UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Solving the 3-SAT problem using quantum algorithms

Autor: Arnau Casau Playà

Director: Jordi Delgado Pin (Departament de Ciències de la Computació)

Titulació: Grau en Enginyeria Informàtica

Especialitat: Computació

Memòria del projecte

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

26 de maig del 2023

Abstract

Quantum computing is an area of computer science that applies quantum mechanics principles to create a different computational paradigm than the one we are used to. This project explores how to solve 3-SAT using quantum algorithms to find a satisfying assignment of a given formula.

The first approach to the problem consists of using Grover's algorithm with a given formula to find its possible solutions. Grover's algorithm allows us to achieve a quadratic speedup over a brute-force search algorithm that searches in the space of all the possible assignments of the formula. In addition, the Quantum counting algorithm is used to determine how many solutions there are beforehand and thus maximize the probability of success in Grover's by repeating the Grover operator.

Classical computers can solve the problem more efficiently than just using brute-force search. Therefore, the previous quantum algorithm is compared with Schöning's, one of the best classical algorithms for 3-SAT. Finally, Grover's algorithm is generalized and used with Schöning's to obtain a quadratic speedup of the latter.

The different algorithms have been implemented using Qiskit, an open-source software development kit (SDK) developed by IBM to work with quantum computers using the Python Programming Language.

Resumen

La computación cuántica es una área de las ciencias de la computación que aplica los principios de la mecánica cuántica para crear un paradigma de computación diferente del que estamos acostumbrados. Este proyecto explora como solucionar 3-SAT utilizando algoritmos cuánticos para encontrar una asignación que satisfaga una fórmula dada.

El primer enfoque al problema consiste en usar el algoritmo de Grover con una fórmula para encontrar sus posibles soluciones. El algoritmo de Grover nos permite conseguir un speedup cuadrático respecto a un algoritmo de fuerza bruta que busca en el espacio de todas las posibles asignaciones de una fórmula. Además, se utiliza el algoritmo Quantum counting para determinar de antemano cuantas soluciones tiene nuestra fórmula y así maximizar la probabilidad de éxito en el algoritmo de Grover repitiendo su operador.

Los ordenadores clásicos pueden solucionar el problema de una forma más eficiente que a fuerza bruta y por eso se compara el algoritmo anterior con uno de los mejores algoritmos clásicos para 3-SAT como es el algoritmo de Schöning. Finalmente, se generaliza el algoritmo de Grover y se usa junto al algoritmo de Schöning para conseguir un speedup cuadrático de este último.

Los diferentes algoritmos han sido implementados con Qiskit, un software desarrollado por IBM que nos permite trabajar con ordenadores cuánticos, mediante el lenguaje de programación Python.

Resum

La computació quàntica és una àrea de les ciències de la computació que aplica els principis de la mecànica quàntica per crear un paradigma de computació diferent del que estem acostumats. Aquest projecte explora com solucionar 3-SAT utilitzant algorismes quàntics per trobar una assignació que ens satisfaci una fórmula donada.

El primer enfocament del problema consisteix a utilitzar l'algorisme de Grover amb una fórmula per trobar les seves possibles solucions. L'algorisme de Grover, ens permet aconseguir un speedup quadràtic respecte a un algorisme de força bruta que busca en l'espai de totes les possibles assignacions d'una fórmula. A més, s'utilitza l'algorisme Quantum counting per determinar el nombre de solucions que té la nostra fórmula i així poder maximitzar la probabilitat d'èxit en l'algorisme de Grover mitjançant la repetició del seu operador.

Els ordinadors clàssics poden solucionar el problema d'una forma més eficient que a força bruta, i per això es compara l'algorisme quàntic anterior amb un dels millors algoritmes clàssics per 3-SAT com és l'algorisme de Schöning. Finalment, l'algorisme de Grover és generalitzat i utilitzat juntament amb l'algorisme de Schöning par aconseguir un speedup quadràtic d'aquest últim.

Els diferents algoritmes han estat implementats amb Qiskit, un programari desenvolupat per IBM que ens permet treballar amb ordinadors quàntics fent ús del llenguatge de programació Python.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció i context

La computació quàntica és un camp multidisciplinari que uneix aspectes de les ciències de la computació, física i matemàtiques i que utilitza la mecànica quàntica per resoldre problemes complexos. Es tracta d'un model de computació diferent de la computació clàssica que ens permet resoldre certs problemes d'una forma més ràpida explotant conceptes com la superposició o l'entrellaçament. En aquest model s'utilitzen els qubits com a unitat mínima d'informació.

També ens obre la porta a molts problemes interessants que són impossibles de resoldre en un ordinador clàssic, no perquè siguin irresolubles, sinó per la quantitat de recursos que es necessita per resoldre casos realistes del problema. Algunes aplicacions actuals són en Aprenentatge automàtic, en problemes d'optimització o en la simulació de sistemes físics, entre d'altres.

Quan parlem d'algorismes quàntics podem fer una distinció entre dos grans grups. El primer grup són els algorismes basats en l'algorisme de Shor [1] i la transformada de Fourier que ens proporcionen un speedup exponencial respecte als millors algoritmes clàssics coneguts i el segon gran grup són els algorismes basats en l'algorisme de Grover [2] per realitzar cerques quàntiques que, en aquest cas, ens proporciona un speedup quadràtic sobre els millors algoritmes clàssics.

Aquest és un treball de fi de grau (TFG) de modalitat A, que se situa en el marc de la Facultat d'Informàtica de Barcelona (FIB) dins de la Universitat Politècnica de Catalunya (UPC) per al grau d'Enginyeria informàtica en l'especialitat de Computació.

L'objectiu del treball és resoldre el problema NP-complet 3-SAT mitjançant la implementació d'algorismes quàntics basats ens les idees darrere de l'algorisme de Grover i l'algorisme de Shor i fer una anàlisi de complexitat comparant els resultats amb algorismes clàssics, en concret, amb l'algorisme de Schöning [3]. A més es dissenyarà un circuit que combini els dos models de computació utilitzant l'algorisme de Grover i l'algorisme de Schöning tal com proposa A. Ambainis [4] per aconseguir un algorisme encara més eficient.

Aquest document servirà com a guia per qualsevol lector per entendre en profunditat el funcionament dels algorismes proposats.

1.1 Definició de conceptes

En aquest apartat podem veure una breu definició d'alguns dels conceptes claus esmentats a la introducció. Tot i que més endavant s'explicaran amb més detall, per tal d'entendre l'objectiu del treball i els algorismes proposats és convenient tenir clar el seu significat.

1.1.1 Qubit

Per entendre que és un qubit primer hem de saber que és un bit. El bit és la unitat mínima d'informació en la computació clàssica i pot tenir dos valors, 0 o 1.

Anàlogament, podem veure el qubit de la mateixa manera, com la unitat mínima d'informació en computació quàntica i pot prendre els valors $|0\rangle$ o $|1\rangle$ que, com podem imaginar, són els equivalents als valors 0 i 1 respectivament en la computació clàssica. La gran diferència i avantatge sobre els bits clàssics és que a més podem trobar un qubit en una combinació lineal dels dos estats, comunament anomenat, superposició:

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

La notació $'|\cdot\rangle'$ s'anomena notació de Dirac i és la notació estàndard pels diferents estats en mecànica quàntica. Els valors que trobem multiplicant cada qubit (α i β) s'anomenen amplituds i són nombres complexos.

Un qubit també es pot representar d'una forma més visual utilitzant un vector en una esfera anomenada esfera de Bloch. Més endavant s'explicarà amb més detall, però aquí podem veure un exemple:

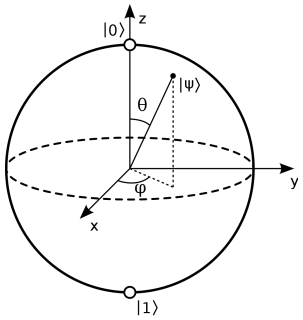


Figura 1.1: Esfera de Bloch
Font: Wikipedia [5]

1.1.2 Funció d'ona

En mecànica quàntica, una funció d'ona és una descripció matemàtica d'un estat quàntic d'una partícula en funció del moment, temps, posició i spin. El símbol que se sol utilitzar per a una funció d'ona és la lletra psi, ψ. La funció d'ona és una funció complexa i el seu producte amb la seva complexa conjugada (mòdul al quadrat) representa la densitat de probabilitat de trobar la partícula en un lloc i en un instant.

El col·lapse d'una funció d'ona és un procés físic relacionat amb el problema de la mesura de la mecànica quàntica que consisteix en la variació de l'estat inicial d'un sistema després haver obtingut una mesura.

1.1.3 Superposició

La superposició estableix que, igual que les ones en la física clàssica, es poden agregar dos o més estats quàntics i el resultat serà un altre estat quàntic vàlid. Ocorre quan un objecte es troba alhora en dos o més valors observables.

Un qubit en superposició ens està representant alhora totes les possibles configuracions fins que és mesurat. En mesurar un estat quàntic, la funció d'ona col·lapsa i l'estat es mesura com 0 o 1. En aquest estat final, el qubit actua com un bit clàssic.

La superposició dona als ordinadors quàntics un paral·lelisme inherent, permeten processar moltes operacions simultàniament.

1.1.4 Entrellaçament

L'entrellaçament és un efecte quàntic que correlaciona el comportament de dues entitats separades. Quan dos qubits estan entrellaçats, els canvis en un qubit afecten directament a l'altre i el coneixement sobre un dels qubits es converteix en coneixement immediat sobre l'altre. Els processadors quàntics poden extreure conclusions sobre una partícula mesurant una altra, com per exemple, determinar el spin. Els algoritmes quàntics exploten aquestes relacions per trobar solucions més ràpides a problemes complexes.

1.2 Problema a resoldre, 3-SAT

En aquest treball ens centrarem en com resoldre el problema de satisfacibilitat [6] [7] que va ser un dels primers problemes a ser identificat com a NP-complet per Stephen Cook en l'any 1971 [8]. En concret ens centrarem en la seva versió amb 3 literals per clàusula, la qual continua sent un problema NP-complet.

En aquest problema volem saber si, donada una expressió booleana amb variables i sense quantificadors, hi ha alguna assignació per totes les seves variables que fan que l'expressió sigui certa. Aquesta expressió booleana es coneix com a fórmula i està formada per diferents clàusules que en el nostre cas, en tractar-se de 3-SAT, contindran únicament tres variables o literals.

Una instància d'aquest problema podria ser la següent fórmula ϕ en forma normal conjuntiva (CNF):

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Les fórmules són donades en la seva forma normal conjuntiva (CNF) la qual consisteix en una conjunció de clàusules (entre parèntesis), a on cada clàusula és una disjunció de literals i cada literal és una variable booleana (com x) o la seva negació (com $\neg x$). Una assignació que satisfà una fórmula és una assignació de valors booleans ($true$ o $false$) a cada variable tal que cada clàusula conté almenys un literal amb valor $true$.

Més formalment podem definir el problema com, donada una fórmula booleana $\phi = \bigwedge_{i=1}^{m}(C_i)$ en forma normal conjuntiva (CNF), sobre un conjunt de variables booleanes $X = x_1, ..., x_n$, decidir si existeix una assignació $A : X \to \{0, 1\}$ tal que $A \models \phi$.

Si ens fixem en l'exemple anterior podem veure com una assignació que satisfà la fórmula ϕ podria ser $\langle x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1\rangle$, ja que:

$$\begin{aligned}
\phi &= (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4) \\
&= (0 \vee \neg 0 \vee 1) \wedge (0 \vee 1 \vee \neg 1) \\
&= (0 \vee 1 \vee 1) \wedge (0 \vee 1 \vee 0) \\
&= 1 \wedge 1 \\
&= 1
\end{aligned} \tag{1.1}$$

1.3 Actors implicats

Els actors implicats que podem trobar són:

1. Empreses. El problema que es resol en aquest treball té un munt d'aplicacions reals i qualsevol empresa que hagi de resoldre el problema es pot plantejar utilitzar un ordinador quàntic quan tinguin suficients qubits i estiguin preparats per resoldre problemes de grans dimensions.

2. Estudiants. Per tal de dissenyar els algorismes s'han d'entendre dos dels algorismes més importants de la computació quàntica i aquest treball pot brindar un enfocament pràctic que ajudi a qualsevol persona interessada a aprendre sobre computació quàntica i pot servir com a guia per aplicar-los a altres projectes.

3. Investigadors. La forma de resoldre aquest tipus de problemes és coneguda i pot ser exportada, adaptada i afegida en altres investigacions sobre altres problemes coneguts.

Capítol 2

Justificació

Aquest treball, com s'explica en la introducció, es podria dividir en dues parts.

La primera part consisteix en l'aplicació de l'algorisme de Grover al problema 3-SAT. Com veurem més endavant, per tal d'aplicar aquest algorisme en casos generals del problema farem ús d'un algorisme anomenat Quantum counting [9] com a pas previ a l'anterior esmentat. El que es proposa és conegut [10] [11] i podem trobar diverses implementacions [12], però la principal dificultat d'aquesta primera part, a causa del poc contingut de computació quàntica al llarg del grau, és entendre en profunditat l'algorisme de Grover i el Quantum counting per tal de poder aplicar-los al problema que volem.

En la segona part del treball, utilitzant la idea de A. Ambainis [4], combinarem l'algorisme de Grover i l'algorisme de Schöning per tal de millorar l'algorisme obtingut a la primera part del treball. Aquesta tècnica, tot i que és coneguda i sovint mencionada en la literatura, no es detalla la seva implementació, i per això el circuit que es proposarà és el resultat d'hores d'investigació i idees pròpies.

Tot i ser un treball de caràcter teòric també s'implementaran tots els algorismes esmentats utilitzant Qiskit, un programari desenvolupat per IBM que ens permet treballar amb ordinadors quàntics i simuladors fent ús del llenguatge de programació Python. En el cas de la segona part del projecte, a causa de les grans dimensions de l'algorisme i a l'ús de simuladors en un ordinador clàssic, no podem implementar directament el mateix algorisme teòric i, per tant, es farà alguna modificació per tal d'adaptar-lo.

Les implementacions dels algorismes es faran públiques a GitHub en diversos Jupyter Notebooks per tal de poder descriure cada pas dels algoritmes i que el resultat final serveixi com a guia per qualsevol lector interessant en aprendre més sobre computació quàntica i els seus usos.

Capítol 3

Abast

3.1 Objectius i sub-objectius

En aquest apartat podem veure més en profunditat tots els objectius del projecte organitzats en cinc grups i els seus sub-objectius dins d'aquests. En apartats anteriors s'ha comentat que l'objectiu principal és resoldre 3-SAT amb algoritmes quàntics, però aquí podem veure amb més detall tot el que volem aconseguir. Els objectius són:

1. Analitzar i entendre en profunditat l'algoritme de Grover.
2. Analitzar i entendre en profunditat l'algoritme de Quantum counting que, en essència, és l'algorisme de quantum phase estimation utilitzat en l'algorisme de Shor però amb l'operador de Grover.
3. Dissenyar un algoritme quàntic per resoldre 3-SAT basat en les idees anteriors capaç d'aconseguir un speedup quadràtic sobre un algorisme de brute-force search.
   (a) Utilitzar l'algorisme de Quantum counting per trobar el nombre de solucions d'una instància del problema.
   (b) Utilitzar l'algorisme de Grover per a poder trobar solucions concretes al problema.
   (c) Fer una anàlisi de complexitat que tingui en compte els dos algorismes anteriors per poder observar el speedup que aconseguim.
   (d) Comparar la complexitat resultant de l'algorisme anterior amb algoritmes clàssics com l'algorisme de Schöning per comprovar que hi ha altres mètodes millors.
4. Dissenyar un algorisme quàntic que aconsegueixi un speedup quadràtic sobre l'algorisme clàssic de Schöning.
   (a) Analitzar i entendre l'algorisme clàssic de Schöning.
   (b) Generalitzar l'algorisme de Grover per tal de poder combinar-lo amb l'algorisme de Schöning i aconseguir un speedup quadràtic d'aquest.
   (c) Fer una anàlisi de complexitat per comparar-lo amb els algorismes anteriors i demostrar que hi ha una millora.
5. Portar els algorismes teòrics a la pràctica explorant diferents simuladors quàntics utilitzant Qiskit/Python.
   (a) Analitzar els diferents simuladors disponibles amb Qiskit per veure quin s'adapta millor a cada algorisme.

(b) Desenvolupar l'algorisme de Grover aplicat a 3-SAT i que es pugui utilitzar per a qualsevol instància del problema.

(c) Desenvolupar l'algorisme que combina Grover i Schöning i que es pugui fer servir per a qualsevol instància del problema.

3.2 Requeriments

Per tal d'aconseguir els objectius definits en l'apartat anterior necessitem una sèrie de requisits, com poder ser:

- Coneixement bàsic de computació quàntica. Entendre els conceptes i algorismes bàsics per poder estudiar en profunditat els algorismes de Grover i Shor.
- Un entorn. Per tal de poder provar els algorismes que es desenvolupen es necessita un entorn a on o bé simular els algorismes o executar-los en un ordinador quàntic real suposant que tenim suficients recursos.
- Recursos o temps. A l'hora de simular els circuits quàntics, la quantitat de memòria RAM que es necessita creix exponencialment amb el nombre de qubits i, per tant, es necessita un ordinador amb recursos o, per contra, es poden utilitzar altres mètodes de simulació a on no cal disposar de tanta memòria, però a canvi necessitarem esperar més temps fins a rebre els resultats dels algorismes.

3.3 Obstacles i riscos

Tots els projectes poden tenir obstacles o riscos que poden fer que no tot vagi com un havia planejat a l'inici. En aquest treball un conjunt d'obstacles i riscos poden ser:

1. El temps. Aquest treball té un fort component de recerca per tal de desenvolupar els algorismes i donat el temps que es disposa existeix la possibilitat de no poder assolir tots els objectius marcats.
2. Informació disponible. Tot i que els algorismes per resoldre el problema són coneguts, en l'algorisme que combina els dos models de computació no es disposa d'informació sobre el disseny del mateix i això podria ser un gran obstacle, ja sigui en el temps que s'ha d'emprar o en el resultat final d'aquest.
3. Simulacions. Els circuits quàntics seran simulats en ordinadors clàssics, i per això les instàncies de 3-SAT amb les que es podrà provar no seran molt gran. Els simuladors poden suposar un problema, ja que si un algorisme necessita més qubits dels quals disposem en el simulador, aquest només podrà ser presentat de forma teòrica.
4. Inexperiència. La computació quàntica és un camp el qual, pràcticament, no es toca al llarg de la carrera i que és molt diferent de la computació clàssica. Aquest punt pot ser un obstacle a l'hora de dissenyar i desenvolupar els algoritmes.
5. Recursos. El que era un requisit també es pot convertir en un obstacle, ja que com no sabem abans de dissenyar els algorismes, la quantitat de qubits que necessitarem, tampoc podem conèixer la quantitat de memòria que es necessita i podria ser un problema per certes simulacions.

Capítol 4

Metodologia

En aquest apartat trobem la metodologia que s'ha escollit i la forma d'organitzar les diferents tasques. També podem trobar les eines que s'utilitzaran en aquest procés i la forma de validar la feina feta.

4.1 Metodologia àgil

Per poder realitzar aquest treball i com hem vist als objectius, s'ha de realitzar una feina d'investigació per aprendre els algorismes que serviran com a base per desenvolupar la resta. A causa d'això, la metodologia a escollir ha de ser flexible, ja que sempre hauríem de poder descartar feina feta i tornar a l'etapa d'investigació en cas que els resultats no siguin els esperats, es vulgui millorar o afegir funcionalitats.

Per això, s'ha escollit una metodologia àgil com és Scrum [13], amb la que poder fraccionar els diferents objectius en petites tasques. En el nostre cas, a cada objectiu s'ha d'arribar d'una forma seqüencial passant pels anteriors, ja que aquests tenen certes dependències. Les úniques tasques que són més modulars i es poden realitzar en diferents etapes són les d'implementació en codi dels algorismes i aquesta metodologia ens brindarà cicles de desenvolupament curts per tal de poder provar que s'arriba als resultats esperats. A l'hora de definir les tasques ens basarem en els diferents objectius definits en l'apartat Objectius i sub-objectius que utilitzarem com a blocs i que dividirem en tasques petites amb les quals anirem treballant.

La validació del projecte es durà a terme mitjançant reunions amb el director al final de cada un dels objectius amb l'excepció de certes tasques que poden tenir una complexitat major com les de dissenyar els algorismes d'una forma teòrica. A més en les tasques de desenvolupament de codi es farà ús dels simuladors disponibles per veure que realment els resultats són correctes, ja que ens permeten portar a terme moltes execucions i extreure estadístiques.

4.2 Eines

Les eines que s'utilitzaran per facilitar el seguiment de la metodologia anteriorment esmentada són:

- Trello. Es tracta d'una eina de gestió de projectes en línia amb la qual mitjançant un taulell i unes targetes que representen tasques a realitzar podem portar un seguiment de tot el que ens queda pendent en un determinat moment. També ens permet marcar les tasques com completades o fins i tot afegir anotacions sobre problemes que ens anem trobant per tal d'evitar que caiguin en l'oblit.

