
	19	$\|512\rangle$	2	10 i 6	7	$\|768\rangle$	4	10 i 6
	7	$\|512\rangle$	*not found*	-	23	$\|768\rangle$	4	10 i 6
	29	$\|0\rangle$	-	-	7	$\|256\rangle$	4	10 i 6

Taula 10.7: Simulacions del QGT 04 per factoritzar el 30

	y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
33	25	$\|1229\rangle$	5 (*odd*)	-	19	$\|1435\rangle$	10	11 i 3
	20	$\|1229\rangle$	*not found*	-	4	$\|390\rangle$	5 (*odd*)	-
	19	$\|1431\rangle$	10	11 i 3	31	$\|1229\rangle$	5 (*odd*)	-
	4	$\|1639\rangle$	5 (*odd*)	-	4	$\|819\rangle$	5 (*odd*)	-
	17	$\|1229\rangle$	*not found*	-	32	$\|1024\rangle$	2	33 i 1
	16	$\|1638\rangle$	5 (*odd*)	-	13	$\|205\rangle$	10	11 i 3

Taula 10.8: Simulacions del QGT 04 per factoritzar el 33

	y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
35	24	$\|1365\rangle$	*not found*	-	29	$\|0\rangle$	-	-
	24	$\|0\rangle$	-	-	6	$\|1024\rangle$	2	7 i 5
	27	$\|1024\rangle$	*not found*	-	13	$\|1536\rangle$	4	5 i 7
	8	$\|512\rangle$	4	5 i 7	33	$\|853\rangle$	12	5 i 7
	6	$\|0\rangle$	-	-	13	$\|512\rangle$	4	5 i 7
	9	$\|0\rangle$	-	-	8	$\|0\rangle$	-	-
	2	$\|683\rangle$	*not found*	-	16	$\|1364\rangle$	3 (*odd*)	-
	34	$\|0\rangle$	-	-	2	$\|171\rangle$	12	5 i 7

Taula 10.9: Simulacions del QGT 04 per factoritzar el 35

	y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
39	29	$\|341\rangle$	6	3 i 13	35	$\|1706\rangle$	6	3 i 13
	8	$\|512\rangle$	4	13 i 3	23	$\|1024\rangle$	*not found*	-
	2	$\|341\rangle$	*not found*	-	16	$\|1364\rangle$	3 (*odd*)	-
	4	$\|1366\rangle$	*not found*	-	29	$\|682\rangle$	*not found*	-
	10	$\|342\rangle$	6	13 i 3	37	$\|853\rangle$	12	13 i 3
	16	$\|0\rangle$	-	-	34	$\|0\rangle$	-	-
	11	$\|1536\rangle$	*not found*	-	4	$\|683\rangle$	*not found*	-
	22	$\|0\rangle$	-	-	38	$\|1024\rangle$	2	39 i 1

Taula 10.10: Simulacions del QGT 04 per factoritzar el 39

42

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
11	$\|1705\rangle$	6	6 i 14	31	$\|1707\rangle$	6	14 i 6
25	$\|683\rangle$	3 (*odd*)	-	37	$\|0\rangle$	-	-
29	$\|1024\rangle$	2	6 i 14	11	$\|337\rangle$	6	6 i 14
19	$\|341\rangle$	6	14 i 6	29	$\|0\rangle$	-	-
23	$\|341\rangle$	6	6 i 14	17	$\|341\rangle$	6	42 i 2
5	$\|340\rangle$	6	42 i 2	37	$\|683\rangle$	3 (*odd*)	-

Taula 10.11: Simulacions del QGT 04 per factoritzar el 42

45

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
19	$\|1024\rangle$	2	5 i 9	26	$\|0\rangle$	-	-
22	$\|1195\rangle$	12	5 i 9	31	$\|0\rangle$	-	-
23	$\|512\rangle$	*not found*	-	44	$\|0\rangle$	-	-
16	$\|1366\rangle$	3 (*odd*)	-	32	$\|1536\rangle$	*not found*	-
38	$\|682\rangle$	*not found*	-	37	$\|1536\rangle$	4	5 i 9
4	$\|0\rangle$	-	-	31	$\|1366\rangle$	3 (*odd*)	-
44	$\|1024\rangle$	2	45 i 1	41	$\|341\rangle$	6	9 i 5