- GitHub. Aquesta eina ens serà útil com a control de versions en les diferents implementacions de codi, ja que podrem tenir el nostre treball guardat en el núvol de forma segura i alhora tenir accés a versions antigues en cas de voler restaurar la feina feta. A més ens servirà, un cop acabar el projecte, per fer públic el codi i que serveixi per a altres persones interessades.
- Qiskit. Utilitzarem els diferents simuladors que ens proporciona IBM per tal de poder executar i fer proves amb el codi que anem desenvolupant. Els algorismes quàntics són probabilístics i Qiskit ens permet executar-los diverses vegades i acabar extraient les diferents freqüències de cada output que ens servirà per verificar el treball fet.

Capítol 5

Planificació temporal

En aquest capítol podem trobar la planificació temporal del treball, la qual ha estat realitzada tenint en compte que l'inici d'aquest va ser el 20/02/2023 amb la sessió de presentació del GEP i durarà fins a aproximadament el 26-30/06/2023 a on s'escollirà un dia dins del rang anterior per la defensa del treball.

El nombre de crèdits del TFG és de 18 i cada crèdit s'estima en una càrrega de treball de 30 hores [14], per tant, s'espera que es facin un total de 540 hores de feina que repartirem entre les diferents tasques.

A continuació podem trobar la descripció de totes les tasques a realitzar identificades mitjançant codis amb la seva durada en hores, les dependències que podem trobar i els recursos (humans i materials) que seran necessàries. També trobem un diagrama de Gantt i finalment, es proposen solucions a obstacles eventuals que ens podem trobar al llarg del TFG.

5.1 Descripció de les tasques

En aquesta secció podem trobar totes les tasques que es preveu realitzar en el projecte, agrupades en quatre categories. Cada tasca té una descripció del seu propòsit, una estimació de les hores que es dedicaran i l'enumeració de les dependències amb tasques anteriors per poder ser començades.

5.1.1 T1 - Gestió del projecte

Aquí podem trobar agrupades totes les tasques relacionades amb la gestió del projecte, a on les tasques que engloben el contingut del GEP i que ens serveixen per assentar les bases del projecte tenen un gran pes, sobretot a l'inici del treball. El GEP s'ha descompost en tres tasques com podrem veure a continuació i s'han dividit els 3 ECTS en parts iguals entre elles.

- T1.1 - Context i abast: En aquesta tasca es defineix l'abast del projecte en el context del seu estudi. S'indica l'objectiu general del TFG, la seva contextualització i com es desenvoluparà. Aquesta primera tasca no té cap dependència i se li ha assignat 30 hores que correspon a 1 dels 3 ECTS del GEP.
- T1.2 - Planificació temporal: Es definirà la planificació temporal per l'execució del TFG proporcionat una descripció de les diferents fases del projecte, les tasques que es realitzaran i com aquestes tenen dependències entre elles. Aquesta tasca dependrà del context i abast definits en la tasca T1.1 i se li ha assignat 30 hores que correspon a 1 ECTS del GEP.

- T1.3 - Pressupost i sostenibilitat: Després de definir la planificació temporal, s'estimarà el pressupost necessari per dur a terme el projecte i s'analitzarà la seva sostenibilitat. Aquesta tasca té una estimació de 30 hores de feina, ja que correspon a l'últim ECTS del GEP.
- T1.4 - Documentació: Aquesta tasca és una de les més importants que es realitzarà durant el transcurs del treball. Inclou les entregues pròpies del GEP i la documentació de la part tècnica del projecte, per això se li ha assignat 65 hores de feina.
- T1.5 - Reunions: Consisteix en les diferents trobades amb el director per planificar l'abast del projecte, realitzar el seguiment d'aquest i resoldre problemes o dubtes que poden sorgir. Se li ha assignat 20 hores que es repartiran al llarg del quadrimestre en el qual es fa el TFG.
- T1.6 - Defensa del treball: Aquesta tasca inclou la preparació de la defensa del TFG que es realitzarà al juny davant del tribunal. Es començarà a preparar un cop enllestida la documentació del projecte. Se li dedicarà 15 hores, ja que també es prepararan les demostracions dels diferents algorismes.

5.1.2 T2 - Recerca

Aquest grup de tasques ens representa la feina de recerca que s'ha de fer prèvia a poder dissenyar o desenvolupar els algorismes.

- T2.1 - Algorisme de Grover: S'estudiarà el funcionament d'aquest algorisme que és el més important i serveix com a base per posteriorment dissenyar les nostres solucions. Se li dedicarà un total de 50 hores.
- T2.2 - Quantum counting: Un cop realitzada la recerca de l'algorisme de Grover s'investigarà com utilitzar el Quantum counting, ja que el necessitarem conjuntament amb l'anterior per construir la nostra primera solució. Se li dedicarà també 50 per la gran envergadura.
- T2.3 - Algorisme de Schöning: S'investigarà com funciona l'algorisme clàssic de Schöning i com combinar-lo amb l'algorisme de Grover per tal de crear el segon circuit que es proposarà. Igual que amb les altres dues tasques, se li dedicarà 50 hores.
- T2.4 - Qiskit: S'estudiarà les diferents opcions per simular els algorismes anteriors i poder desenvolupar les nostres solucions. Gràcies al coneixement previ de la sintaxi del llenguatge, aquesta tasca ens portarà menys que les anteriors i se li ha assignat un total de 25 hores.

5.1.3 T3 - Disseny i anàlisis

Aquí trobarem les tasques relacionades amb el disseny dels diferents algorismes quàntics de forma teòrica i la seva anàlisi de complexitat. Aquest disseny és previ a la seva implementació en codi i posterior a la recerca.

- T3.1 - Disseny teòric primer algorisme: En aquesta tasca es dissenyarà de forma teòrica un algorisme per resoldre 3-SAT mitjançant l'algorisme de Grover i amb ajuda del Quantum counting estudiats anteriorment. Se li dedicarà un total de 30 hores.
- T3.2 - Anàlisi de complexitat primer algorisme: Es farà una anàlisi de complexitat de l'algorisme resultant de la tasca T3.1 i se li dedicarà 20 hores, ja que hi ha moltes parts a analitzar.
- T3.3 - Disseny teòric segon algorisme: Es dissenyarà de forma teòrica un segon algorisme que combini l'algorisme de Schöning i l'algorisme de Grover. Com que no es disposa de tanta informació se li dedicarà un total de 45 hores.
- T3.4 - Anàlisi de complexitat segon algorisme i comparació amb el primer: Es farà una anàlisi de complexitat del segon algorisme per tal de poder observar si hi ha hagut una millora o no. Es dedicarà un total de 10 hores.

5.1.4 T4 - Implementació

Aquest és l'últim grup de tasques i que ens representa la implementació en codi dels algorismes dissenyats en les tasques anteriors.

- T4.1 - Implementació en codi del primer algorisme: En aquesta tasca s'implementarà amb Python i Qiskit el primer algorisme dissenyat a la tasca T3.1. Se li dedicarà un total de 40 hores.
- T4.2 - Implementació en codi del segon algorisme: En aquesta tasca s'implementarà amb Python i Qiskit el segon algorisme dissenyat a la tasca T3.3. Se li dedicarà un total de 30 hores.

A continuació podem trobar una taula resum de les tasques esmentades en els apartats anteriors així com el temps en hores que dedicarem a cada una i les dependències que podem trobar:

Codi	Tasca	Temps	Dependència
T1	**Gestió del projecte**	**190 h**	
T1.1	Context i abast	30 h	
T1.2	Planificació temporal	30 h	T1.1
T1.3	Pressupost i sostenibilitat	30 h	T1.2
T1.4	Documentació	65 h	
T1.5	Reunions	20 h	
T1.6	Defensa del treball	15 h	T1.4
T2	**Recerca**	**175 h**	
T2.1	Algorisme de Grover	50 h	
T2.2	Quantum counting	50 h	T2.1
T2.3	Algorisme de Schöning	50 h	T2.1,T2.2
T2.4	Qiskit	25 h	
T3	**Disseny i anàlisis**	**105 h**	
T3.1	Disseny teòric primer algorisme	30 h	T2.1,T2.2
T3.2	Anàlisi de complexitat primer algorisme	20 h	T3.1
T3.3	Disseny teòric segon algorisme	45 h	T2.3
T3.4	Anàlisi de complexitat segon algorisme i comparació amb el primer	10 h	T3.3
T4	**Implementació**	**70 h**	
T4.1	Implementació en codi del primer algorisme	40 h	T3.2
T4.2	Implementació en codi del segon algorisme	30 h	T3.4
Total		**540 h**	

Taula 5.1: Taula resum de totes les tasques agrupades per blocs.

5.2 Recursos necessaris

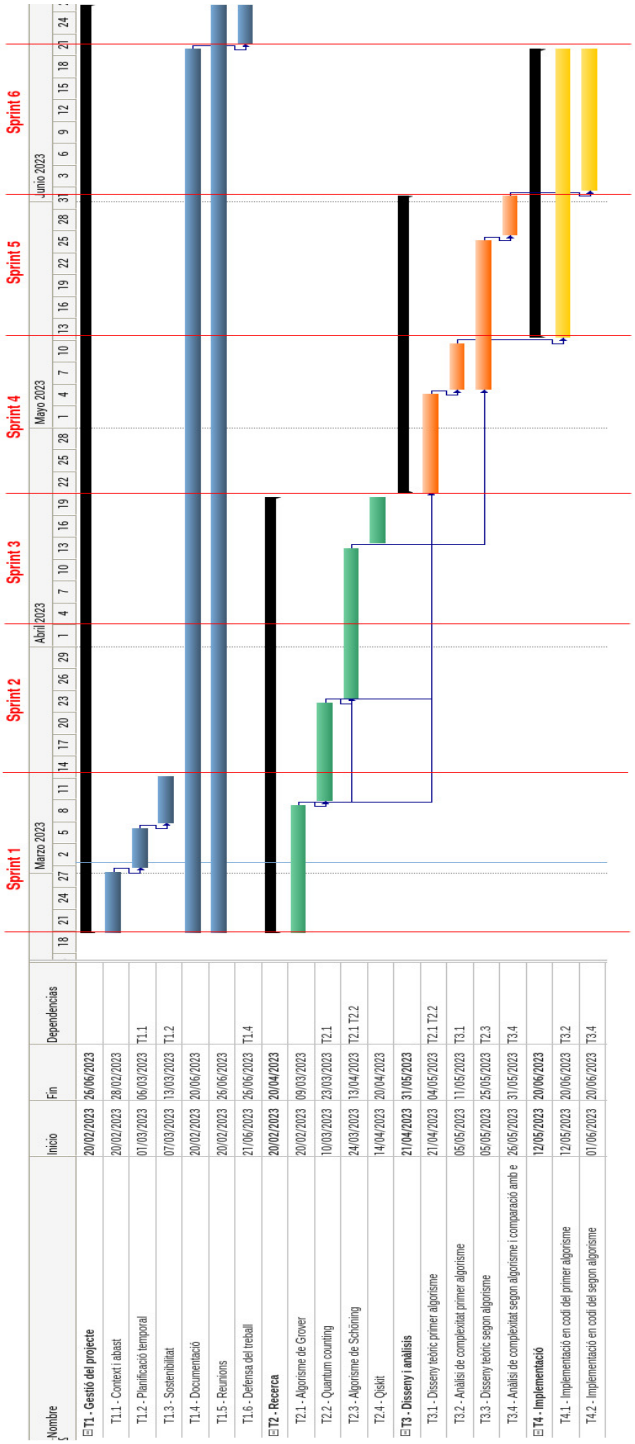
Per la realització d'aquest treball es requereixen uns certs recursos que podem diferenciar entre recursos humans i materials. En els recursos humans, tot i que aquest treball es fa de manera individual, és necessari assumir els rols de cap de projecte o director que s'encarregui de la planificació del projecte, d'investigador que sigui capaç de dissenyar els diferents algorismes d'una manera teòrica i de programador capaç de transformar els dissenys teòrics en codi.

Com a recursos materials, necessitem un ordinador per tal de poder simular els diferents algorismes, un lloc de treball amb connexió a internet per poder dur a terme les diferents tasques d'investigació i un entorn, que en aquest cas ens el proporciona Qiskit, amb el qual poder treballar amb ordinadors

quàntics sigui en un entorn real o mitjançant simuladors.

Finalment, necessitarem un cert programari per tal de poder implementar els algorismes com pot ser Visual Studio Code, GitHub per tal de tenir un control de versions i altres programes com Gantter o Trello per la part de gestió del projecte.

5.3 Diagrama de Gantt



Nombre	Inicio	Fin	Dependencias
ET1 - Gestió del projecte	20/02/2023	26/06/2023	
T1.1 - Context i abast	20/02/2023	28/02/2023	
T1.2 - Planificació temporal	01/03/2023	06/03/2023	T1.1
T1.3 - Sostenibilitat	07/03/2023	13/03/2023	T1.2
T1.4 - Documentació	20/02/2023	20/06/2023	
T1.5 - Reunions	20/02/2023	26/06/2023	
T1.6 - Defensa del treball	21/06/2023	26/06/2023	T1.4
ET2 - Recerca	20/02/2023	20/04/2023	
T2.1 - Algorisme de Grover	20/02/2023	09/03/2023	
T2.2 - Quantum counting	10/03/2023	23/03/2023	T2.1
T2.3 - Algorisme de Schöning	24/03/2023	13/04/2023	T2.1 T2.2
T2.4 - Qskit	14/04/2023	20/04/2023	
ET3 - Disseny i anàlisis	21/04/2023	31/05/2023	
T3.1 - Disseny teòric primer algorisme	21/04/2023	04/05/2023	T2.1 T2.2
T3.2 - Anàlisi de complexitat primer algorisme	05/05/2023	11/05/2023	T3.1
T3.3 - Disseny teòric segon algorisme	05/05/2023	25/05/2023	T2.3
T3.4 - Anàlisi de complexitat segon algorisme i comparació amb e	26/05/2023	31/05/2023	T3.4
ET4 - Implementació	12/05/2023	20/06/2023	
T4.1 - Implementació en codi del primer algorisme	12/05/2023	20/06/2023	T3.2
T4.2 - Implementació en codi del segon algorisme	01/06/2023	20/06/2023	T3.4

Figura 5.1: Diagrama de Gantt
Font: Elaboració pròpia amb Gantter [15]

5.4 Gestió del risc

En aquest treball, a causa que s'ha d'investigar sobre molts conceptes que no es veuen al llarg de la carrera, un dels principals problemes que podem tenir és el fet d'haver d'investigar sobre temes que a priori, per desconeixement inicial, no es comptava amb ells. Per això, s'ha decidit sobreestimar les hores de dedicació de diverses tasques com són les de recerca. Com podem veure a la taula 1.1, se li ha assignat 50 hores a cada algorisme per tal d'assegurar-nos d'adquirir el coneixement necessari al principi del treball.

També s'ha sobreestimat les hores en el disseny dels algorismes, sobretot del segon, a causa de la poca informació disponible per construir el circuit i la inexperiència a l'hora de dissenyar-los. Com podem veure a la figura 1.1, es comença a fer recerca des del primer dia alhora que es van duent a terme tasques de gestió del projecte, ja que aquestes no ens modificaran en el coneixement que necessitarem més endavant.

S'ha dividit el treball en sis sprints en els quals sempre tindrem reunions amb el director per assegurar que s'estan complint els objectius i anirem redactant la documentació per tal de no acumular feina més tard.

Un altre risc que vam veure és el fet que els algorismes dissenyats necessitin més recursos dels que els simuladors disponibles tenen. En cas que això passes es miraria d'implementar els algorismes amb instàncies del problema molt senzilles però fent més èmfasis en el procés que en el resultat final.

Finalment, en el cas que l'ordinador que fem servir no sigui capaç de simular els algorismes quàntics, no per culpa del simulador, sinó pel hardware que disposem, s'explorarien alternatives com Google Colab [16] a on es poden sol·licitar màquines amb més recursos.

Capítol 6

Pressupost

En aquest capítol podem trobar l'estudi econòmic del projecte a on s'estimaran els costos de personal, genèrics, de contingència i d'imprevistos. Al final veurem mecanismes per controlar les desviacions que puguin aparèixer en relació amb el pressupost.

6.1 Costos de personal

Per la realització del treball necessitarem un cap de projecte que s'encarregui de les tasques de gestió i presentació del treball, un investigador capaç de dissenyar els algorismes que necessitarem pel projecte i un programador capaç de convertir els dissenys teòrics en funcionalitats pràctiques.

En la taula 6.1 podem veure els diferents costos que tindrem per cada treballador. Per calcular els salaris s'ha utilitzat el sou mitjà per hores segons la plataforma talent [17] [18] [19] i s'ha afegit el cost de la seguretat social, que suposa el 30% del sou.

Perfil	Sou brut/hora	S.S.	Cost total
Cap de projecte	18,46 €/h	5.54 €/h	24 €/h
Investigador	12,91 €/h	3.87 €/h	16,78 €/h
Programador	14,62 €/h	4.39 €/h	19,01 €/h

Taula 6.1: Costos per hora del personal

Un cop tenim els costos per cada treballador podem calcular el CPA (cost per activitat) a on utilitzarem les tasques definides en el diagrama de Gantt. Com podem veure a la taula 6.2, cada tasca té associada un únic treballador i el cost de la tasca consisteix en el nombre d'hores que se li dedicarà multiplicat pel salari del treballador en concret tenint en compte les despeses de la seguretat social com s'ha vist a la taula 6.1.

Codi Tasca	Perfil	Temps	Cost
T1.1	Cap de projecte	30	720 €
T1.2	Cap de projecte	30	720 €
T1.3	Cap de projecte	30	720 €
T1.4	Cap de projecte	65	1560 €
T1.5	Cap de projecte	20	480 €
T1.6	Cap de projecte	15	360 €
T2.1	Investigador	50	839 €
T2.2	Investigador	50	839 €
T2.3	Investigador	50	839 €
T2.4	Investigador	25	419,50 €
T3.1	Investigador	30	503,40 €
T3.2	Investigador	20	335,60 €
T3.3	Investigador	45	755,10 €
T3.4	Investigador	10	167,80 €
T4.1	Programador	40	760,40 €
T4.2	Programador	30	570,30 €
Total (CPA)			**10.589,10 €**

Taula 6.2: Costos salarials de cada treballador separats per tasques

6.2 Costos genèrics

En aquesta secció podem trobar els costos genèrics, els quals són independents a les tasques prèviament esmentades. També calcularem les diferents amortitzacions a on tindrem en compte que la durada del projecte és d'uns 5 mesos.

- Espai de treball. S'ha optat per llogar un espai compartit de *coworking* [20] per la flexibilitat que ens dona i pels preus que podem trobar. L'espai [21] escollit té un preu de 249 €/mes, el qual inclou tot el que necessitem com pot ser el mobiliari, la connexió a internet o el cost d'electricitat.

- Hardware. Per dur a terme les diferents tasques, els treballadors necessitaran un ordinador portàtil. S'ha estimat que el cap de projecte i l'investigador utilitzaran un portàtil de gamma baixa-mitja d'uns 500 €, mentre que el programador necessitarà un ordinador amb més recursos per poder simular els diferents algorismes que s'ha estimat sobre uns 1000 €. S'estima que la vida útil dels dos ordinadors de gamma baixa-mitja és de 4 anys mentre que el del programador és de 6 anys, per tant, l'amortització resultant és de $\frac{5}{48} \cdot 1000 + \frac{5}{72} \cdot 1000 = 173,60$ €.

- Software. En aquest projecte utilitzarem diverses eines de software com poden ser Visual Studio Code [22] per les implementacions en codi, GitHub [23] per tenir un control de versions i fer públic el projecte a l'acabar, Overleaf [24] per la documentació, Trello [25] per tal de poder organitzar les tasques i Gantter [26] per realitzar el diagrama de Gantt. Tot el programari anterior és gratuït menys Gantter el qual té un cost de 5 €/mes.

A continuació podem trobar una taula a on veiem el cost amortitzat del hardware que farem servir en el projecte. L'espai no apareix com a amortització, ja que és un lloguer que finalitzarà al cap de cinc mesos igual que les llicències de software. Podem trobar la fórmula utilitzada per calcular el cost total en l'apartat de hardware de dalt.

Element	Cost total	Vida útil	Amortització
Portatil gamma baixa 1	500 €	4 anys	52,08 €
Portatil gamma baixa 2	500 €	4 anys	52,08 €
Portatil gamma alta	1000 €	6 anys	69,44 €
Total hardware			**173,60 €**

Taula 6.3: Taula resum dels costos amortitzats en el hardware.

També podem trobar una taula que resumeix els costos en l'espai de treball i el software que utilitzarem. El cost total de cada element està calculat multiplicant el seu cost mensual pels cinc mesos que dura el projecte.

Element	Cost mensual	Mesos	Cost total
Espai de treball	249 €/mes €	5	1.245 €
Visual Studio Code	0 €	5	0 €
GitHub	0 €	5	0 €
Overleaf	0 €	5	0 €
Trello	0 €	5	0 €
Gantter	5 €	5	25 €
Total espai i software			**1.270 €**

Taula 6.4: Taula resum dels costos d'espai i software.

Finalment, amb els càlculs realitzats a les taules anteriors podem concloure que els **costos genèrics totals (CG) són de 1.443,60 €**.

6.3 Contingències

Cal tenir en compte que poden sorgir diferents contratemps o complicacions al llarg del projecte que no podem anticipar i, per tant, necessitem tenir un marge de seguretat. La partida de contingència es calcula com un percentatge del valor total del pressupost i en el sector informàtic se sol fixar una ràtio d'entre el 10% i el 20%.

En el nostre cas escollirem un valor mitjà del 15% que podem calcular de la següent manera: $(TotalCPA + TotalCG) \cdot 0.15 = (10.589,10 \text{ €} + 1.443,60€) \cdot 0.15 =$ **1.804,91 €**.

6.4 Imprevistos

Finalment, cal tenir en compte els impediments o obstacles mencionats anteriorment. El cost de cada imprevist es calcula com la despesa que ens suposaria multiplicada per la seva probabilitat de succeir. Com s'ha comentat en seccions anteriors, pel fet que comprem els ordinadors abans de tenir tots els algoritmes dissenyats correm el risc que per algun algorisme necessitem més recursos i, per tant, es contractaria Google Colab [27] durant els 2 últims mesos. Un altre imprevist pot ser la necessitat de substituir algun ordinador dels treballadors per mal funcionament o augmentar les hores en el disseny dels algorismes i/o la implementació.

A continuació podem veure una taula a on es detallen els costos. Cal remarcar que s'ha decidit comptar amb 25 hores extres pel disseny i 25 hores extres per la implementació per pal·liar l'últim risc esmentat.

Imprevist	**Despesa**	**Probabilitat**	**Cost**
Google Colab	22,38 €	30%	6,71 €
Nou portatil gamma baixa 1	500 €	25%	125 €
Nou portatil gamma baixa 1	500 €	25%	125 €
Nou portatil gamma alta	1000 €	15%	150 €
Increment temps disseny (25 h)	419,50 €	30%	125,85 €
Increment temps implementació (25 h)	475,25 €	30%	142,58 €
Total Imprevistos			**675,14 €**

Taula 6.5: Resum dels costos a causa d'imprevistos

6.5 Cost total

A continuació podem veure una taula amb el cost total del projecte que s'aconsegueix sumant els costos finals calculats en els quatre apartats anteriors.

Concepte	**Cost**
Total CPA	10.589,10 €
Total CG	1.443,60 €
Total Costos (Total CPA + Total CG)	**12.032,70 €**
Contingència	1.804,91 €
Total CD+CI+Contingència	**13.837,61 €**
Total imprevistos	675,14 €
Total	**14.512,75 €**

Taula 6.6: Cost total del projecte

6.6 Control de gestió

En aquesta secció definirem un control de costos per tal de poder comparar i avaluar les desviacions entre el pressupost realitzat i els costos reals del projecte. S'ha decidit aprofitar les reunions de seguiment per actualitzar el pressupost al final de cada etapa. Fent això, podrem saber en quins punts del projecte es produeixen les desviacions, el motiu específic que ens ha portat a la desviació, sigui positivament o negativament i l'import concret d'aquesta, ja que podrem comprar-ho amb el que teníem previst en el pressupost.

Si el cost real d'una tasca/etapa del projecte és major al pressupostat, podrem valorar quina part ha estat cobert per les contingències i quina part per la partida d'imprevistos. El pressupost s'ha realitzat tenint en compte possibles fallades del hardware, la necessitat de software addicional o la necessitat d'invertir hores extres en les diferents etapes, per tant, podríem tenir desviacions positives en certs punts del projecte que s'aprofitaran per compensar tasques a on les desviacions siguin negatives.

Per calcular les diferents desviacions ho farem de la següent manera:

- Desviació cost de personal per tasca:

  $(cost\_estimat - cost\_real) \cdot hores\_reals$

- Desviació cost de les tasques:

  $(hores\_estimades - hores\_reals) \cdot cost\_real$

- Desviació total costos de personal:
  $cost\_estimat - cost\_real$
- Desviació total costos genèrics:
  $cost\_estimat - cost\_real$
- Desviació total costos per imprevistos i contingències:
  $cost\_estimat - cost\_real$
- Desviació total d'hores del projecte:
  $hores\_estimades - hores\_reals$
- Desviació total de costos del projecte:
  $cost\_estimat - cost\_real$

Capítol 7

Sostenibilitat

7.1 Autoavaluació

Després de contestar l'enquesta de sostenibilitat m'he adonat que el meu coneixement sobre el tema es força escàs. Fins al moment, en planificar un projecte mai havia tingut en compte la sostenibilitat i sempre m'havia centrat en aspectes més econòmics o de rendiment dels programes realitzats. A l'hora de pensar en sostenibilitat dins de la informàtica sempre ho havia relacionat amb dispositius tangibles i no tant en el d'envolupament de software, tot i que també necessiti el recolzament de dispositius físics per funcionar. Un altre aspecte important que he de millorar en tractar temes de sostenibilitat és que normalment tendeixo a pensar més en els problemes que poden sorgir curt termini que a llarg termini.

Gràcies a l'enquesta m'he adonat que la sostenibilitat és un punt molt important a tenir en compte i que va molt més enllà que només pensar en la contaminació dels productes.

7.2 Dimensió econòmica

El cost de la realització del projecte està descrit en el capítol anterior i podem trobar tant els recursos humans com materials. Pel que fa als recursos humans són mínims, ja que només comptem amb un equip de tres persones i, encara que alguna tasca s'ha sobreestimat, el temps assignat a cada activitat és quasi l'adequat tenint en compte els coneixements que s'han d'adquirir. Els altres costos també són raonables perquè no trobem cap eina o dispositiu innecessari i s'ha intentat buscar les opcions més assequibles.

Si ho comparem amb solucions existents, veiem que no aconseguim una millora econòmica, perquè en el primer algorisme proposat utilitzem una solució existent, que, per tant, no representa una millora econòmica, i en el segon, no ho podem determinar, a causa de la poca informació sobre l'estat de l'art en la implementació, però, poden existir algorismes més sofisticats que requereixen menys recursos per funcionar que els presentats en aquest treball i, en conseqüència, tampoc representaran una millora econòmica.

7.3 Dimensió ambiental

L'impacte ambiental del projecte no s'ha tingut en compte a l'hora de fer la planificació, però com podem veure al capítol anterior estem utilitzant els recursos indispensables per la seva realització. En el nostre cas també fem ús d'un espai compartit de coworking que ens permet reduir la despesa

energètica necessària. El projecte és quasi totalment teòric i, per tant, a part de l'esmentat, no suposarà un impacte ambiental negatiu. Comparat amb altres projectes semblants no es pot aconseguir cap millora sense tenir en compte els recursos hardware que van ser usats en altres projectes i que es desconeixen.

7.4 Dimensió social

La computació quàntica és un camp d'estudi que al llarg de la carrera es veu molt poc i personalment, crec que aquest projecte em pot ajudar molt a assentar les bases i aprendre molts conceptes nous. En tractar-se d'un projecte de gran envergadura també m'ajudarà molt a agafar pràctica en la planificació d'aquest tipus de treballs i a seguir una metodologia de feina determinada.

Com s'ha comentat anteriorment, la solució proposada en aquest treball no millora a les actuals, però sí que pot ser de gran ajuda a qualsevol estudiant que es vulgui iniciar en el camp i necessiti una guia completa per entendre els diferents algorismes que es treballaran i com portar-los a la pràctica. De totes maneres no crec que existeixi una necessitat social real per aquest treball més enllà de l'acadèmica.

Capítol 8

Computació quàntica

En aquest capítol podem trobar una introducció a diferents conceptes que ens seran d'utilitat a l'hora d'entendre els diferents algorismes proposats en el treball. Es revisitaran conceptes com els qubits i s'explicaran conceptes nous com les portes quàntiques o els circuits quàntics.

8.1 Qubits

Com s'ha explicat breument a la introducció del treball, un qubit és la unitat mínima d'informació en computació quàntica i a diferència dels bits en computació clàssica, aquests es pot trobar en superposició, és a dir, en una combinació lineal de dos estats base com són $|0\rangle$ i $|1\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{8.1}$$

Els valors α i β són nombres complexos i la notació que utilitzem per descriure els qubits s'anomena notació de Dirac a on representem els estats com a vectors de 2-dimensions amb entrades també complexes de la següent manera:

$$\textbf{ket:} \qquad |a\rangle = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{8.2}$$

$$\textbf{bra:} \qquad \langle b| = |b\rangle^\dagger = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}^\dagger = (b_1^* \; b_2^*) \tag{8.3}$$

$$\textbf{bra-ket:} \qquad \langle b|a\rangle = a_1 \cdot b_1^* + a_2 \cdot b_2^* = \langle a|b\rangle^* \in \mathbb{C} \tag{8.4}$$

$$\textbf{ket-bra:} \qquad |a\rangle\langle b| = \begin{pmatrix} a_1 b_1^* & a_1 b_2^* \\ a_2 b_1^* & a_2 b_2^* \end{pmatrix} \tag{8.5}$$

En l'expressió 8.2 podem trobar la notació pel que anomenarem ket i que ens serveix per representar un estat quàntic que com veiem és un vector columna amb dues components. Per altra banda, a l'expressió 8.3 trobem la notació pel que anomanem bra, que tot i ser similar al ket, podem veure com equival a un vector fila a on les seves components són complexes conjugades[1][2] respecte a l'estat quàntic original que representaríem amb un ket.

Les expressions 8.4 i 8.5 són dues operacions que podem fer amb dos estats quàntics i que són equivalents a l'inner product i a l'outer product en àlgebra lineal respectivament. Seguint la notació anterior definim els estats $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ i $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, els quals formen una base de $\mathbb{C}^2$ i, per tant, en ser

[1] Sigui A una matriu, $A^\dagger$ representa l'operador adjunt o Hermitian conjugate de la matriu A, $A^\dagger = (A^T)^*$.
[2] Sigui B una matriu o un nombre complex, B^* representa el complex conjugat de B.

ortogonals i realitzar l'inner product tenim $\langle 0|1\rangle = 1 \cdot 0 + 0 \cdot 1 = 0$. Cal remarcar també que els estats quàntics estan normalitzats i, en conseqüència, en realitzar l'inner product amb ells mateixos arribem a $\langle \psi|\psi\rangle = 1$.

Estats de múltiples qubits

Normalment, treballarem amb sistemes amb més d'un qubit i els representarem mitjançant un producte tensorial de tots els nostres estats. Per exemple si tenim dos qubits, un en l'estat $|0\rangle$ i l'altre en l'estat $|1\rangle$, el podem representar com:

$$|0\rangle \otimes |1\rangle \equiv |0\rangle|1\rangle \tag{8.6}$$

Com veiem podem simplificar la notació i ometre el símbol de producte tensorial. Un altre exemple més il·lustratiu pot ser el producte tensorial de dos qubits en superposició a on veiem que el resultat es pot representar amb un vector de quatre components:

$$|\psi\rangle \otimes |\gamma\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) \tag{8.7}$$

$$= ac|0\rangle|0\rangle + ad|0\rangle|1\rangle + bc|1\rangle|0\rangle + bd|1\rangle|1\rangle \tag{8.8}$$

$$= \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} \tag{8.9}$$

El procés anterior és fàcilment generalitzable i es pot veure com sempre tindrem 2^q components en el vector resultant del producte tensorial a on q és el nombre de qubits del sistema, tot i que en aquest document no ho veurem en detall.

Mesures

Finalment, per saber en quin estat es troba un qubit, hem de mesurar-lo. Quan mesurem un qubit, tot i que aquest es trobi en superposició, sempre acabarà col·lapsant en un dels estats de la base. Aquesta mesura és destructiva, és a dir, un cop mesurat el qubit, aquest tindrà un valor de la base i no estarà més en superposició. Per exemple, el qubit de l'expressió 8.1, quan el mesurem, ens retornarà el valor $|0\rangle$ o $|1\rangle$.

Les mesures es fan respecte a una base i fins al moment només hem vist la base $\{|0\rangle, |1\rangle\}$ que es correspon en mesurar respecte a l'eix Z de l'esfera de Bloch que podem veure a la figura 1.1, però podem realitzar una mesura utilitzant altres bases com per exemple aquestes dues que equivalen a mesurar l'estat quàntic en els eixos X i Y de l'esfera de Bloch respectivament:

$$\{|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\ |-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\} \tag{8.10}$$

$$\{|+i\rangle := \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle),\ |-i\rangle := \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)\} \tag{8.11}$$

L'estat resultant de la mesura en qualsevol base vindrà donat de forma probabilística com postula la regla de Born. La regla de Born és un postulat fonamental de la mecànica quàntica que diu el següent:

Postulat 1 (Regla de Born) *La probabilitat que un estat $|\psi\rangle$ col·lapsi durant una mesura en la base $\{|x_1\rangle, |x_2\rangle\}$ en l'estat $|x_1\rangle$ ve donat per:*

$$P(x_1) = |\langle x_1|\psi\rangle|^2 \qquad , \sum_i P(x_i) = 1 \tag{8.12}$$

A continuació podem veure un exemple de mesura de l'estat $|\psi\rangle = \frac{1}{\sqrt{3}}(|0\rangle + \sqrt{2}|1\rangle)$ en la base $\{|0\rangle, |1\rangle\}$:

$$P(0) = |\langle 0|\psi\rangle|^2 = \left|\langle 0|\frac{1}{\sqrt{3}}(|0\rangle + \sqrt{2}|1\rangle)\right|^2 = \left|\frac{1}{\sqrt{3}}\langle 0|0\rangle + \sqrt{\frac{2}{3}}\langle 0|1\rangle\right|^2 = \frac{1}{3} \tag{8.13}$$

$$P(1) = |\langle 1|\psi\rangle|^2 = \left|\langle 1|\frac{1}{\sqrt{3}}(|0\rangle + \sqrt{2}|1\rangle)\right|^2 = \left|\frac{1}{\sqrt{3}}\langle 1|0\rangle + \sqrt{\frac{2}{3}}\langle 1|1\rangle\right|^2 = \frac{2}{3} \tag{8.14}$$

Com podem veure, l'estat anterior col·lapsarà en l'estat $|0\rangle$ amb una probabilitat de $P(0) = \frac{1}{3}$ i en l'estat $|1\rangle$ amb una probabilitat de $P(1) = \frac{2}{3}$. Si sumem les dues probabilitats, podem veure com hem comprès totes les opcions possibles, ja que tenim una probabilitat d'1.

Esfera de Bloch

Una forma de visualitzar els qubits és mitjançant l'esfera de Bloch a on podem veure els qubits com vectors dins de l'esfera. Podem escriure qualsevol estat normalitzat com:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{8.15}$$

a on $\phi \in [0, 2\pi]$ descriu una fase relativa i $\theta \in [0, \pi]$ determina la probabilitat de mesurar $|0\rangle$ o $|1\rangle$.

L'esfera de Bloch és una esfera de radi $|\vec{r}| = 1$ que podem veure en la figura 1.1 i a on les coordenades de cada estat venen donades pel Bloch vector $\vec{r} = \begin{pmatrix} sin\theta \cdot cos\phi \\ sin\theta \cdot sin\phi \\ cos\theta \end{pmatrix}$.

A continuació podem veure les coordenades dels sis estats que apareixen en les tres bases que hem vist anteriorment:

$$|0\rangle: \quad \theta = 0, \phi\ qualsevol \quad \rightarrow \quad \vec{r} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{8.16}$$

$$|1\rangle: \quad \theta = \pi, \phi\ qualsevol \quad \rightarrow \quad \vec{r} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{8.17}$$

$$|+\rangle: \quad \theta = \frac{\pi}{2}, \phi = 0 \quad \rightarrow \quad \vec{r} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{8.18}$$

$$|-\rangle: \quad \theta = \frac{\pi}{2}, \phi = \pi \quad \rightarrow \quad \vec{r} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \tag{8.19}$$

$$|+i\rangle: \quad \theta = \frac{\pi}{2}, \phi = \frac{\pi}{2} \quad \rightarrow \quad \vec{r} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{8.20}$$

$$|-i\rangle: \quad \theta = \frac{\pi}{2}, \phi = \frac{3\pi}{2} \quad \rightarrow \quad \vec{r} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \tag{8.21}$$

Cal remarcar que en l'esfera de Bloch, els angles són el doble de grans que en l'espai de Hilbert, ja que ens interessa que vectors ortogonals com els dels estats $|0\rangle$ i $|1\rangle$ tinguin un angle de 180^{o} i no de 90^{o}. També és interessant destacar que les mesures en cadascun dels diferents eixos es correspon en una projecció de l'estat en aquell eix.

8.2 Circuits quàntics

En la teoria de computabilitat podem trobar diferents models de computació com per exemple els autòmats finits, els autòmats amb pila o les màquines de Turing, entre d'altres. En computació quàntica s'utilitza un model anomenat circuit model en el qual utilitzarem circuits que anomenarem circuits quàntics i a on la computació consisteix en una seqüència de portes quàntiques, mesures, inicialització de qubits i altres possibles accions.

En la següent figura podem veure un exemple d'un circuit quàntic a on tenim un conjunt de qubits inicialitzats a l'estat $|0\rangle$ i a on se li apliquen un conjunt de portes quàntiques. Cada qubit té una línia horitzontal que ens serveix com a línia de temps per les diferents portes i que llegirem d'esquerra a dreta. Aquestes línies es poden confondre amb cables a causa de la seva semblança amb els circuits digitals, però en aquest cas, només estan representant una sèrie d'esdeveniments sobre els diferents qubits.

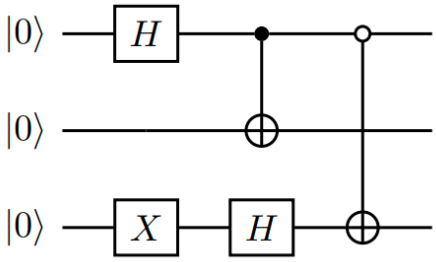


Figura 8.1: Exemple de circuit quàntic
Font: Elaboració pròpia

8.3 Portes quàntiques

En els circuits quàntics, per tal d'operar sobre els qubits utilitzem portes quàntiques. Aquestes portes es poden representar en forma de matriu i, per tant, la seva aplicació serà en forma de multiplicació sobre els vectors que representen els estats quàntics que hem vist a l'apartat 8.1. Com que en computació quàntica tots els canvis sobre un estat han de poder ser revertits (menys les mesures) i alhora volem preservar la norma dels qubits després de l'aplicació d'una porta (els estats quàntics estan normalitzats), les diferents matrius que utilitzem han de ser unitàries, és a dir, si U és una matriu i $U^\dagger$ és la seva matriu conjugada transposada, $U^\dagger U = I$, a on I és la matriu identitat.

Sabent això, en aquest apartat veurem totes les portes quàntiques que utilitzarem posteriorment en aquest treball. Començant per les portes que afecten un sol qubit i acabant per les portes multi-qubit, veurem la representació matricial de cada una juntament amb la simbologia que farem servir per referir-nos a elles dins dels circuits.

8.3.1 Portes quàntiques d'un qubit

Existeixen moltes portes quàntiques d'un sol qubit, però per la realització d'aquest treball només necessitarem fer ús d'un subconjunt d'elles.