Taula 10.12: Simulacions del QGT 04 per factoritzar el 45

51

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
31	$\|3584\rangle$	*not found*	-	11	$\|2560\rangle$	*not found*	-
41	$\|1280\rangle$	16	17 i 3	44	$\|1024\rangle$	*not found*	-
25	$\|3584\rangle$	8	17 i 3	13	$\|1024\rangle$	4	17 i 3
38	$\|2048\rangle$	*not found*	-	10	$\|1792\rangle$	16	17 i 3
49	$\|3584\rangle$	8	17 i 3	47	$\|1024\rangle$	4	17 i 3
35	$\|0\rangle$	-	-	41	$\|1024\rangle$	*not found*	-
16	$\|0\rangle$	-	-	46	$\|3840\rangle$	16	17 i 3

Taula 10.13: Simulacions del QGT 04 per factoritzar el 51

55

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
41	$\|3690\rangle$	10	11 i 5	29	$\|821\rangle$	*not found*	-
38	$\|1229\rangle$	*not found*	-	46	$\|3277\rangle$	*not found*	-
28	$\|2048\rangle$	*not found*	-	19	$\|0\rangle$	-	-
14	$\|3687\rangle$	10	5 i 11	29	$\|819\rangle$	*not found*	-
47	$\|1843\rangle$	20	5 i 11	28	$\|1843\rangle$	20	5 i 11
19	$\|3277\rangle$	*not found*	-	6	$\|409\rangle$	10	11 i 5

Taula 10.14: Simulacions del QGT 04 per factoritzar el 55

57

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
46	$\|684\rangle$	6	19 i 3	55	$\|455\rangle$	9 (*odd*)	-
2	$\|2731\rangle$	*not found*	-	29	$\|1138\rangle$	18	57 i 1
49	$\|0\rangle$	-	-	47	$\|3404\rangle$	*not found*	-
52	$\|0\rangle$	-	-	49	$\|2731\rangle$	3 (*odd*)	-
41	$\|2730\rangle$	*not found*	-	26	$\|3413\rangle$	6	3 i 19
41	$\|0\rangle$	-	-	13	$\|1138\rangle$	18	19 i 3
44	$\|2048\rangle$	*not found*	-	32	$\|2048\rangle$	*not found*	-
16	$\|2731\rangle$	*not found*	-	26	$\|1365\rangle$	*not found*	-
25	$\|455\rangle$	9 (*odd*)	-	37	$\|2048\rangle$	2	19 i 3

Taula 10.15: Simulacions del QGT 04 per factoritzar el 57

63

y_0	l_j	Període (r)	Factors	y_0	l_j	Període (r)	Factors
40	$\|2731\rangle$	*not found*	-	46	$\|2732\rangle$	3 (*odd*)	-
41	$\|2048\rangle$	*not found*	-	52	$\|2048\rangle$	*not found*	-
23	$\|0\rangle$	-	-	61	$\|2048\rangle$	*not found*	-
23	$\|1365\rangle$	*not found*	-	29	$\|2730\rangle$	*not found*	-
50	$\|683\rangle$	6	9 i 7	43	$\|0\rangle$	-	-
25	$\|2731\rangle$	3 (*odd*)	-	22	$\|1366\rangle$	3 (*odd*)	-
8	$\|2048\rangle$	2	9 i 7	52	$\|3414\rangle$	6	7 i 9
32	$\|2048\rangle$	*not found*	-	43	$\|0\rangle$	-	-
59	$\|683\rangle$	6	63 i 1	4	$\|1365\rangle$	3 (*odd*)	-

Taula 10.16: Simulacions del QGT 04 per factoritzar el 63

A continuació, es detallen els resultats estadístics de les 210 simulacions realitzades.[6] Es pot veure que la suma dels percentatges en vermell correspon a la probabilitat que l'algorisme falli (68,57 %), mentre que el percentatge en blau correspon al percentatge d'èxit.

	Nombre	Percentatge
simulacions totals	210	100,00 %
factors no trivials trobats	137	65,24 %
factoritzacions correctes	66	31,43 %
factoritzacions trivials	22	10,48 %
odd periods trobats	30	14,29 %
$\|0\rangle$ mesurats	42	20,00 %
not found trobats	50	23,81 %

Taula 10.17: Resum de les simulacions realitzades amb el QGT 04

[6] S'ha de tenir en compte que entre els valors escollits en les simulacions hi ha el 25 i el 27 que corresponen a nombres que no es poden factoritzar mitjançant aquest algorisme.