Porta Pauli-X

Aquesta porta és l'equivalent quàntic a la porta lògica NOT que podem utilitzar en computació clàssica respecte a la base estàndard $\{|0\rangle, |1\rangle\}$, la qual està representada en l'eix Z de l'esfera de Bloch que hem vist anteriorment. Aquesta transformació rota el nostre estat quàntic π radiants al voltant de l'eix X de l'esfera de Bloch. Podem representar aquesta porta amb la següent matriu:

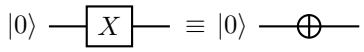
$$X = \sigma_x = NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{8.22}$$

A continuació podem veure un exemple de l'aplicació d'aquesta porta sobre l'estat en superposició de l'expressió 8.1 utilitzant tant la notació de Dirac com la seva forma matricial:

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = (\alpha|1\rangle + \beta|0\rangle) \tag{8.23}$$

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \tag{8.24}$$

Per tal d'utilitzar la porta en un circuit quàntic tenim dues representacions possibles:



Porta Hadamard

La porta Hadamard és una de les portes més útil que existeixen, ja que ens permet crear una superposició quan aquesta s'aplica als estats base $|0\rangle$ i $|1\rangle$. Si ens fixem en l'esfera de Bloch, la transformació pot ser vista com una rotació de $\frac{\pi}{2}$ radians sobre l'eix Y seguida d'una rotació de π radiants al voltant de l'eix X. Aquesta és la seva matriu:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8.25}$$

Seguidament, podem trobar un exemple de l'aplicació de la porta sobre els estats $|0\rangle$ i $|1\rangle$:

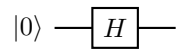
$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \tag{8.26}$$

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+\rangle \tag{8.27}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{8.28}$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = |-\rangle \tag{8.29}$$

Com veiem, la transformació pot ser vista com un canvi de base entre $\{|0\rangle, |1\rangle\}$ i $\{|+\rangle, |-\rangle\}$, ja que si la tornem a aplicar als estats resultats de les expressions anteriors, recuperaríem els estats inicials. Finalment, podem veure el símbol que utilitzarem per representar aquesta transformació:



Porta Ry

La porta Ry és un dels operadors de rotació que podem utilitzar. Aquesta porta ens permet rotar el nostre estat $\frac{\theta}{2}$ radiants al voltant de l'eix Y, i la podem representar amb la següent matriu:

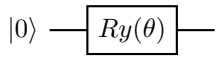
$$Ry(\theta) = \begin{pmatrix} cos\left(\frac{\theta}{2}\right) & -sin\left(\frac{\theta}{2}\right) \\ sin\left(\frac{\theta}{2}\right) & cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{8.30}$$

Per veure un exemple d'una aplicació d'aquesta porta, podem rotar l'estat $|0\rangle$ $\frac{\pi}{2}$ radiants i veurem com aconseguim el mateix efecte que la porta NOT:

$$Ry(\pi)|0\rangle = |1\rangle \tag{8.31}$$

$$Ry(\pi)|0\rangle = \begin{pmatrix} cos\left(\frac{\pi}{2}\right) & -sin\left(\frac{\pi}{2}\right) \\ sin\left(\frac{\pi}{2}\right) & cos\left(\frac{\pi}{2}\right) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} cos\left(\frac{\pi}{2}\right) \cdot 1 - sin\left(\frac{\pi}{2}\right) \cdot 0 \\ sin\left(\frac{\pi}{2}\right) \cdot 1 + cos\left(\frac{\pi}{2}\right) \cdot 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{8.32}$$

Per acabar, aquí podem veure el símbol que utilitzarem per representar la porta en un circuit:



Porta Phase shift

Aquesta és una porta que afecta a un sol qubit la qual ens permet afegir una fase a un qubit quan aquest es troba en l'estat $|1\rangle$.

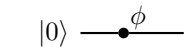
$$R(\phi)|0\rangle = |0\rangle \tag{8.33}$$

$$R(\phi)|1\rangle = e^{i\phi}|1\rangle \tag{8.34}$$

Els canvis que realitzem amb aquesta porta no afecten la probabilitat de mesurar un estat o un altre, però sí que modifiquen la seva fase i per tant ens serà útil en termes d'interferència. La matriu que representa aquesta transformació és la següent:

$$R(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \tag{8.35}$$

A l'hora d'utilitzar aquesta transformació en un circuit farem servir el següent símbol:



Generalització - Porta U

La porta U ens permet realitzar una rotació en l'esfera de Bloch especificant tres angles diferents. Aquesta porta és capaç de replicar qualsevol altra rotació sobre un qubit i ens serà d'utilitat a l'hora d'implementar els algorismes en codi, ja que en funció dels simuladors que utilitzem no tindrem totes les portes disponibles. Aquí podem veure la seva matriu:

$$U(\theta, \phi, \lambda) = \begin{pmatrix} cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{8.36}$$

A continuació podem trobar les equivalències amb les quatre portes esmentades anteriorment:

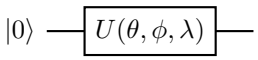
$$U(\pi, 0, \pi) = X \tag{8.37}$$

$$U(\frac{\pi}{2}, 0, \pi) = H \tag{8.38}$$

$$U(\theta, 0, 0) = Ry(\theta) \tag{8.39}$$

$$U(0, \phi, 0) = R(\phi) \tag{8.40}$$

El símbol que ens representarà aquesta porta és el següent:



8.3.2 Portes quàntiques multi-qubit

A més de les portes d'un sol qubit que acabem de veure, per la implementació dels diferents algorismes, necessitarem portes multi-qubit.

Porta Control NOT

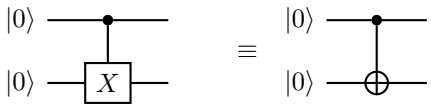
Aquesta porta, que també anomenarem CNOT, ens permet aplicar la porta NOT vista en l'apartat anterior, però utilitzant dos qubits, un d'ells actuarà com a control i l'altre com a target. Aplicarem la porta NOT al qubit target, si i només si, el qubit de control està en l'estat $|1\rangle$, i en cas contrari no farem res. Aquesta porta no fa cap canvi en el qubit de control.

$$CNOT|x\rangle \otimes |y\rangle = |x\rangle \otimes |x \oplus y\rangle \tag{8.41}$$

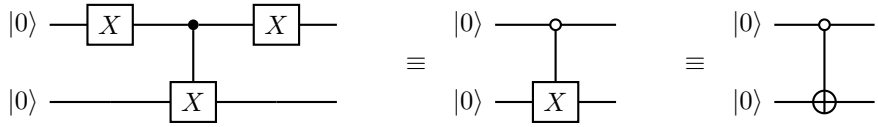
A diferència de les portes anteriors, aquesta porta es representa amb una matriu 4×4 pel fet que necessitem dos qubits per aplicar-la.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{8.42}$$

Aquesta porta es pot representar de dues maneres com també passava amb la porta NOT. El primer qubit, el qual té un punt negre, l'utilitzem com a control, mentre que el segon qubit és el target:



Hi haurà casos en els quals ens pot interessar aplicar la porta NOT al qubit target quan el qubit de control sigui $|0\rangle$ en comptes de quan sigui $|1\rangle$. En aquests casos podem utilitzar dues portes NOT al qubit de control abans i després d'utilitzar la CNOT. A l'hora de representar aquest cas en un circuit podem dibuixar les portes NOT o utilitzar un punt blanc en comptes de negre en el qubit de control:



Porta Control Hadamard

La següent porta que necessitarem pels nostres algorismes és la porta Hadamard amb control. Aquesta porta funciona de la mateixa manera que l'anterior, però en aquest cas, aplicarem la porta Hadamard en comptes de la NOT al qubit target. Per tal de veure quina matriu tindrà aquesta transformació,

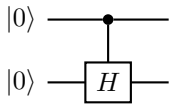
podem veure primer com en la matriu de la porta CNOT, que trobem en l'expressió 8.42, a baix a la dreta podem trobar la matriu NOT 2×2 que hem vist anteriorment.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{8.43}$$

Això és a causa que només aplicarem aquella porta quan el primer qubit sigui $|1\rangle$. Aquest fet ens dona una idea de com poder generalitzar qualsevol porta que tingui un qubit de control, ja que només haurem d'inserir la matriu 2×2 de la transformació d'un qubit en lloc de les posicions marcades en blau. En el cas de la porta control Hadamard tindrem:

$$CH = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{8.44}$$

Per tal de representar la porta en un circuit utilitzarem un símbol molt semblant al de la porta CNOT, ja que només haurem de canviar la porta quàntica que s'aplica al qubit target:

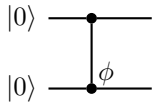


Porta Control Phase shift

Aquesta transformació, també anomenada porta CR o CPhase, de forma anàloga a la porta control Hadamard, aplica la porta phase shift a un qubit target en funció del valor d'un qubit de control. Com hem vist anteriorment, per crear la matriu només hem de substituir la caixa blava de l'expressió 8.43 per la matriu 2×2 que hem vist en l'apartat anterior.

$$CR(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix} \tag{8.45}$$

El símbol que farem servir per a aquesta porta quàntica és el següent:



Porta Toffoli

Una de les portes quàntiques que més utilitzarem en aquest treball és la porta Toffoli. Aquesta porta és universal, és a dir, que qualsevol computació es pot fer en termes exclusivament d'aquesta porta. És també coneguda com a CCNOT o control control NOT i necessita tres qubits per poder ser aplicada. Aquesta porta és molt semblant a la porta CNOT, però podrem fer servir més d'un qubit de control que decidiran si aplicar una porta NOT al target o no. Aquí podem trobar la seva matriu:

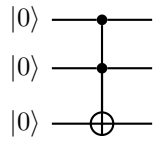
$$CCNOT = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{8.46}$$

Si ens fixem en la matriu anterior, veiem que té la mateixa forma que la matriu de la porta CNOT, ja que a baix a la dreta, en les entrades corresponent a quan els dos primers qubits, que actuant com a control, són $|11\rangle$, podem trobar la porta NOT d'un qubit.

Si inicialitzem el qubit target a $|0\rangle$, podem veure la porta Toffoli com una porta AND, ja que acabarem en l'estat $|1\rangle$ en el target si i nomes si, els dos qubits de control estan també a l'estat $|1\rangle$.

$$CCNOT(|x\rangle \otimes |y\rangle \otimes |z\rangle) = |x\rangle \otimes |y\rangle \otimes |(x \wedge y) \oplus z\rangle \tag{8.47}$$

La seva representació dins d'un circuit serà la mateixa que la que teníem en la porta CNOT però amb dos qubits de control:



En aquest treball també utilitzarem variants d'aquesta porta a on podem utilitzar més de dos qubits de control, tot i que la representació serà la mateixa afegint els qubits de control que necessitem. Aquesta variant s'anomena multi-controlled Toffoli o multi-controlled X i es pot descompondre en petits portes Toffoli com les explicades anteriorment i diversos qubits auxiliars.

Porta Swap

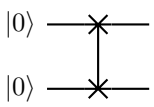
Aquesta és una porta molt sencilla de dos qubits que ens permet intercambiar dos qubits de posició.

$$SWAP(|x\rangle|y\rangle) = |y\rangle|x\rangle \tag{8.48}$$

Aquesta és la seva matriu a on podem veure que els estats $|00\rangle$ i $|11\rangle$ no es veuen afectats:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8.49}$$

El símbol que farem servir per representar la porta SWAP consisteix en dues creus en els dos qubits que volem intercanviar:



Capítol 9

Algorisme de cerca per 3-SAT

En aquest projecte com s'explica a l'apartat 1.2, anomenat "Problema a resoldre, 3-SAT", dissenyarem un algorisme capaç de trobar solucions a instàncies del problema 3-SAT. L'enfocament que veurem en aquest capítol consisteix a fer una cerca en l'espai de totes les possibles assignacions de les variables d'una fórmula per trobar aquelles que la satisfacin.

Abans de dissenyar l'algorisme quàntic podem veure una forma naïf de resoldre el problema clàssicament mitjançant força bruta. Com veiem en el següent procediment podem comprovar totes les possibles assignacions començant per la primera fins a trobar una que satisfà la fórmula.

Algorithm 1 Algorisme de força bruta per 3-SAT

```
Input: Fórmula f en 3-CNF amb n variables
Output: Una assignació que satisfà la fórmula f o -1 si f és insat
assig = 0
while assig < 2^n do
    if assig satisfà la fórmula f then
        Retornar assig
    end if
    assig += 1
end while
Retornar −1
```

Si tenim un algorisme que actua com un oracle per comprovar si una assignació satisfà la fórmula i ens respon en temps polinòmic, el procediment anterior, en el pitjor dels casos, haurà de comprovar totes les assignacions possibles i tindrà una complexitat de $O(2^n \cdot poly(n))$. Tot i ser un algorisme molt poc eficient ens dona una idea sobre com poder abordar el problema quànticament, ja que podem realitzar la cerca explotant conceptes com la superposició per aconseguir un algorisme millor.

En els següents apartats s'introdueixen dos algorismes quàntics, com són l'algorisme de Grover i el Quantum counting, que ens serviran per dissenyar el nostre algorisme per resoldre 3-SAT al final del capítol.

9.1 Algorisme de Grover

L'algorisme de Grover és un algorisme quàntic que ens permet realitzar una cerca en una seqüència no ordenada de N elements realitzant $O(\sqrt{N})$ comprovacions de què un element és el que estem buscant

o no. Comparant-ho amb un algorisme clàssic, ens permet aconseguir un speedup quadràtic, ja que, per trobar l'element que estem buscant en la seqüència, hauríem de comprovar en mitjana $\frac{N}{2}$ elements i en el pitjor cas els N elements.

Per tal de poder dur a terme aquestes comprovacions suposarem que tenim un oracle capaç de saber si un element és el que estem buscant o no. Aquest element pot ser vist com la solució de la cerca i no ha de per què ser únic.

9.1.1 Creació de l'oracle

Abans de començar a detallar l'algorisme de Grover podem veure amb més detall com funciona l'oracle que utilitzarem. L'oracle actuarà sobre tots els estats quàntics afegint una fase negativa a tots aquells estats que siguin solució del nostre problema. Si suposem que només estem buscant un estat solució anomenat $|\omega\rangle$ podem veure l'aplicacio de l'oracle sobre un estat qualsevol $|x\rangle$ de la següent manera:

$$O_\omega|x\rangle = \begin{cases} |x\rangle & \text{if } x \neq \omega \\ -|x\rangle & \text{if } x = \omega \end{cases} \tag{9.1}$$

L'oracle també es pot representar com una matriu diagonal a on l'entrada corresponent a l'element que estem buscant tindrà la fase negativa. Per exemple, si tenim un total de 3 qubits i $\omega = 011$, tindrem la següent matriu:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9.2}$$

Existeixen molts problemes que són computacionalment difícils de trobar una solució, però que, en canvi, són fàcilment verificables. Per aquests problemes podem crear una funció f amb una entrada x que sigui capaç de retornar $f(x) = 1$ si x és solució del problema i $f(x) = 0$ en cas contrari. Per representar el canvi de fase que farem en els estats solució podem utilitzar un qubit auxiliar que negarem depenent del que retorni la funció f. Per tant, l'aplicació de l'oracle O es pot veure com:

$$|x\rangle|q\rangle \overset{O}{\longmapsto} |x\rangle|q \oplus f(x)\rangle, \tag{9.3}$$

a on $|x\rangle$ és l'estat que estem comprovant si és solució o no, $\oplus$ representa una suma mòdul 2 i $|q\rangle$ és un qubit auxiliar en el que el seu valor serà negat només si $f(x) = 1$. El qubit $|q\rangle$ serà inicialitzar a l'estat $|-\rangle = \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$, ja que quan el neguem aconseguim el mateix afecte que afegint una fase negativa al davant. Per tant, podem representar l'aplicació anterior com:

$$|x\rangle|-\rangle \overset{O}{\longmapsto} (-1)^{f(x)}|x\rangle|-\rangle \tag{9.4}$$

Com que el segon qubit sempre es queda igual, no caldrà que el mesurem i, per tant, el podem ometre:

$$|x\rangle \overset{O}{\longmapsto} (-1)^{f(x)}|x\rangle \tag{9.5}$$

9.1.2 Pasos a seguir

L'algorisme de Grover es pot descompondre en 5 passos que analitzarem a continuació:

1. Crear l'estat d'igual superposició de tots els qubits
2. Aplicar l'oracle O
3. Aplicar la transformació Hadamard $H^{\otimes n}$
4. Afegir una fase de -1 a tots els estats excepte a l'estat $|0\rangle^{\otimes n}$ (Difusor)
5. Aplicar la transformació Hadamard $H^{\otimes n}$

Els passos 2, 3, 4 i 5 es poden agrupar en el que anomenarem *Operador de Grover* o *Iteració de Grover*. En la figura 9.1 podem veure l'aspecte que tindrà el circuit a on utilitzarem n qubits en superposició i un conjunt de qubits auxiliars que variarà depenent del problema que volem resoldre.

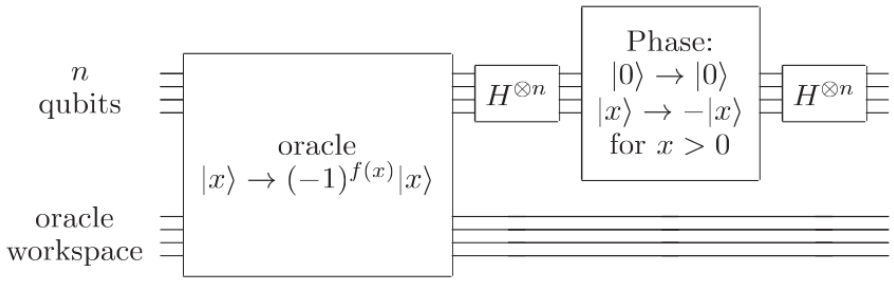


Figura 9.1: Operador de Grover
Font: Quantum Computation and Quantum Information [11]

El circuit de la figura anterior es pot encapsular en una porta que anomenarem G i que haurem de repetir k cops per tal de maximitzar la probabilitat d'obtenir un estat solució com veurem més endavant. A la figura 9.2, veiem que l'operador de Grover s'aplica després del pas 1 que consisteix a crear una superposició.

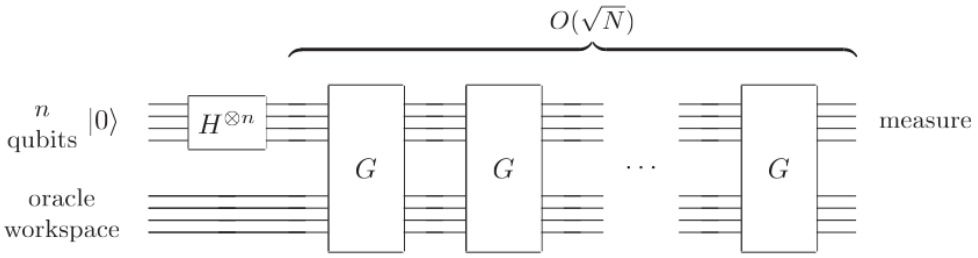


Figura 9.2: Algorisme de Grover
Font: Quantum Computation and Quantum Information [11]

Pas 1: Crear l'estat d'igual superposició de tots els qubits

En el primer pas, per crear la superposició, aplicarem una transformació Hadamard als n primers qubits els quals es troben inicialitzats a l'estat $|0\rangle^{\otimes n}$ com podem veure a la figura 9.2. Si a l'estat resultat l'anomenarem $|\psi\rangle$ i definim $N = 2^n$ tenim el següent:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{9.6}$$

L'estat anterior representa la superposició de tots els possibles elements, però, definint M com el nombre total de solucions, podem descompondre el sumatori en, per una banda, els estats que no són solució, d'ara endavant, $|\alpha\rangle$, i per altre, els estats que si són solució, a partir d'ara, $|\beta\rangle$:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \in NoSol} |x\rangle \tag{9.7}$$

$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{x \in Sol} |x\rangle \tag{9.8}$$

En els dos estats anteriors estem englobant tots els elements possibles i, per tant, podem representar el nostre estat $|\psi\rangle$ de la següent manera:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle \tag{9.9}$$

Pas 2: Aplicar l'oracle O

En aquest pas, com hem vist anteriorment, afegirem una fase negativa a tots els estats que són solució. Com en el pas 1 hem definit els estats que són solució com $|\beta\rangle$, podem veure l'oracle en notació de Dirac de la següent manera:

$$O = I - 2|\beta\rangle\langle\beta| \tag{9.10}$$

En l'expressió anterior I representa la matriu identitat i si l'apliquem a l'estat $|\psi\rangle$ podem veure com realment afegeix una fase negativa als estats solució i deixa igual els altres.

$$O|\psi\rangle = (I - 2|\beta\rangle\langle\beta|) \left(\sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle\right) \tag{9.11}$$

$$= \sqrt{\frac{N-M}{N}}(|\alpha\rangle - 2|\beta\rangle\langle\beta|\alpha\rangle) + \sqrt{\frac{M}{N}}(|\beta\rangle - 2|\beta\rangle\langle\beta|\beta\rangle) \tag{9.12}$$

$$= \sqrt{\frac{N-M}{N}}|\alpha\rangle - \sqrt{\frac{M}{N}}|\beta\rangle \tag{9.13}$$

El pas de l'expressió 9.12 a la 9.13 és pel fet que $\langle\beta|\alpha\rangle = 0$ i $\langle\beta|\beta\rangle = 1$. Això és així perquè $|\alpha\rangle$ i $|\beta\rangle$ són ortogonals, ja que no comparteixen cap estat i tots els estats es poden veure com vectors de la base canònica en un espai N-dimensional.

Pas 3, 4 i 5: Inversió sobre la mitjana

Els passos 3, 4 i 5 es poden tractar junts en un pas que anomenarem inversió sobre la mitjana. Abans de fer-ho, podem representar el pas 4 amb la notació de Dirac com hem fet al pas 2:

$$D_4 = 2|0\rangle\langle 0|^{\otimes n} - I \tag{9.14}$$

A l'expressió resultant del pas 4, si li apliquem per davant i de darrere les Hadamard $H^{\otimes n}$ dels passos 3 i 5 i tenim en compte com hem vist al pas 1 que $H^{\otimes n}|0\rangle = |\psi\rangle$, obtenim el següent que anomenarem D:

$$D = H^{\otimes n} D_4 H^{\otimes n} \tag{9.15}$$
$$= H^{\otimes n}(2|0\rangle\langle 0|^{\otimes n} - I)H^{\otimes n} \tag{9.16}$$
$$= 2|\psi\rangle\langle\psi| - I \tag{9.17}$$

Les transformacions 9.10 i 9.17 es poden veure com dues matrius i podem representar el circuit de la figura 9.1 com el seu producte de la següent manera tenint en compte que l'ordre d'aplicació de les matrius en el producte és de dreta a esquerra:

$$G = DO = (2|\psi\rangle\langle\psi| - I)\ (I - 2|\beta\rangle\langle\beta|) \tag{9.18}$$

9.1.3 Interpretació geomètrica i complexitat

En aquest apartat veurem una interpretació geomètrica de l'algorisme per acabar d'entendre millor el seu funcionament i també ens servirà per determinar la seva complexitat. Com hem comentat abans podem representar el nostre estat $|\psi\rangle$ normalitzat en funció dels estats $|\alpha\rangle$ i $|\beta\rangle$ i, ja que aquests dos últims són ortogonals poder visualitzar el nostre estat com un vector en un espai de 2 dimensions de la següent manera:

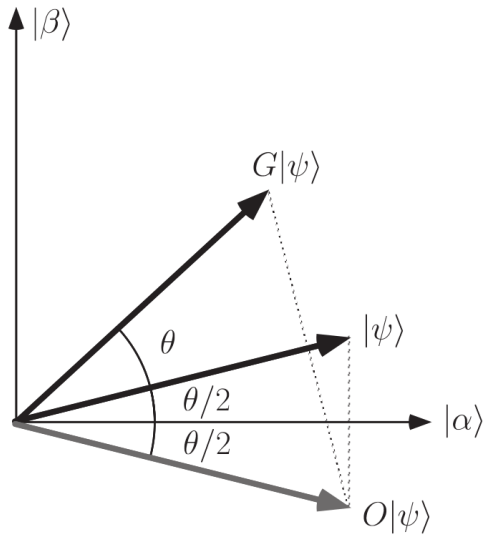


Figura 9.3: Acció d'una sola iteració de Grover
Font: Quantum Computation and Quantum Information [11]

En la figura 9.3, veiem el nostre estat $|\psi\rangle$ amb un cert angle $\frac{\theta}{2}$ d'obertura que en aquest apartat veurem com calcular. També podem veure l'aplicació de l'oracle O i finalment l'aplicació de G, que és equivalent, a l'aplicació de la transformació D a l'estat resultant de l'aplicació de l'oracle. Si ens fixem en les expressions 9.10 i 9.17 podem veure com estem realitzant dues reflexions, una sobre l'estat $|\beta\rangle$ i l'altre sobre l'estat $|\psi\rangle$. Com sabem que dues reflexions seguides equivalen a una rotació podem concloure que cada cop que apliquem els passos 2, 3, 4 i 5 estem rotant el nostre estat en un cert angle θ.

L'estat $|\psi\rangle$, es pot representar en funció de l'angle θ de la següent manera:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|\alpha\rangle + sin\left(\frac{\theta}{2}\right)|\beta\rangle \tag{9.19}$$

Si igualem l'amplitud que multiplica l'estat $|\beta\rangle$ en l'expressió 9.9 amb el sinus de l'expressió 9.19, podem veure el valor de l'angle θ:

$$sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{M}{N}} \tag{9.20}$$

$$\theta = 2 \cdot arcsin\left(\sqrt{\frac{M}{N}}\right) \tag{9.21}$$

Amb la rotació esmentada anteriorment que estem realitzant cada cop que apliquem l'operador de Grover, estem apropant l'estat $|\psi\rangle$ a l'estat de les solucions $|\beta\rangle$. Aquest apropament significa augmentar l'amplitud de tots els estats solució i, per tant, augmentar la probabilitat d'obtenir una solució quan mesurem. Cada iteració que fem de l'algorisme rotem l'estat θ radiants i, en conseqüència, després de k iteracions arribaríem al següent estat:

$$|\psi\rangle = cos\left((2k+1)\cdot\frac{\theta}{2}\right)|\alpha\rangle + sin\left((2k+1)\cdot\frac{\theta}{2}\right)|\beta\rangle \tag{9.22}$$

Si realitzem més iteracions del compte correm el risc sobrepassar l'estat $|\beta\rangle$ i acabar amb una probabilitat pitjor d'obtenir una solució. Per tal d'acotar superiorment el nombre d'iteracions que farem, calcularem quan ha de valer k en l'expressió 9.22 per tal de tenir una probabilitat d'1 d'aconseguir una solució, ja que seria el millor dels casos.

$$sin^2\left((2k+1)\cdot\frac{\theta}{2}\right) = 1 \tag{9.23}$$

Això passarà quan l'angle de dins del sinus al quadrat de l'expressió anterior sigui igual a $\frac{\pi}{2}$ radiants, és a dir 90 graus:

$$(2k+1)\cdot\frac{\theta}{2} = \frac{\pi}{2} \tag{9.24}$$

A partir de l'expressió 9.24 podem determinar una fita superior del nombre de repeticions que farem de l'operador G:

$$k = \frac{\pi}{4}\cdot\frac{1}{arcsin\left(\sqrt{\frac{M}{N}}\right)} - \frac{1}{2} \leq \frac{\pi}{4}\cdot\sqrt{\frac{N}{M}} - \frac{1}{2} \leq \left\lceil\frac{\pi}{4}\cdot\sqrt{\frac{N}{M}}\right\rceil \tag{9.25}$$

L'expressió anterior ens ajuda a veure quina serà la complexitat de l'algorisme. Com veiem, podrem trobar una solució amb $O(\sqrt{\frac{N}{M}})$ repeticions de l'oracle. En pitjor cas només tindrem una solució o un element a buscar i, per tant, arribem a la complexitat de $O(\sqrt{N})$ que es comentava al principi del capítol.

Normalment, acostumarem a tenir més d'una solució i com veiem, el nombre de crides a l'oracle depèn del nombre total d'elements i del nombre total de solucions. Això pot ser un problema si no coneixem el nombre total de solucions abans d'executar l'algorisme de Grover. Existeixen diverses tècniques per solucionar aquest problema, però en aquest treball utilitzarem un altre algorisme anomenat Quantum counting com a pas previ al càlcul de les iteracions com veurem en el següent apartat. Un altre problema seria que tinguem més de la meitat d'elements com a solucions, ja que llavors amb una sola iteració de l'algorisme sobrepassaríem l'estat $|\beta\rangle$. Per solucionar això, utilitzarem un qubit extra en la superposició per tal d'ampliar l'espai de cerca al doble i només marcarem com a solució els estats a on aquest nou qubit sigui 0 per tal d'assegurar-nos de tenir com a mínim la meitat d'elements com a no solució.

9.2 Quantum counting

En aquest apartat veurem el funcionament de l'algorisme Quantum counting el qual ens permet contar el nombre de solucions d'un problema determinat basant-se en l'algorisme de quantum phase estimation . Per entendre el funcionament de l'algorisme ens ajudarem de la interpretació geomètrica de Grover de l'apartat anterior. Com hem vist, en un espai 2D podem veure l'algorisme de Grover com una rotació i, per tant, poder representar una iteració del seu operador amb la següent matriu:

$$G = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \tag{9.26}$$

Si mirem els valors propis que té la matriu G podem veure com són $e^{i\theta}$ i $e^{i(2\pi-\theta)}$. En ser una matriu unitària i tenir valors propis diferents sabem que també tenim dos vectors propis diferents que són ortogonals entre ells i, per tant, podem representar el nostre estat $|\psi\rangle$ en funció d'ells de la següent manera:

$$|\psi\rangle = \gamma|v_1\rangle + \delta|v_2\rangle \tag{9.27}$$

L'estat $|\psi\rangle$ en l'expressió 9.27 està en superposició dels dos vectors propis que anomenem $|v_1\rangle$ i $|v_2\rangle$. Aquesta representació de l'estat serà la que utilitzarem en aquest apartat per entendre l'algorisme.

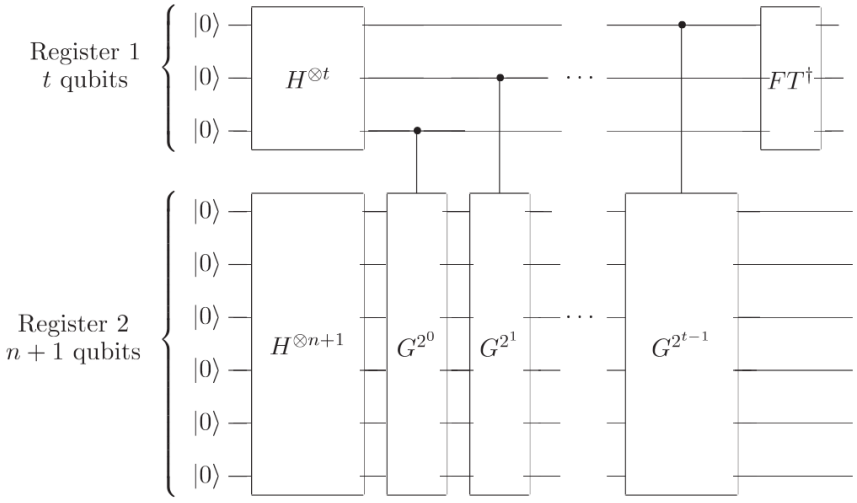


Figura 9.4: Algorisme Quantum counting
Font: Quantum Computation and Quantum Information [11]

En la figura 9.4 podem veure el circuit que ens representa l'algorisme a on tenim dos registres diferents. El primer registre conté t qubits en superposició que ens serviran per calcular l'angle θ d'una iteració de l'algorisme de Grover, mentre que el segon registre, després d'aplicar les portes Hadamard, l'utilitzarem per representar l'estat $|\psi\rangle$ amb un qubit extra per crear l'espai ampliat que es comentava al final de l'apartat anterior. Com el registre 2 és una superposició dels vectors propis, podem representar l'estat de la següent manera:

$$|\tilde{x}\rangle = \frac{\gamma}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|v_1\rangle + \frac{\delta}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|v_2\rangle \tag{9.28}$$

L'algorisme consisteix a aplicar repetides vegades l'operador de Grover, però amb la seva versió amb control. Si reagrupem els qubits en superposició i mirem quin és l'efecte de totes les aplicacions de l'operador de Grover, però només sobre el vector propi $|v_1\rangle$ arribem a la següent expressió:

$$|\tilde{x}\rangle = \frac{\gamma}{\sqrt{2^t}} \cdot (|0\rangle + e^{i\theta 2^{t-1}}|1\rangle) \cdot (|0\rangle + e^{i\theta 2^{t-2}}|1\rangle) \cdot ... \cdot (|0\rangle + e^{i\theta 2^0}|1\rangle) \cdot |v_1\rangle \tag{9.29}$$

Com podem veure estem multiplicant repetides vegades pel seu valor propi corresponent. En l'expressió anterior també s'hauria de fer el mateix amb el vector propi $|v_2\rangle$ i el valor propi $e^{i(2\pi-\theta)}$, però per simplificar l'explicació només veurem el procés amb el primer vector propi. Cal remarcar que al final de l'algorisme acabarem amb una superposició i, per tant, podrem mesurar tant el primer vector propi com el segon.

Per aplicar l'últim pas cal definir la transformada de Fourier quàntica, ja que acabarem aplicant la seva inversa. Aquesta transformació funciona igual que la seva versió discreta clàssica i la podem representar de la següent manera:

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i x y}{N}} |y\rangle \tag{9.30}$$

$$= \frac{1}{\sqrt{N}} \sum_{y_1=0}^{1} \sum_{y_2=0}^{1} ... \sum_{y_n=0}^{1} e^{2\pi i x (\sum_{l=1}^{n} y_l \cdot 2^{-l})} |y_1 y_2 ... y_n\rangle \tag{9.31}$$

$$= \frac{1}{\sqrt{N}} \sum_{y_1=0}^{1} \sum_{y_2=0}^{1} ... \sum_{y_n=0}^{1} \otimes_{l=1}^{n} e^{2\pi i x y_l \cdot 2^{-l}} |y_1 y_2 ... y_n\rangle \tag{9.32}$$

$$= \frac{1}{\sqrt{N}} \otimes_{l=1}^{n} \left[\sum_{y=0}^{1} e^{\frac{2\pi i x y}{N}} |y\rangle \right] \tag{9.33}$$

$$= \frac{1}{\sqrt{N}} \cdot (|0\rangle + e^{\frac{2\pi i x}{2^1}}|1\rangle) \cdot (|0\rangle + e^{\frac{2\pi i x}{2^2}}|1\rangle) \cdot ... \cdot (|0\rangle + e^{\frac{2\pi i x}{2^n}}|1\rangle) \tag{9.34}$$

Com podem veure per tal d'arribar al resultat hem anat canviant la notació del nostre qubit entre la seva forma en decimal $|y\rangle$ i la seva forma en binari $|y_1 y_2 ... y_n\rangle$. Si ens fixem en l'expressió final que hem obtingut veiem que és molt semblant a l'expressió 9.29 i, per tant, si apliquem la transformació inversa, podem obtenir l'angle θ que estem buscant.

Si comparem les expressions 9.29 i 9.34 veiem que només difereixen del terme que multiplica l'estat $|1\rangle$ en cada qubit. Sabent això, podem igualar els dos primers termes i veure quin valor obtenim si apliquem la inversa de la transformada de Fourier al primer registre.

$$e^{\frac{2\pi i x}{2^1}} = e^{i \cdot \theta \cdot 2^{t-1}} \tag{9.35}$$

$$\frac{2\pi i x}{2^1} = i \cdot \theta \cdot 2^{t-1} \tag{9.36}$$

$$x = \frac{\theta \cdot 2^t}{2\pi} \tag{9.37}$$

De l'expressió 9.37 podem concloure que l'angle θ que estem buscant serà $\theta = \frac{2\pi x}{2^t}$ a on la x serà el valor mesurat al final del Quantum counting. Cal recordar que només estem fent els càlculs amb el primer vector propi i, per tant, a la pràctica podem trobar tant θ com $2\pi - \theta$. Un cop sabem l'angle podem tornar a l'expressió 9.21 i calcular el nombre de solucions M. Com podem veure a continuació qualsevol valor propi ens portarà al resultat correcte.

$$M = sin^2\left(\frac{\theta}{2}\right) \cdot N = sin^2\left(\frac{2\pi - \theta}{2}\right) \cdot N \tag{9.38}$$

Finalment, es pot veure l'algorisme només necessita $\Theta(\sqrt{N})$ crides a l'oracle de Grover si establim el nombre de qubits en el primer registre com $t = \lceil \frac{n}{2} \rceil + 3$.

$$\sum_{t=0}^{\lceil \frac{n}{2} \rceil + 2} 2^t = 2^{\lceil \frac{n}{2} \rceil + 3} - 1 = O(\sqrt{2^n}) \tag{9.39}$$

L'angle que aconseguim al final aquest algorisme és aproximat i a més gran sigui el nombre de qubits t que utilitzem, més petit serà l'error d'aproximació. Per a més informació sobre l'algorisme Quantum counting es pot consultar al llibre *Quantum Computation and Quantum Information* [11] a la pàgina 263.

9.3 Descripció de l'algorisme

Un cop explicats els algorismes de Grover i Quantum counting ja podem veure en què consistirà el nostre algorisme per resoldre 3-SAT. La idea principal és utilitzar l'algorisme de Grover per a buscar una assignació que satisfaci una fórmula donada. Com no podem saber quantes solucions tenim abans d'executar Grover, farem ús de l'algorisme Quantum counting com a pas previ. Aquí podem veure els passos a seguir:

1. Crear l'operador de Grover específic per 3-SAT.
2. Utilitzar Quantum counting per calcular el nombre de solucions de la fórmula 3-CNF donada.
3. Calcular el nombre de repeticions de l'operador de Grover que farem.
4. Utilitzar l'algorisme de Grover amb el nombre de repeticions de l'operador calculades anteriorment per aconseguir una solució.

Per poder entendre com aplicar tots els passos necessitem veure com funciona l'algorisme de Grover aplicat a 3-SAT, ja que veurem com crear l'operador de Grover que necessitem en el Quantum counting (porta G en la figura 9.4) i com representarem les diferents assignacions. Tot i no ser el primer algorisme que farem servir en el procediment anterior, si és l'algorisme base i tots els altres passos ens serviran per executar-lo degudament.

Algorisme de Grover per resoldre 3-SAT

En la secció 9.1.2 podem veure els 5 passos que hem de realitzar per aplicar l'algorisme de Grover i en les figures 9.1 i 9.2 veiem el circuit sencer. En el nostre circuit utilitzarem n qubits inicialitzats a 0, a on cada qubit representarà una variable de la fórmula 3-CNF que volem resoldre. En aplicar una transformació Hadamard a cada qubit en el primer pas, aconseguim una superposició de tots els possibles valors que poden tenir les variables i, per tant, totes les assignacions possibles. Podem veure aquest efecte amb la següent fórmula d'exemple:

$$\phi = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \tag{9.40}$$

En la fórmula ϕ hi ha en total tres variables diferents $\{x_1, x_2, x_3\}$. Cada variable té dos possibles valors, així que, en total tenim $2^3 = 8$ assignacions diferents que van des de $\langle x_1 = 0, x_2 = 0, x_3 = 0\rangle$ fins $\langle x_1 = 1, x_2 = 1, x_3 = 1\rangle$. Si creem un circuit amb 1 qubit per variable i apliquem una Hadamard a cada un, aconseguim la següent expressió que, com podem veure, representa totes les assignacions possibles com volíem:

$$H^{\otimes 3}|000\rangle = \frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + ... + |110\rangle + |111\rangle) \tag{9.41}$$

Amb això ja podem representar l'espai d'estats en el que buscarem una solució i continuar amb el pas 2.

El següent pas consisteix a aplicar un oracle que ha de ser capaç de determinar si una assignació satisfà la fórmula o no. Per veure com creem aquest oracle, primer, veurem que succeeix quan neguem una fórmula com per exemple la de l'expressió 9.40.

$$\neg\phi = (x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \tag{9.42}$$

En l'expressió 9.42 veiem com la formula en CNF de l'expressió 9.40 es converteix en una fórmula en DNF (disjunctive normal form) a on cada clàusula és una conjunció de literals. També podem veure que si una assignació satisfà aquesta nova fórmula, clarament, no satisfà l'altre perquè per satisfer una clàusula necessitem la conjunció dels valors dels literals que fan falsa la clàusula original. Sabent això podem construir un circuit amb tants qubits auxiliars com clàusules i mitjançant portes Toffoli comprovar si estem satisfent cada clàusula de fórmula negada o no, ja que això ens donarà, automàticament, informació sobre la fórmula original. L'oracle que farem servir, amb la fórmula d'exemple de l'expressió 9.40 té el següent aspecte:

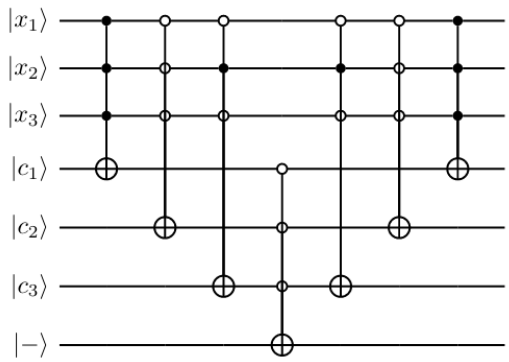


Figura 9.5: Oracle de l'algorisme de Grover per la fórmula ϕ (9.40)
Font: Elaboració pròpia

Com podem veure estem utilitzant els 3 qubits en superposició del pas 1 que representen les variables $\{x_1, x_2, x_3\}$, un qubit auxiliar per cada clàusula inicialitzats a l'estat $|0\rangle$ i finalment un qubit que es troba en l'estat $|-\rangle$ com s'explica en l'apartat 9.1.1. Per cada clàusula fem servir una porta Toffoli que negarà el valor del seu qubit auxiliar si l'assignació no satisfà la clàusula de la fórmula original. Per tal de poder aplicar la fase de -1 a les solucions, emprarem, de nou, una porta Toffoli que negarà l'estat $|-\rangle$ si tots els qubits auxiliars segueixen a 0, perquè voldrà dir que no s'ha satisfet cap clàusula de la fórmula negada i per conseqüència, s'han satisfet totes les de la fórmula original. Finalment, aplicarem totes les portes Toffoli a la inversa per revertir els possibles canvis en els qubits auxiliars i deixar-los a punt per si hem de realitzar més d'una iteració de l'operador de Grover.

Aquest oracle no és genèric i, per tant, canviarà depenen de la fórmula que estem tractant, tot i que, la manera en la qual es crea sempre és la mateixa i només caldrà aplicar les portes Toffoli amb els qubits de control pertinents.