Finalment, es presenta la mateixa taula però obviant les simulacions per factoritzar el 25 i el 27, els quals no es poden factoritzar amb aquest algorisme per ser potències de nombres primers.[7]

	Nombre	Percentatge
simulacions totals	188	100,00 %
factors no trivials trobats	137	72,87 %
factoritzacions correctes	66	35,11 %
factoritzacions trivials	14	7,45 %
odd periods trobats	23	12,23 %
$\|0\rangle$ mesurats	41	21,81 %
not found trobats	44	23,40 %

Taula 10.18: Resum de les simulacions realitzades amb el QGT 04 sense comptabilitzar el 25 i el 27

[7] També s'hauria d'obviar el 18, ja que és un enter de la forma $2 \cdot p^k$ i, per tant, té arrel primitiva.

CAPÍTOL

11

SÍNTESI I CONCLUSIONS

11.1 Conclusions

En començar el projecte estava molt motivat pel potencial latent de la computació i dels algorismes quàntics. Llegir nous articles, assistir a conferències i buscar més informació sobre aquests aspectes m'ha permès fer-me una idea més realista de la situació actual: la computació quàntica necessita un salt tecnològic important per poder evolucionar més enllà del marc teòric actual. En efecte, l'absència de l'ordinador quàntic per executar els algorismes quàntics és una limitació que he tingut present al llarg del projecte i que espero que desaparegui en un futur més o menys proper. A més, com he mencionat en la introducció, tinc molt clar que la tecnologia actual està limitada per la famosa llei de Moore i, com a conseqüència, els transistors cada vegada són més petits i els efectes quàntics cada vegada més importants. Com ha passat moltes altres vegades en la història de la ciència, els avenços teòrics previs al desenvolupament de la tecnologia permetran un impuls important quan arribi el canvi. Per tant, trobo molt encertats els estudis teòrics previs a l'ordinador quàntic i espero poder contribuir-hi juntament amb molts altres investigadors, científics i enginyers.

Pel que fa a la computació acumulativa que he desenvolupat, estic content dels resultats als quals he arribat fins ara. Com que el *swap* és la base de la permutació d'elements, els problemes de grups i de grafs es poden plantejar com a casos particulars de funcions acumulatives connectades en cascada i, a més, el paral·lelisme quàntic pot donar la solució eficient que encara no existeix dels problemes d'aquestes disciplines. Encara he d'estudiar molts aspectes sobre la coherència i la projecció dels estats (i de la implementació física de les matrius unitàries), però estic molt motivat a continuar treballant en aquesta idea i, si no resulta viable, consideraré que he trobat una manera de «no computar» que pot ser útil per a la pròxima idea.

Finalment, el projecte *Quantum Graph Tool* m'ha permès (i em permetrà) aprendre més sobre la computació quàntica, la teoria de grafs i la programació per a dispositius mòbils. Com a enginyer de telecomunicacions, programar aquestes aplicacions ha estat una bona tasca de reciclatge, ja que m'he hagut d'enfrontar sol a un nou llenguatge i entorn de programació i, a més, m'he hagut d'adaptar a les noves tecnologies.

Actualment, sóc desenvolupador autònom de iOS i he penjat una vintena d'aplicacions que em proporcionen uns ingressos modestos però constants i espero que el nombre d'aplicacions i la qualitat d'aquestes sigui cada vegada més elevat. A més, les aplicacions QGT són d'una temàtica tan específica i es dirigeixen a un públic tan reduït que la competència en aquest segment és pràcticament nul·la. Tot i això, com comento en el projecte, el meu interès principal és donar unes eines que permetin a l'usuari estudiar les noves maneres de computar i que motivin els joves científics i enginyers per avançar en aquesta branca tan fascinant.

11.2 Continuació

Per continuar aquest projecte hi ha dues vies principals que són, d'una banda, l'estudi de la computació acumulativa i, de l'altra, programar noves aplicacions QGT.