Per acabar, s'han d'aplicar els passos 3, 4 i 5 de l'algorisme de Grover, amb els que, conjuntament amb l'oracle, ens serviran per crear l'operador de Grover G que fem servir també en el Quantum counting. Aquests passos es poden aplicar tal com s'explica en la secció 9.1.2, però per tal d'utilitzar

menys portes farem l'oposat del que es diu en el pas 4, és a dir, afegirem una fase de -1 a l'estat $|0\rangle^{\otimes n}$. Aquesta seria la nova transformació en notació de Dirac:

$$D_{new} = H^{\otimes n} D_{4\_new} H^{\otimes n} \tag{9.43}$$
$$= H^{\otimes n}(I - 2|0\rangle\langle 0|^{\otimes n})H^{\otimes n} \tag{9.44}$$
$$= I - 2|\psi\rangle\langle\psi|^{\otimes n} \tag{9.45}$$

Això es pot fer perquè si comparem les transformacions de les expressions 9.17 i 9.45, veiem que només es diferencien d'una fase global de -1 i, per tant, a efectes de probabilitat, en mesurar, són equivalents. A continuació podem trobar el circuit que en representaria els passos 3, 4 i 5, també coneguts com a inversió sobre la mitjana en la fórmula 3-CNF de l'expressió 9.40:

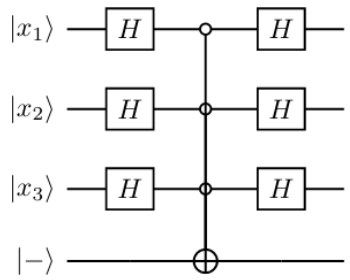


Figura 9.6: Inversió sobre la mitjana per la fórmula ϕ (9.40)
Font: Elaboració pròpia

Un cop sabem com crear l'oracle i la inversió sobre la mitjana de l'algorisme de Grover per 3-SAT, ja sabem com crear l'operador de Grover i només ens faltaria repetir-lo els cops que sigui necessari per maximitzar la probabilitat d'obtenir una solució. Si tornem a l'algorisme que es proposa per resoldre 3-SAT al principi d'aquesta secció 9.3, veiem que aquest operador es crearà en el primer pas, ja que el necessitem per poder executar el Quantum counting.

Per veure quina seria la complexitat del nostre algorisme podem analitzar els 4 passos que realitzem per separat. En el primer i tercer pas, per construir l'operador de Grover i per calcular el nombre de repeticions, dedicarem un temps polinòmic al nombre de variables, i en els algorismes de Grover i Quantum counting, com hem vist anteriorment, tenim una complexitat[1] de $O(\sqrt{N})$ en tots dos. Per tant, la complexitat final del nostre algorisme és $O(2poly(n) + 2\sqrt{N}) = O(\sqrt{N})$, a on veiem com aconseguim un speedup quadràtic en el nombre de crides a l'oracle respecte a l'algorisme 1 de força bruta que hem vist al principi del capítol.

Per entendre que la creació de l'oracle no és exponencial al nombre de variables, tot i que, necessitem crear dues portes Toffoli per cada clàusula i no sabem quantes clàusules tindrem, cal remarcar, que el nombre màxim de clàusules que podem tenir en una fórmula qualsevol de 3-SAT és polinòmic al nombre de variables. En la següent expressió podem veure el nombre de clàusules de 3 literals possibles amb un conjunt de $2n$ variables, ja que també tenim en compte les seves negacions:

$$\#Clàusules = \binom{2n}{3} = \frac{2n!}{3!(2n-3)!} = \frac{2n \cdot (2n-1) \cdot (2n-2)}{3!} = O(n^3) \tag{9.46}$$

[1] En aquest projecte quan es parla de complexitat en els algorismes quàntics esmentats, s'utilitza el model de Query Complexity [28], en el qual tenim en compte només el nombre de crides de l'oracle.

Capítol 10

Speedup quadràtic sobre algorismes clàssics

Com hem vist, amb l'algorisme de Grover aconseguim un speedup quadràtic sobre el nombre de crides a l'oracle que hauríem de realitzar amb l'algorisme clàssic de força bruta. En aquest capítol començarem veient com existeixen algorismes clàssics amb diferents enfocaments que el de força bruta i com aquests són capaços de millorar el que teníem fins ara. Seguidament, veurem com obtenir un speedup sobre un algorisme clàssic donat generalitzant l'algorisme de Grover i finalment es donarà un altre algorisme quàntic capaç de resoldre 3-SAT i amb millor temps d'execució esperat que l'algorisme de Grover.

10.1 Computació clàssica

A continuació, veurem que existeixen diferents tècniques i algorismes clàssics capaços de millorar en temps a l'algorisme aconseguit en el capítol anterior. Fins al moment hem vist un algorisme clàssic per resoldre 3-SAT mitjançant força bruta amb una complexitat[1] de $\tilde{O}(2^n)$ a on n és el nombre de variables de la fórmula. També hem vist l'aplicació de l'algorisme de Grover al problema per obtenir una solució fent $O(\sqrt{2^n})$ crides a l'oracle.

Clàssicament, podem trobar molts algorismes amb una millor complexitat que els dos anteriors com per exemple [3] [29] [30] [31] [32] [33] [34] [35], amb les seves respectives complexitats de $O(1.334^n)$, $O(1.3302^n)$, $O(1.32971^n)$, $O(1.3290^n)$, $O(1.32793^n)$, $O(1.3238^n)$, $O(1.32113^n)$ i $O(1.321^n)$. El primer algorisme de tots és l'algorisme proposat per Uwe Schöning i en el següent apartat veurem en què consisteix, ja que és un algorisme simple i fàcil d'entendre amb unes particularitats que ens seran d'utilitat més tard.

10.1.1 Algorisme de Schöning

L'algorisme de Schöning és un random walk que ens serveix per trobar una assignació que satisfaci una fórmula k-CNF amb una probabilitat de com a mínim $\left(\frac{1}{2}\left(1+\frac{1}{k-1}\right)\right)^n$. En el nostre cas, com estem intentant resoldre 3-SAT, tenim una probabilitat de succés major a $\left(\frac{3}{4}\right)^n$. Aquest algorisme com veurem a continuació és polinòmic i aquí podem veure en què consisteix:

[1] La notació d'O-gran amb una tilda ($\tilde{O}$) és equivalent a la notació convencional, O-gran, però ignorant factors logarítmics. És a dir, $f(n) \in \tilde{O}(h(n)) \iff \exists k : f(n) \in O(h(n)log^k(h(n)))$.

Algorithm 2 Algorisme de Schöning

Input: Fórmula f en k-CNF amb n variables
Escollir una assginació inicial $x \in \{0,1\}^n$ aleatòriament
loop $3n$ cops
  if x satisfà la fórmula **then**
    Retornar x
  end if
  Escollir una clàusula C entre les que no estan satisfetes per x
  Escollir un dels k literals de C aleatòriament i canviar el seu valor actual
end loop

Com podem veure al fragment de pseudocodi anterior, el primer pas consisteix a escollir una assignació qualsevol aleatòriament sobre la que realitzarem una sèrie de canvis per intentar arribar a una assignació que realment satisfaci la nostra fórmula. Seguidament, entrarem en un bucle $3n$ iteracions, a on primer comprovarem si l'assignació és certa, en aquest cas acabarem el bucle retornant l'assignació, o si l'assignació no satisfà la fórmula. En aquest segon cas, escollirem una clàusula que no estigui satisfeta aleatòriament i, dins d'aquesta, canviarem el valor d'un dels 3 literals també aleatòriament.

Per visualitzar millor l'algorisme podem representar el random walk amb un graf a on cada vèrtex equival a una assignació possible de les variables de la fórmula i a on tindrem una aresta per cada possible canvi de valor de les variables dins del bucle de $3n$ iteracions. L'algorisme de Schöning començarà en un vèrtex qualsevol i s'anirà movent pel graf fins a arribar al límit d'iteracions o fins que arribi a una assignació que satisfaci la fórmula. A continuació podem veure un graf d'exemple per la fórmula 9.40 a on les assignacions que fan certa la fórmula estan marcades amb un doble cercle:

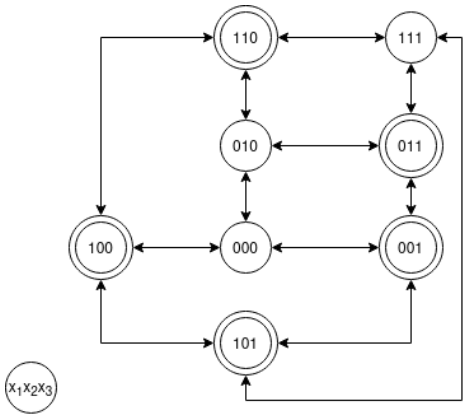


Figura 10.1: Schöning walk per la fórmula ϕ (9.40)
Font: Elaboració pròpia

Si analitzem els passos que realitzem i sabent que 3-SAT pertany a la classe NP, veiem que la complexitat de l'algorisme és polinòmica al nombre de variables de la fórmula. El factor exponencial que s'ha comentat abans ve donat de la seva probabilitat de succés. En ser un algorisme probabilístic amb una probabilitat p, per tal d'obtenir una solució amb alta probabilitat hauríem de repetir l'algorisme uns $\frac{1}{p}$ cops, deixatant-nos una complexitat total de $O(\left(\frac{4}{3}\right)^n \cdot poly(n)) = O(1.334^n \cdot poly(n)) = \tilde{O}(1.334^n)$. Si, com hem fet amb l'algorisme de força bruta, veiem el bucle de $3n$ iteracions com un oracle, podem crear un algorisme que amb $O(1.334^n)$ crides ens retorni una assignació amb alta probabilitat.

A continuació podem veure una taula resum amb el nombre de crides als seus respectius oracles que hauríem de realitzar amb l'algorisme de força bruta, l'algorisme de Grover i l'algorisme de Schöning.

Algorisme	**Tipus**	**Complexitat**
Força bruta	Clàssic	$O(2^n)$
Grover search	Quàntic	$O(1,4142^n)$
Schöning	Clàssic	$O(1,3334^n)$

Taula 10.1: Taula resum dels algorismes vistos i les seves complexitats.

10.2 Amplitude Amplification

L'anàlisi que s'ha fet de l'algorisme de Grover al capítol anterior es pot generalitzar fàcilment en el que anomenem Amplitude Amplification [36]. Si suposem que tenim un algorisme A (clàssic o quàntic) amb una probabilitat p de trobar una solució, quan aquest s'aplica a l'estat $|0\rangle^{\otimes n}$ obtindrem:

$$|U\rangle = A|0\rangle^{\otimes n} = (1-\sqrt{p})\,|\alpha\rangle + \sqrt{p}\,|\beta\rangle \tag{10.1}$$

L'estat $|U\rangle$, igual que ens passava en l'expressió 9.9 amb $|\psi\rangle$, ens queda en funció dels estats solució $|\alpha\rangle$ i no solució $|\beta\rangle$. Sabent això, podem comprovar que si apliquem el mateix esquema que l'algorisme de Grover, però partint de l'estat $|U\rangle$ i canviant la transformació Hadamard per la nova transformació A podem generalitzar l'algorisme. A continuació veiem els 5 passos de l'Amplitude Amplification:

1. Crear $|U\rangle = A|0\rangle^{\otimes n}$
2. Aplicar l'oracle O
3. Aplicar la transformació $A^\dagger$
4. Afegir una fase de -1 a tots els estats excepte a l'estat $|0\rangle^{\otimes n}$ (Difusor)
5. Aplicar la transformació A

En el primer pas creem l'estat $|U\rangle$ com veiem a l'expressió 10.1. Com que continua sent un estat normalitzat, el podem representar de la mateixa manera que l'estat $|\psi\rangle$ en la figura 9.3 i veure els següents passos com una rotació. En l'algorisme de Grover havíem agrupat els passos 2, 3, 4 i 5 en una porta anomenada G que efectuava dues reflexions, una sobre l'estat $|\beta\rangle$ i l'altre sobre l'estat $|\psi\rangle$. En aquesta generalització, per tal de fer la mateixa rotació, però sobre l'estat $|U\rangle$ necessitem fer la primera reflexió d'igual forma sobre l'estat $|\beta\rangle$ i la segona, en aquest cas, sobre el nou estat $|U\rangle$. Per tal de realitzar aquesta última reflexió, haurem de canviar els passos 3 i 5 per tal de crear l'estat $|U\rangle$, en comptes de l'estat $|\psi\rangle$ com fins ara. En la següent expressió podem veure com podem utilitzar les transformacions A i $A^\dagger$ juntament amb el pas 4 que anomenem D_4.

$$D = AD_4A^\dagger \tag{10.2}$$
$$= A(2|0\rangle\langle 0|^{\otimes n} - I)A^\dagger \tag{10.3}$$
$$= 2|U\rangle\langle U| - I \tag{10.4}$$

En el pas 3 utilitzem la transformació inversa i complexa conjugada per tal d'afectar el bra de l'expressió 10.3. Això també passava en l'algorisme de Grover i la transformació Hadamard, però com que $H = H^\dagger$ no es notava. Finalment, podem veure la porta G resultant de l'Amplitude Amplification:

$$G = DO = (2|U\rangle\langle U| - I)\,(I - 2|\beta\rangle\langle\beta|) \tag{10.5}$$

Com veiem, els passos anteriors generalitzen l'algorisme de Grover a on $A = H^{\otimes n}$ i $p = \frac{M}{N}$ sent M és el nombre de solucions del problema. Utilitzant aquest mètode podem aconseguir un speedup quadràtic de qualsevol algorisme clàssic que ens permeti preparar l'estat A. Clàssicament, necessitaríem realitzar $\frac{1}{p}$ repeticions de l'algorisme A per tal de trobar una solució amb alta probabilitat, mentre que quànticament podem comprovar de forma anàloga al capítol anterior com en tenim prou amb $\frac{1}{\sqrt{p}}$ crides a l'oracle. Per comprovar-ho n'hi hauria suficient en substituir l'amplitut $\sqrt{\frac{M}{N}}$ per $\sqrt{p}$ a l'expressió 9.20 que ens portarà a definir $\theta = 2 \cdot arcsin(\sqrt{p})$ i continuar amb els mateixos càlculs.

10.3 Descripció de l'algorisme

A continuació veurem un segon algorisme quàntic que utilitzant l'Amplitude Amplification i l'algorisme clàssic de Schöning aconsegueix un millor temps d'execució esperat que el primer algorisme presentat en el capítol anterior, en l'apartat 9.3. Concretament, aconseguim un speedup quadràtic de l'algorisme clàssic de Schöning.

Com s'ha comentat en l'apartat 10.1.1, l'algorisme de Schöning és un random walk amb una probabilitat $p \geq \left(\frac{3}{4}\right)^n$ de trobar una solució. Equivalentment, si ens fixem en la figura 10.1, cada conjunt de $3n$ passos partint des d'una assignació qualsevol té una probabilitat de trobar una solució major a $\left(\frac{3}{4}\right)^n$, ja que equival a una execució de l'algorisme de Schöning. Sabent això, podem crear un circuit quàntic que ens generi un estat en superposició per representar tots els possibles camins de longitud $3n$ partint des de tots els possibles vèrtexs. Aquest estat en superposició, com hem vist en els algorismes anteriors, el podem dividir en els estats que no són solució i els estats que són solució de la següent manera:

$$|U\rangle = \sqrt{\left(\frac{1}{4}\right)^n}\,|\alpha\rangle + \sqrt{\left(\frac{3}{4}\right)^n}\,|\beta\rangle \tag{10.6}$$

Com veiem, l'expressió que aconseguim és exactament el que hem definit a l'expressió 10.1 i, per tant, aquest nou circuit farà la funció de l'algorisme A en l'Amplitude Amplification. Per tal de poder aplicar l'algorisme sencer i obtenir un speedup quadràtic de l'algorisme de Schöning, haurem de dissenyar un oracle capaç de decidir quan un vèrtex inicial i un camí de $3n$ passos arriben a una solució de la nostra fórmula, és a dir, simular el bucle de $3n$ iteracions però amb les eleccions aleatòries predefinides.

A continuació podem veure l'algorisme que es proposa en aquest apartat i que s'explicarà a continuació:

1. Crear el circuit A en funció de les variables d'una fórmula donada.
2. Crear l'oracle O específic per la fórmula donada.
3. Utilitzar Amplitude Amplification amb el circuit A i l'oracle O per trobar un vèrtex inicial i un camí que portin a una solució amb alta probabilitat.
4. Utilitzar l'algorisme clàssic de Schöning amb el vèrtex inicial i el camí resultant de l'Amplitude Amplification per trobar una assignació que, amb alta probabilitat, satisfaci la fórmula donada.

Crear el circuit A

El circuit que anomenem A ens servirà per inicialitzar el nostre estat quàntic. Per tal d'arribar a un estat com el de l'expressió 10.6 podem veure el bucle de l'algorisme de Schöning com un algorisme determinista amb un input addicional $r \in \{0,1\}^n \times \{0,1,2\}^{3n}$ a on definim l'assignació inicial de n variables amb les posicions dels $3n$ canvis dels valors de les variables que farem i crear un estat en

superposició de tots els possibles inpunts r per realitzar una cerca en aquest espai.

Per crear aquest estat és fàcil veure que els primers n qubits, que representen les variables de la fórmula, han d'estar en superposició. Per aconseguir això n'hi ha prou amb utilitzar la porta Hadamard en els primers n qubits com també fèiem a l'algorisme de Grover al capítol anterior.

Per representar la tria d'un literal en cada un dels 3n passos de l'algorisme de Schöning podem crear 3n cops l'estat $\frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$. Aquest estat està en superposició i col·lapsarà amb igual probabilitat en l'estat $|00\rangle$, $|01\rangle$ o $|10\rangle$. Podem utilitzar la representació decimal d'aquests estats per a decidir quin literal dins d'una clàusula es modificarà.

Per tal de construir l'estat $\frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$ farem servir 2 qubits inicialitzats a $|00\rangle$, ja que amb 1 de sol no en tindríem prou per representar tres valors. Utilitzarem la porta quàntica Ry(θ) que ens representa una rotació de $\frac{\theta}{2}$ graus a l'eix Y i una porta Hadamard amb control com podem veurem a continuació.

1.	$\|00\rangle$	***Estat inicial***
2.	$\left[\sqrt{\frac{2}{3}}\|0\rangle + \sqrt{\frac{1}{3}}\|1\rangle\right]\|0\rangle$	*Aplicar Ry(θ) sobre el primer qubit.*
3.	$\sqrt{\frac{2}{3}}\|00\rangle + \sqrt{\frac{1}{3}}\|10\rangle$	*Reorganitzar termes.*
4.	$\sqrt{\frac{2}{3}}\|0\rangle\left[\sqrt{\frac{1}{2}}\|0\rangle + \sqrt{\frac{1}{2}}\|1\rangle\right] + \sqrt{\frac{1}{3}}\|10\rangle$	*Hadamard al segon qubit si el primer qubit és 0.*
5.	$\sqrt{\frac{1}{3}}\|00\rangle + \sqrt{\frac{1}{3}}\|01\rangle + \sqrt{\frac{1}{3}}\|10\rangle$	*Reorganitzar termes.*
6.	$\sqrt{\frac{1}{3}}(\|00\rangle + \|01\rangle + \|10\rangle)$	*Treure factor comú* $\sqrt{\frac{1}{3}}$.

L'angle θ que necessitem per crear el nostre estat és $\theta = 2 \cdot arccos \cdot (\sqrt{\frac{2}{3}})$. Per entendre d'on surt l'angle que estem fent servir cal recordar que la porta Ry equival a la següent matriu:

$$\begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{10.7}$$

Crear l'oracle O

El següent pas consisteix a crear un oracle que, donada una assignació i un conjunt de $3n$ posicions de les variables que canviarem el seu valor dins d'una clàusula, sigui capaç de decidir si arribem a una assignació que satisfà la nostra fórmula o no. Per fer això haurem de simular el bucle de $3n$ iteracions de l'algorisme de Schöning quànticament. El circuit resultant serà molt gran si tenim moltes variables, per culpa del nombre d'iteracions a realitzar, però per poder veure en què consisteix el circuit podem utilitzar una fórmula petita amb dues clàusules i quatre literals com la següent:

$$\gamma = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4) \tag{10.8}$$

A continuació podem trobar una representació de l'oracle a on també veiem la inicialització de l'estat quàntic esmentat en el pas anterior. En el circuit apareix només la simulació de la primera iteració de les $3n$ que realitzarà l'oracle amb la fórmula γ per tal explicar el seu disseny:

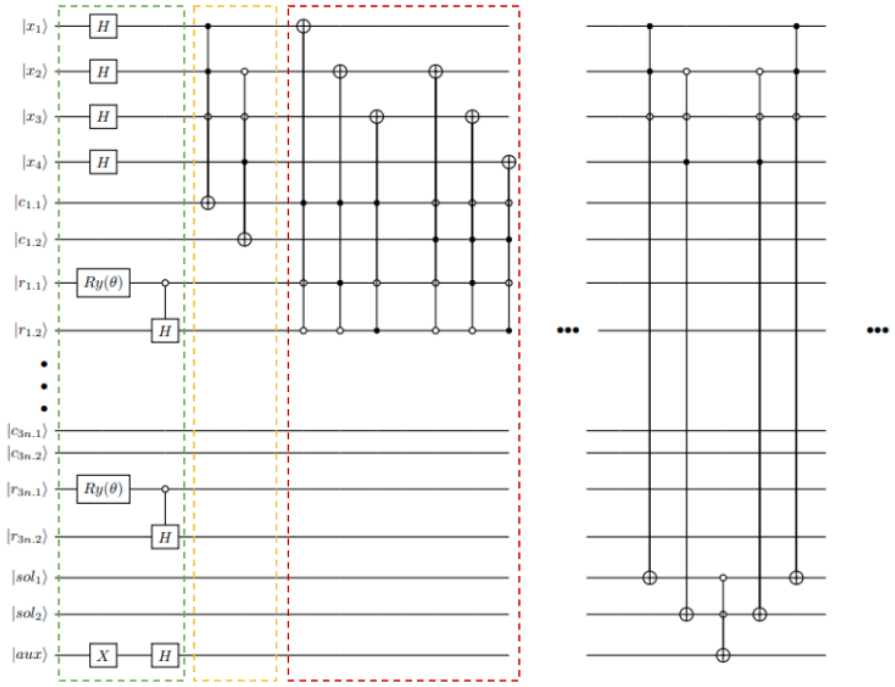


Figura 10.2: Inicialització i oracle del segon algorisme quàntic per la fórmula γ (10.8)
Font: Elaboració pròpia

Com podem veure en el circuit anterior tenim 3 fases marcades en diferents colors:

- En verd trobem la inicialització dels qubits que hem realitzat en el pas anterior.
- En taronja tenim la comprovació de les clàusules que no estan satisfetes en la primera iteració.
- En vermell podem trobar el canvi de valor d'un dels 3 literals de la primera clàusula no satisfeta en la primera iteració.

Com podem veure, en el pas taronja, per determinar quines clàusules estan satisfetes i quines no, fem el mateix que en l'algorisme de Grover per 3-SAT del capítol anterior, utilitzar una porta Toffoli amb tants qubits auxiliars inicialitzats a $|0\rangle$ com clàusules a la fórmula a on els qubits de control són els literals de les clàusules negats. En negar al seu valor sabem que si el qubit auxiliar canvia a $|1\rangle$ vol dir que la clàusula no estava satisfeta, ja que els qubits de control són una conjunció dels únics valors que fan falsos cada literal. En cada iteració de la simulació de l'algorisme de Schöning haurem d'utilitzar qubits auxiliars diferents per poder recordar quines clàusules no estaven satisfetes a cada pas.

Per altra banda, en el pas vermell, podem veure com estem fent servir, també, portes Toffoli per negar un dels literals de la primera clàusula no satisfeta. En aquest pas, tindrem tantes portes Toffoli com nombre de clàusules multiplicat per 3, ja que a cada literal li correspon una a cada aparició que tingui a la fórmula. Si ens fixem en una única clàusula, veurem que sempre tenim tres portes Toffoli, una per cada variable. Els qubits de control són escollits per tal d'assegurar-nos que l'estat $\frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$ d'aquella iteració col·lapsarà en la posició del literal que afecta la porta Toffoli. També ens assegurem que la clàusula a la qual pertany el literal no estigui satisfeta i que totes les clàusules anteriors ho hagin estat perquè volem actuar només en una clàusula.

Demostració

Per veure que l'oracle realment comprova que una assignació inicial amb un conjunt de $3n$ passos arriba a una assignació que satisfà la fórmula o no tal com es fa en l'algorisme de Schöning, podem analitzar si estem realitzant els mateixos passos a cada iteració. En l'algorisme clàssic, a cada iteració poden passar dues coses:

1. L'assignació satisfà la fórmula i no se li fa cap canvi
2. L'assignació no satisfà la fórmula i se li canvia el valor d'un únic literal d'una de les clàusules no satisfetes

En el nostre oracle, a cada iteració comencem comprovant quines clàusules estan satisfetes i quines no en el pas taronja utilitzant qubits auxiliars. Per tant, la informació de la qual disposem abans del pas vermell és un conjunt de qubits $c_{iter,claus}$ que tindran valor 1 si la seva clàusula no està satisfeta, dos qubits $r_{iter,1}, r_{iter,2}$ que col·lapsarà en un dels valors $\{0,1,2\}$ i que ens defineixen la posició del literal que canviarem el seu valor dins d'una clàusula no satisfeta, i finalment, com tenim accés a la fórmula, sabem quines són les variables de cada clàusula $\{l_{claus,1}, l_{claus,2}, l_{claus,3}\}$. Sabent això i que cada porta Toffoli actual com una porta AND amb tots els seus qubits de control, podem definir les següents restriccions per canviar el valor de cada un dels literals de la següent manera:

$$\left.\begin{array}{l}(r_{1.1}=0 \wedge r_{1.2}=0 \wedge c_{1.1}=1) \rightarrow l_{1.1}\\(r_{1.1}=1 \wedge r_{1.2}=0 \wedge c_{1.1}=1) \rightarrow l_{1.2}\\(r_{1.1}=0 \wedge r_{1.2}=1 \wedge c_{1.1}=1) \rightarrow l_{1.3}\end{array}\right\} \textit{Primera clàusula} \tag{10.9}$$

$$\left.\begin{array}{l}(r_{1.1}=0 \wedge r_{1.2}=0 \wedge c_{1.1}=0 \wedge c_{1.2}=1) \rightarrow l_{2.1}\\(r_{1.1}=1 \wedge r_{1.2}=0 \wedge c_{1.1}=0 \wedge c_{1.2}=1) \rightarrow l_{2.2}\\(r_{1.1}=0 \wedge r_{1.2}=1 \wedge c_{1.1}=0 \wedge c_{1.2}=1) \rightarrow l_{2.3}\end{array}\right\} \textit{Segona clàusula} \tag{10.10}$$

$$\vdots$$

$$\left.\begin{array}{l}(r_{1.1}=0 \wedge r_{1.2}=0 \wedge (\sum_{i=0}^{n-1} c_{1.i})=0 \wedge c_{1.n}=1) \rightarrow l_{n.1}\\(r_{1.1}=1 \wedge r_{1.2}=0 \wedge (\sum_{i=0}^{n-1} c_{1.i})=0 \wedge c_{1.n}=1) \rightarrow l_{n.2}\\(r_{1.1}=0 \wedge r_{1.2}=1 \wedge (\sum_{i=0}^{n-1} c_{1.i})=0 \wedge c_{1.n}=1) \rightarrow l_{n.3}\end{array}\right\} \textit{Clàusula n-ària} \tag{10.11}$$

Com veiem, en l'expressió general en una clàusula n-ària, una porta Toffoli en concret sempre té en compte que tots els qubits auxiliars de les clàusules anteriors a la que intentarà modificar hagin estat satisfetes per l'assignació actual. Amb aquesta comprovació ens assegurem que només es puguin activar tres portes Toffoli, concretament, les que intentaran modificar una variable d'aquella clàusula. Per assegurar-nos que, efectivament, només s'activa una de les tres, s'utilitzen els qubits $r_{iter,1}, r_{iter,2}$ que col·lapsaran només en una posició en concret.

Per tant, podem veure com realment només modificarem el valor d'un únic literal d'una de les clàusules no satisfetes, en concret, la primera que trobem. Per altra banda, també és fàcil veure que si totes les clàusules ja estan satisfetes, cap tindrà el seu qubit auxiliar a 1 i cap porta Toffoli realitzarà cap canvi, tal com l'algorisme clàssic feia.

Per crear aquest oracle que simuli l'algorisme de Schöning necessitem un total de $24nc+2c+1$ portes Toffoli, a on c és el nombre de clàusules i n és el nombre de variables.

Els passos taronja i vermell es repeteixen $3n$ cops fent servir els qubits auxiliars destinats per aquella iteració. Un cop s'han simulat les 3n iteracions de l'algorisme de Schöning cal tornar a comprovar si l'assignació resultant satisfà la fórmula o no i en cas afirmatiu negar l'últim qubit per tal d'aplicar un canvi de fase de -1 com fèiem també en Grover. Aquest últim pas es pot veure al final de la figura 10.2 fora de les 3 fases definides a dalt.

Finalment, haurem d'aplicar tots els $3n$ passos taronja i vermell a la inversa per tal de revertir els canvis realitzats en ells qubits auxiliars per deixar-los a $|0\rangle$ per les properes iteracions de l'Amplitude Amplification com també fèiem en l'algorisme de Grover.

En la següent expressió podem trobar el nombre total de qubits que necessitarem pel circuit a on c és el nombre de clàusules i n és el nombre de variables de la fórmula:

$$\#qubits = n + 3n \cdot (2 + c) + c + 1 = O(n^4) \tag{10.12}$$

Tenint en compte que com hem vist al capítol anterior el nombre de total de clàusules que podem tenir amb n variables es pot acotar en $O(n^3)$, el nombre total de qubits que necessitem en pitjor cas és de l'ordre de $O(n^4)$.

Aplicació de l'Amplitude Amplification

Un cop tenim el circuit A i l'oracle O, ja podem aplicar l'amplitude Amplification per obtenir una solució amb alta probabilitat. Al final del circuit mesurarem els qubits que representen les variables de la fórmula per aconseguir una assignació inicial i també tots els qubits $r_{iter,1}, r_{iter,2}$ de cada iteració per tal de saber els canvis que haurem de realitzar a l'assignació inicial per arribar a una solució. El resultat obtingut el podem veure com una cadena de bits clàssics a on el de menys pes equival al valor de la primera variable en l'assignació i el de més pes equival a la posició del literal que canviarem el seu valor en l'última de les $3n$ iteracions.

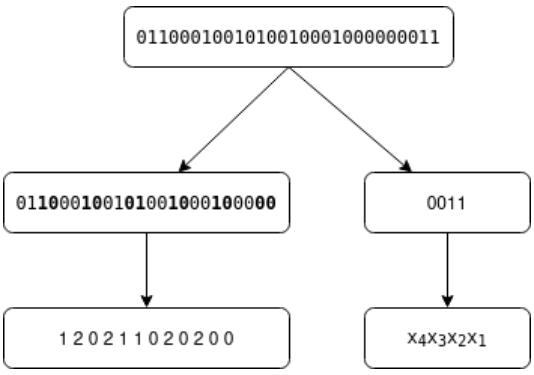


Figura 10.3: Resultat després de mesurar els qubits en l'Ampltiude Amplification amb la fórmula γ (10.8)

Font: Elaboració pròpia

Els passos 2, 3, 4 i 5 de l'Amplitude Amplification també s'han de repetir k cops com feiem amb l'algorisme de Grover. El nombre de repeticions ens portarà a la complexitat de l'algorisme i seguint les mateixes formules que en el capitol anterior i com s'ha explicat en l'apartat 10.2 arribem a la següent expressió:

$$k = \frac{\pi}{4} \cdot \frac{1}{arcsin\left(\sqrt{p}\right)} - \frac{1}{2} \leq \frac{\pi}{4} \cdot \frac{1}{\sqrt{p}} - \frac{1}{2} \leq \left\lceil \frac{\pi}{4} \cdot \frac{1}{\sqrt{p}} \right\rceil \tag{10.13}$$

Com sabem que $p = \left(\frac{3}{4}\right)^n$ podem concloure que, en aquest cas, la complexitat de l'Amplitude Amplification és $O(\sqrt{\left(\frac{4}{3}\right)^n})$. La probabilitat p, en el paper original [3], es calcula en el pitjor cas, és a dir, suposant que només tenim una assignació que satisfà la fórmula. En general tindrem més d'una solució i, per tant, el nombre total de repeticions dels passos 2, 3, 4 i 5 serà menor. Per calcular la seva complexitat no ens afecta, ja que ens interessa el worst-case scenario, però quan portem l'algorisme a la pràctica pot ser un problema.

A la pràctica si repetim els passos més de k cops, estem disminuint la probabilitat d'aconseguir una solució i, per tant, haurem de saber quantes solucions tenim per calcular k d'una forma més realista. Per fer això podem tornar a utilitzar el Quantum counting o podem suposar que no sabem la probabilitat d'èxit del circuit A i aplicar l'algorisme QSearch introduït per G. Brassard, P. Høyer, M. Mosca i A. Tapp en el paper Quantum Amplitude Amplification and Estimation [37] que ens assegura que podem trobar una solució en $O(\frac{1}{\sqrt{a}})$ a on a és la probabilitat d'èxit real de l'algorisme A.

Aplicació de l'algorisme clàssic de Schöning

Finalment, aplicarem un cop l'algorisme clàssic de Schöning utilitzant el resultat obtingut amb l'Amplitude Amplification del pas anterior. Aquest últim pas és necessari, ja que com hem vist, la cadena de bits que tenim ens defineix l'assignació inicial i els passos a seguir per l'algorisme clàssic.

En aquest pas modificarem l'algorisme clàssic de Schöning per substituir les eleccions aleatòries per consultes a la cadena de bits. Com amb l'algorisme quàntic hem seleccionat una cadena de bits que ens porta a una solució amb alta probabilitat només ens caldrà executar l'algorisme clàssic un cop. Per tant, aquest pas té una complexitat polinòmica al nombre de variables de la fórmula com hem vist a l'apartat 10.1.1.

Per calcular la complexitat total del segon algorisme proposat per resoldre 3-SAT podem anar pas a pas. Els passos 1 i 2 tenen una complexitat polinòmica, ja que el nombre de portes que utilitzarem és polinòmic al nombre de variables, l'Amplitude Amplification té una complexitat exponencial de $O(\sqrt{\left(\frac{4}{3}\right)^n})$ en pitjor cas i l'algorisme de Schöning clàssic, igual que els passos 1 i 2, és polinòmic en n. Per tant, podem concloure que el nostre algorisme té una complexitat total de $O(3poly(n) + \sqrt{\left(\frac{4}{3}\right)^n}) = O(\sqrt{\left(\frac{4}{3}\right)^n}) = O(1,1547^n)$.

A continuació podem veure una altra vegada la taula resum de tots els algorismes que s'ha parlat en el treball amb les seves respectives complexitats. Com veiem, aquest últim algorisme és el que menys crides a l'oracle ha de realitzar.

Algorisme	**Tipus**	**Complexitat**
Força bruta	Clàssic	$O(2^n)$
Grover search	Quàntic	$O(1,4142^n)$
Schöning	Clàssic	$O(1,3334^n)$
Amplitude Amplification/Schöning	Quàntic/Clàssic	$O(1,1547^n)$

Taula 10.2: Taula resum de tots els algorismes vistos i les seves complexitats.

Capítol 11

Implementació i resultats

En aquest capítol veurem com s'ha realitzat la implementació dels dos algorismes proposats en els capítols anteriors i quins han estat els resultats després d'una simulació utilitzant els simuladors disponibles amb Qiskit.

Podem trobar el codi dels algorismes en el següent enllaç de GitHub: `https://github.com/arnaucasau/Quantum-Computing-3SAT`

11.1 Qiskit

Per implementar els algorismes s'ha utilitzat Qiskit, un programari desenvolupat per IBM que ens permet treballar amb ordinadors quàntics fent ús del llenguatge de programació Python. Qiskit ofereix una gran varietat de simuladors [38], els quals podem fer servir per executar i provar els algorismes. També tenim accés a ordinadors quàntics reals, però, en aquest cas, a causa de la quantitat de qubits que es necessitaven per poder experimentar amb fórmules interessants, s'ha decidit no fer ús d'ells.

En el treball s'han utilitzat dos simuladors diferents. En el primer algorisme veurem com el simulador per defecte, *Statevector*, ja ens serveix per poder simular el circuit, ja que aquest ens ofereix la possibilitat de simular fins a 32 qubits i té una gran varietat de portes quàntiques que estan suportades, incloent-hi totes les que necessitem. Aquest simulador computa la funció d'ona del vector d'estat dels qubits a mesura que les portes i les instruccions són aplicades.

Per altra banda, en el segon algorisme, per poder simular un circuit que ens trobi una assignació que satisfaci una fórmula amb només una solució i que alhora estigui alineada amb el càlcul de la probabilitat de l'algorisme de Schöning en el paper original, s'ha escollit un altre simulador anomenat *Matrix Product State* [39]. Aquest simulador ens permet simular fins a 100 qubits i consisteix en una xarxa tensor que utilitza una representació de producte de matrius pels estats, la qual és molt eficient per estats amb poc entrellaçament com en el nostre cas. El simulador suporta menys portes quàntiques que l'anterior però les suficients per dur a terme l'algorisme.

L'única porta no suportada pel segon simulador és la porta $Ry(\theta)$ que ens fa falta per inicialitzar els qubits, però s'ha utilitzat la porta $U(\theta, \phi, \lambda)$ amb els paràmetres $\phi = 0$ i $\lambda = 0$ la qual és equivalent, com hem vist al capítol 8.

$$U(\theta, \phi, \lambda) = \begin{pmatrix} cos(\frac{\theta}{2}) & -e^{i\lambda} sin(\frac{\theta}{2}) \\ e^{i\phi} sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)} cos(\frac{\theta}{2}) \end{pmatrix} \tag{11.1}$$

11.2 Primer algorisme proposat

En el primer algorisme utilitzarem l'algorisme de Grover per trobar una solució a una fórmula 3-SAT donada a on el codi de la implementació el podem trobar en el notebook anomenat *Part-1-3SAT-Quantum-Computing.ipynb*.

El primer pas de l'algorisme que hem definit a la secció 9.3 consisteix a crear l'operador de Grover específic per 3-SAT i en concret per la fórmula a la qual li volem trobar una solució. Per introduir aquesta fórmula, el notebook, la llegirà d'un arxiu .txt permetent-nos canviar-la quan vulguem. A continuació podem veure l'arxiu amb la fórmula d'exemple que s'ha utilitzat per executar l'algorisme:

Formula 3-SAT d'exemple

```
5 3 # nombre de variables i nombre de clausules
-1 -2 -3 # primera clausula
1 2 3 # segona clausula
1 -2 3 # tercera clausula
```

Com veiem en l'arxiu .txt, primer hem d'especificar en la primera línia el nombre de variables i el nombre de clàusules que utilitzarem. Seguidament, hem d'escriure una clàusula per línia a on cada clàusula tindrà tres literals i cada literal estarà definit per nombres de l'1 al nombre de variables. Farem servir un signe '-' davant d'un literal per indicar que volem la seva versió negada. L'arxiu anterior està codificant la fórmula:

$$\phi = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \tag{11.2}$$

La fórmula anterior només utilitza tres variables tot i que hem especificat que n'hi ha cinc. Si això succeeix vol dir que tindrem dues variables, en aquest cas, x_4 i x_5 que podran prendre qualsevol valor. Un cop sabem la fórmula ja podem crear l'oracle per tal d'utilitzar-lo en el quantum counting i en l'algorisme de Grover com hem vist al capítol 9. A l'hora de crear l'oracle, com s'ha comentat a l'apartat 9.1.3, hem de fer servir un qubit extra per crear un espai ampliat i així assegurar-nos de tenir com a màxim la meitat d'estats com a no solució. Si fem això, podem imaginar que la fórmula anterior té una variable nova x_6 que forçarem que sigui False.

$$\phi_{espai\_ampliat} = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_6) \tag{11.3}$$

El següent pas de l'algorisme, un cop ja tenim creat l'oracle, és executar el quantum counting per tal d'obtenir l'angle θ que ens servirà per calcular el nombre d'iteracions que haurem de realitzar de l'operador de Grover. A la figura 11.1 podem veure el resultat aconseguit després d'executat l'algorisme 1024 cops amb la fórmula 11.3 que podem trobar en el notebook. S'ha establert $t = nombre\_variables$, ja que en les simulacions que podem fer, a causa del nombre de qubits disponible pel simulador, tenim un nombre fitat de variables i no acabarà afectant la complexitat total de l'algorisme.