Pel que fa a les funcions acumulatives, com es comenta en l'apartat anterior, s'ha d'estudiar la coherència i la projecció dels estats. La coherència correspon a assegurar que les funcions acumulatives es comporten correctament per a superposicions d'estats, i. e. si tenim un estat que no sigui pur (p. e. la superoposició de dos estats) a l'entrada, interessa que a la sortida es mantinguin tots els estats i les sortides superposades corresponents sense interferència destructiva. La coherència és necessària perquè es puguin connectar funcions acumulatives en cascada i s'obtingui el comportament desitjat (acumular els estats de cada etapa). La projecció correspon a conèixer si el coeficient d'un estat concret és no nul per determinar si un element pertany al conjunt codificat per l'estat del registre. Una manera de resoldre el problema de la coherència pot ser utilitzar fases relatives entre una etapa i la següent; mentre que l'algorisme de Grover en si, pot donar una solució per al problema de la projecció.

Algunes propostes per a noves aplicacions QGT són: simular més algorismes quàntics teòrics, una evolució del QGT 01 que permeti simular els estats entrellaçats, una aplicació que permeti resoldre problemes clàssics de teoria de grafs (com el problema del carter, la coloració de mapes, l'aparellament òptim, l'algorisme de Kruskal, l'algorisme de Dijkstra...), una calculadora quàntica basada en blocs configurables o un compilador quàntic simple. És probable que els algorismes i les metodologies que sorgeixin durant la programació de les aplicacions generin idees per a noves aplicacions o, fins i tot, per a una nova manera de computar.

BIBLIOGRAFIA

[1] AMETLLER, Ll. (2011). Computació quàntica: apunts de classe.

[2] BENENTI, G. *et al.* (2008). *Principles of Quantum Computation and Information.* World Scientific Publishing Co. Pte. Ltd. ISBN 978-981-238-830-8, Vol.1, p. 49-137.

[3] DEUTSCH, D. (1985). *Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer.* Proceedings of the Royal Society of London A 400, p. 97-117

[4] FEYNMAN, R. P. (1982). *Simulating Physics with Computers.* International Journal of Theoretical Physics, Vol. 21, p. 467-488.

[5] FRENCH, A. P.; TAYLOR, E. F. (1982). *Introducción a la física cuántica.* Reverté, Tomo IV. ISBN 978-84-291-4167-2, p. 1-26, 105-115.

[6] HECHT, J. P. (2005). *Fundamentos de computación cuántica.* Editorial Académica Española. ISBN 978-3-8484-7529-2, p. 32-70.

[7] HILLEGASS, A. (2008). *Programación Mac OS X.* Anaya. ISBN 978-84-415-2805-5.

[8] LEWIS, R. (2010). *iPhone and iPad Apps for Absolute Beginners.* Apress. ISBN 978-1-4302-2701-4.

[9] NIELSEN, M; CHUANG, I. (2000). *Quantum Computation and Quantum Information.* EE.UU.: Cambridge University Press, 1 edition. ISBN 978-1-107-00217-3.

[10] PERRY, R. T. (2010). *The Temple of Quantum Computing.* Versió 1.2.

[11] SANCHEZ, C. (2002). *Física cuántica.* Anaya. ISBN 978-84-368-1656-0, p. 21-45, 73-83.

[12] TRAVESA, A. (1998). *Aritmètica.* Edicions Universitat de Barcelona. ISBN 978-84-8338-031-5, p. 89-165, 265-332.

[13] HORNBERGER, K. *et al.* «Collisional decoherence in matter wave interferometry». A: *Universität Duisburg-Essen* [en línia]. Duisburg (Alemanya): Universitat de Duisburg-Essen, s/d. <http://www.uni-due.de/∼hp0198/research/decoherence/collisionaldeco/index.html> [Consulta: abril 2013]

[14] «The Millikan experiment to verify Einstein's photo-electric relationship». A: *Learning on the loop* [en línia]. Nelson (Nova Zelanda): NMIT, s/d. <http://learningon.theloop.school.nz/moodle/mod/assign/view.php?id=39047> [Consulta: febrer 2013]

[15] «Transistor de un solo átomo y plenamente operativo». A: *NCYT: Noticias de la ciencia y la tecnologia* [en línia]. Barcelona: Amazings/NCYT, 2 març 2012 <http://noticiasdelaciencia.com/not/3642/transistor_de_un_solo_atomo_y_ plenamente_operativo/> [Consulta: febrer 2013]