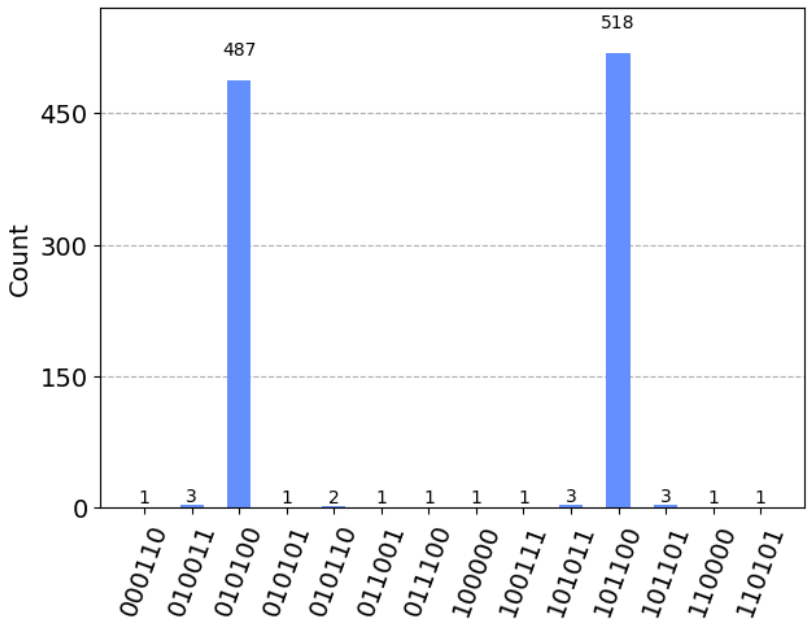


Figura 11.1: Mesures de l'algorisme Quantum Counting
Font: Elaboració pròpia

Com sabem, després d'executar el quantum counting podíem rebre dos resultats a causa que la matriu de rotació de l'operador de Grover té dos valors propis diferents. En aquest cas veiem que els dos valors en binari més probables són 010100 que en decimal és 20 i 101100 que en decimal és 44. Com hem vist a l'apartat 9.2, un valor equival al valor propi amb l'angle θ i l'altre equival al valor propi amb l'angle $2\pi - \theta$. Podem calcular els dos angles aïllant θ de l'expressió 9.37.

$$\theta = \frac{2\pi x}{2^t} = \frac{2\pi \cdot 20}{2^6} = 1.96349 \tag{11.4}$$

$$2\pi - \theta = \frac{2\pi x}{2^t} = \frac{2\pi \cdot 44}{2^6} = 4.31968 \tag{11.5}$$

Un cop tenim els angles, fent ús de l'expressió 9.38 i sabent que amb sis variables tenim $N = 64$ assignacions diferents, podem calcular el nombre total de solucions que tindrem. En aquest cas, com a l'hora de crear l'operador de Grover, per simplificar el nombre de portes necessàries, hem fet l'oposat del pas 4 de l'algorisme de Grover com hem vist a la secció 9.3, ara rebrem el nombre de no solucions:

$$N - M = sin^2\left(\frac{1.96349}{2}\right) \cdot 64 = sin^2\left(\frac{4.31968}{2}\right) \cdot 64 = 44.24616... \tag{11.6}$$

Sabent el nombre de no solucions podem calcular fàcilment que el nombre total aproximat de solucions que tenim són $N - (N - M) = 19.75384 = M$. A continuació hem de calcular el nombre d'iteracions k de l'operador de Grover que haurem de realitzar i per fer això, sabent el nombre de solucions que tenim podem recórrer a la fórmula de l'expressió 9.25:

$$k \leq \left\lceil \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rceil = \left\lceil \frac{\pi}{4} \cdot \sqrt{\frac{64}{19.8}} \right\rceil = \lceil 1.4136793455054264 \rceil = 2 \tag{11.7}$$

Com el nombre d'iteracions ha de ser menor que 2, però és més proper a 1, en aquest notebook s'ha decidit fer només 1 iteració de l'algorisme. A continuació podem veure el resultat obtingut després d'executar 1024 cops l'algorisme de Grover amb el nombre d'iteracions calculat anteriorment:

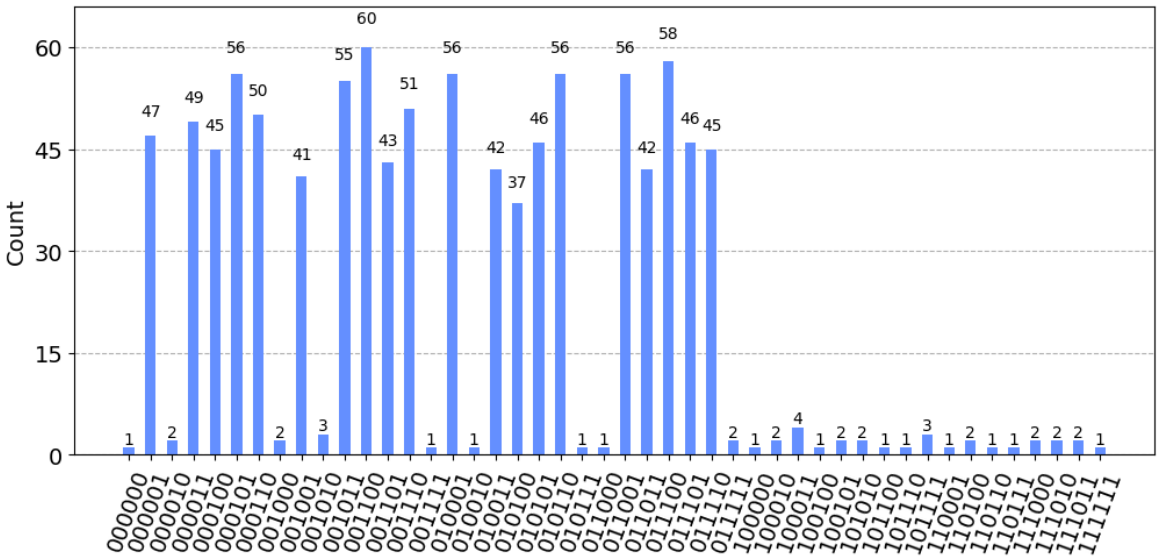


Figura 11.2: Mesures de l'algorisme de Grover
Font: Elaboració pròpia

Com podem veure, hi ha 20 assignacions que semblen ser solucions molt probablement. Això és bon senyal, ja que en el quantum counting hem vist que aproximadament hi havia 19.8 solucions. Si ens fixem més detingudament en la fórmula 11.3, veurem que les solucions que té es corresponen amb les obtingudes amb l'algorisme de Grover tenint en compte que el bit de menys pes de cada string binari es correspon al valor de la variable x_1 i el de més pes equival a la variable x_6 que havíem forçat que fos 0.

Per acabar, podem veure també en la figura 11.2 com hem obtingut una solució 981 vegades de les 1024 execucions de l'algorisme de Grover deixant-nos una probabilitat de 0.958 de trobar una assignació correcta sabent aproximadament el nombre de solucions de la fórmula. Aquesta probabilitat calculada empíricament, pot canviar molt en funció del nombre de solucions que calculem amb el resultat del Quantum counting, ja que llavors realitzaríem més o menys iteracions del compte i no incrementaríem tant la probabilitat d'aconseguir un estat solució. Si comparem aquesta probabilitat amb la que obtindríem amb l'expressió 9.23 i que teòricament hauríem de tenir amb l'angle θ obtingut i realitzant $k = 1$ iteracions, podem veure que és molt semblant a l'aconseguida empíricament:

$$sin^2\left((2k+1)\cdot\frac{\theta}{2}\right) = sin^2\left(3\cdot\frac{1.96349}{2}\right) = 0.933 \tag{11.8}$$

11.3 Segon algorisme proposat

En el segon algorisme utilitzem l'Amplitude Amplification per tal d'aconseguir un speedup quadràtic sobre l'algorisme clàssic de Schöning. En aquest cas podem trobar el codi de la implementació en el notebook anomenat *Part-2-3SAT-Grover-Schoning.ipynb*.

Els dos primers passos de l'algorisme que podem trobar a la secció 10.3 consisteixen a crear el circuit que hem anomenat A i l'oracle O per tal d'inicialitzar els nostres qubits i poder decidir quines assignacions inicials poden arribar a una solució executant l'algorisme clàssic de Schöning. Aquests dos circuits depenen de la fórmula que volem resoldre i, igual que en l'algorisme anterior, la llegirem

d'un arxiu de text. Aquí podem veure l'arxiu .txt amb la fórmula d'exemple que s'ha utilitzat en aquest cas:

Formula 3-SAT d'exemple

```
3 7 # nombre de variables i nombre de clausules
1 2 3 # primera clausula
1 2 -3 # segona clausula
1 -2 3 # tercera clausula
1 -2 -3 # quarta clausula
-1 2 3 # cinquena clausula
-1 2 -3 # sisena clausula
-1 -2 3 # setena clausula
```

Com s'ha comentat anteriorment, en el paper original de l'algorisme de Schöning [3], es calculava la probabilitat d'èxit de l'algorisme en pitjor cas, és a dir quan només tenim una assignació que satisfà la fórmula. Com veiem en l'arxiu .txt, s'ha reproduït aquest pitjor cas amb una fórmula a on només tenim l'assignació $\langle x_1 = 1, x_2 = 1, x_3 = 1\rangle$ com a solució. Aquesta és la fórmula que estem codificant en l'arxiu anterior:

$$\begin{aligned}\phi =&(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \\ &(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)\end{aligned} \tag{11.9}$$

Un com sabem la fórmula ja podem crear els dos circuits tal com s'ha explicat en el capítol anterior i com podem veure a l'inici del notebook. Seguidament, aplicarem l'Ampltiude Amplification amb el circuit A i l'oracle O per tal d'aconseguir, amb alta probabilitat, una assignació inicial i un conjunt de $3n$ posicions de l'1 al 3 per tal de decidir quines variables seran afectades a cada iteració del random walk. Aquest ha estat el resultat que podem trobar després d'una execució de l'algorisme:

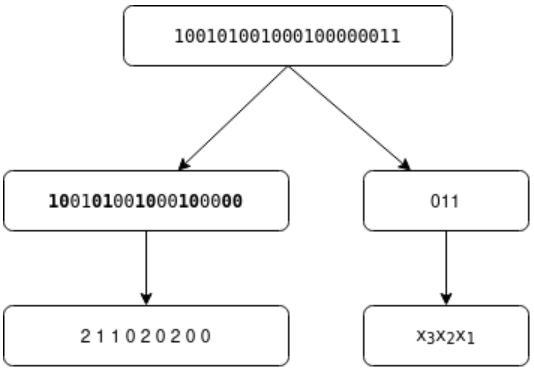


Figura 11.3: Resultat de l'Ampltiude Amplification amb la fórmula ϕ (11.8)
Font: Elaboració pròpia

Tenint en compte que el bit de menys pes equival al primer qubit del circuit, podem veure com l'Amplitude Amplification ens ha retornat que una assignació inicial que ens podria portar a una solució en executar l'algorisme de Schöning pot ser $\langle x_1 = 1, x_2 = 1, x_3 = 0\rangle$. Per tal d'arribar a una solució haurem de canviar els valors de les següents posicions de la primera clàusula no satisfeta a cada iteració de l'algorisme clàssic $\langle iter1 = 0, iter2 = 0, iter3 = 2, iter4 = 0, iter5 = 2, iter6 = 0, iter7 = 1, iter8 = 1, iter9 = 2\rangle$. Cal remarcar que aquestes posicions equivalen a les tries aleatòries que realitzem en l'algorisme clàssic, però pel fet que es calculen totes abans de fer cap comprovació pot donar-se

el cas, com veurem en aquest exemple, que arribem a una solució abans de realitzar totes les iteracions.

A l'hora d'implementar l'Amplitude Amplification s'ha hagut de modificar el pas 4 de l'algorisme que trobem a l'apartat 10.2 per tal d'agilitzar la simulació. Com podem trobar al notebook, l'única porta Toffoli necessària en aquest pas s'ha dividit en diverses portes Toffoli amb pocs qubits de control utilitzant qubits auxiliars.

L'últim pas del nostre algorisme consisteix a executar l'algorisme de Schöning clàssicament amb les tries aleatòries predefinides amb els valors que hem obtingut de l'Amplitude Amplification. Si fem això i partim de l'assignació esmentada anteriorment podem veure com arribem a una solució amb només tres iteracions del random walk:

1. $x_1 = 1,\ x_2 = 1,\ x_3 = 0$ ***Assignació inicial - Resultat Amplitude Amplification***
2. $x_1 = 0,\ x_2 = 1,\ x_3 = 0$ ***1a iteració - flip posició 0 1a clausula no satisfeta***
3. $x_1 = 1,\ x_2 = 1,\ x_3 = 0$ ***2a iteració - flip posició 0 1a clausula no satisfeta***
4. $x_1 = 1,\ x_2 = 1,\ x_3 = 1$ ***3a iteració - flip posició 2 1a clausula no satisfeta***

Finalment, en el notebook podem veure com s'ha executat l'algorisme sencer 1024 cops per comprovar si realment ens retorna una solució amb alta probabilitat o no. Després d'aquestes execucions s'ha comprovat que obtenim una solució un 93.34% de les vegades, realitzant un nombre quadràtic menys de repeticions de l'oracle de les que faria l'algorisme clàssic.

Capítol 12

Conclusions

L'objectiu principal del treball era resoldre 3-SAT utilitzant computació quàntica per a poder aprendre més sobre un model de computació el qual pràcticament es manté desconegut al llarg de la carrera i com aquest podia ajudar-nos a l'hora de resoldre problemes NP-Complets. Aquest objectiu i tots els descrits en el capítol 3 s'han complert satisfactòriament i aquí podem trobar un resum de la feina feta:

- Generar una introducció a la computació quàntica explicant els components necessaris per entendre el treball.
- Entendre i documentar a manera de guia l'algorisme de Grover i el Quantum counting.
- Dissenyar un algorisme quàntic que mitjançant l'algorisme de Grover i el Quantum counting pot trobar solucions amb alta probabilitat a fórmules 3-CNF donades.
- Entendre i documentar a manera de guia l'algorisme de Schöning i com generalitzar l'algorisme de Grover per tal de poder utilitzar-lo en diferents situacions i diferents problemes.
- Dissenyar un algorisme quàntic que fent servir l'algorisme clàssic de Schöning i l'algorisme de Grover generalitzat aconsegueix, no només trobar solucions de 3-SAT, sinó millorar en complexitat els algorismes quàntics i clàssics que s'han vist al llarg del treball.
- Elaborar una anàlisi de complexitat dels algorismes clàssics i quàntics per tal de poder comparar quantes crides a l'oracle haurien de realitzar en cada cas.
- Implementar els dos algorismes quàntics proposats de forma genèrica per qualsevol fórmula amb Jupyter notebook creant així un manual pas a pas perquè qualsevol lector pugui seguir-ho sense haver de recórrer al document complet.

Personalment, aquest treball m'ha permès introduir-me en el món de la computació quàntica i aprendre diversos conceptes que a l'inici d'aquest desconeixia. Tot i tenir algun coneixement previ, els algorismes tractats, com l'algorisme de Grover o el Quantum counting han sigut nous per mi i m'han ajudat a assentar millor les bases que tenia. També he pogut, per primer cop, dissenyar un circuit quàntic propi, aprendre com enfocar diferents problemes amb aquest model de computació i familiaritzar-me en la manera de portar els algorismes teòrics a la pràctica per tal d'executar-los en un simulador o en un ordinador quàntic real.

Aquest treball també ofereix moltes possibilitats a futur, ja que es podria intentar millorar els algorismes per tal d'utilitzar menys qubits auxiliars o fins i tot explorar altres formes d'abordar el problema com mitjançant Quantum walks [40]. A més a més, seria interessant explorar l'execució en un ordinador quàntic real per veure com efectes com per exemple el soroll afecten la computació i com podem dissenyar algorismes quàntics amb tolerància a errors [41].

Bibliografia

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, pp. 1484–1509, oct 1997.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996.

[3] U. Schöning, "A probabilistic algorithm for k-sat and constraint satisfaction problems," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, (USA), p. 410, IEEE Computer Society, 1999.

[4] A. Ambainis, "Quantum search algorithms," *SIGACT News*, vol. 35, p. 22–35, jun 2004.

[5] Wikipedia, "Qubit," 2023. `https://en.wikipedia.org/wiki/Qubit`, Data d'accés: 22-02-2023.

[6] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, pp. 235–236. McGraw-Hill, 2006.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*, pp. 1079–1080. MIT Press, 2009.

[8] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (New York, NY, USA), p. 151–158, Association for Computing Machinery, 1971.

[9] G. Brassard, P. HØyer, and A. Tapp, "Quantum counting," in *Automata, Languages and Programming*, pp. 820–831, Springer Berlin Heidelberg, 1998.

[10] G. Nannicini, "An introduction to quantum computing, without the physics," 2017.

[11] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[12] Qiskit, "Solving satisfiability problems using grover's algorithm," 2022. `https://qiskit.org/textbook/ch-applications/satisfiability-grover.html`, Data d'accés: 22-02-2023.

[13] Scrum.org, "What is scrum?," 2023. `https://www.scrum.org/resources/what-is-scrum`, Data d'accés: 23-02-2023.

[14] FIB-UPC, "Normativa del treball de fi de grau," 2023. `https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-mencio-addicional-gei.pdf`, Data d'accés: 01-03-2023.

[15] Gantter, "#1 cloud-based project management software," 2023. `https://www.gantter.com/`, Data d'accés: 02-03-2023.

[16] Google Research, "Colab," 2023. `https://research.google.com/colaboratory/intl/es/faq.html`, Data d'accés: 02-03-2023.

[17] talent.com, "Salario medio para jefe de proyecto en españa, 2023," 2023. `https://es.talent.com/salary?job=jefe+de+proyecto`, Data d'accés: 07-03-2023.

[18] talent.com, "Salario medio para investigador en españa, 2023," 2023. `https://es.talent.com/salary?job=investigador`, Data d'accés: 07-03-2023.

[19] talent.com, "Salario medio para programador en españa, 2023," 2023. `https://es.talent.com/salary?job=programador`, Data d'accés: 07-03-2023.

[20] Banco Santander S.A, "¿qué es el coworking?," 2022. `https://www.santander.com/es/stories/que-es-el-coworking`, Data d'accés: 07-03-2023.

[21] EasyOffices, "Plaza de cataluña 1, edificio el triangle, 4ª planta," 2022. `https://www.easyoffices.com/es/coworking/barcelona/edificio-el-triangle`, Data d'accés: 07-03-2023.

[22] Microsoft, "Pricing," 2023. `https://visualstudio.microsoft.com/es/vs/pricing/`, Data d'accés: 07-03-2023.

[23] GitHub, "Pricing," 2023. `https://github.com/pricing`, Data d'accés: 07-03-2023.

[24] Overleaf, "Plans and pricing," 2023. `https://www.overleaf.com/user/subscription/plans`, Data d'accés: 07-03-2023.

[25] Trello, "Planes," 2023. `https://trello.com/pricing`, Data d'accés: 07-03-2023.

[26] Gantter, "Powerful planning. simple pricing.," 2019. `https://www.gantter.com/pricing/`, Data d'accés: 07-03-2023.

[27] Google, "Pricing," 2023. `https://colab.research.google.com/signup`, Data d'accés: 08-03-2023.

[28] A. Ambainis, "Understanding quantum algorithms via query complexity," 2017.

[29] T. Hofmeister, U. Schöning, R. Schuler, and O. Watanabe, "A probabilistic 3-sat algorithm further improved," in *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '02, (Berlin, Heidelberg), p. 192–202, Springer-Verlag, 2002.

[30] D. Rolf, *3-SAT in RTIME(*1.32971^n*): Improving Initial Assignments for Randomized Local Search for 3-SAT Using Joint Clause Pairs*. Diploma-Thesis, Department Of Computer Science, Humboldt University Berlin, Germany, 2003.

[31] S. Baumer and R. Schuler, "Improving a probabilistic 3-sat algorithm by dynamic search and independent clause pairs," in *Theory and Applications of Satisfiability Testing* (E. Giunchiglia and A. Tacchella, eds.), (Berlin, Heidelberg), pp. 150–161, Springer Berlin Heidelberg, 2004.

[32] D. Rolf, *3-SAT in RTIME(*1.32793^n*): Improving Randomized Local Search by Initializing Strings of 3-Clauses*. Electronic Colloquium on Computational Complexity (ECCC), 2003.

[33] K. Iwama and S. Tamaki, "Improved upper bounds for 3-sat.," p. 328, 01 2004.

[34] K. Iwama, K. Seto, T. Takai, and S. Tamaki, "Improved randomized algorithms for 3-sat," in *Algorithms and Computation* (O. Cheong, K.-Y. Chwa, and K. Park, eds.), (Berlin, Heidelberg), pp. 73–84, Springer Berlin Heidelberg, 2010.

[35] T. Hertli, R. A. Moser, and D. Scheder, "Improving ppsz for 3-sat using critical variables," 2011.

[36] R. de Wolf, *Quantum Computing: Lecture Notes*, pp. 56–57. 2023.

[37] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," 2002.

[38] Qiskit, "Simulators overview," 2023. `https://quantum-computing.ibm.com/lab/docs/iql/manage/simulator/`, Data d'accés: 06-05-2023.

[39] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, "Matrix product state representations," 2007.

[40] S. E. Venegas-Andraca, "Quantum walks: a comprehensive review," *Quantum Information Processing*, vol. 11, pp. 1015–1106, jul 2012.

[41] A. Paler and S. J. Devitt, "An introduction to fault-tolerant quantum computing," 2015.