TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Los límites de la computación cuántica para la resolución de problemas NP

Autor
Jorge Gangoso Klöck (alumno)

Directores
Jesús García Miranda (tutor1)
Antonio M. Lallena Rojo (tutor2)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Julio de 2023

Los límites de la computación cuántica para la resolución de problemas NP

Jorge Gangoso Klöck (alumno)

Palabras clave: Quantum Computing, NP, Quantum Fourier Transform, Quantum Phase Estimation, Grover's Algorithm, Qbit, 3SAT, TSP

Resumen

Con la llegada de la computación cuántica y la aparición de los primeros servicios de computación cuántica remota se plantea el trabajo de comparar la eficiencia de los algoritmos cuánticos surgidos en la última década, así como de determinar los límites de ésta tecnología emergente.

Para ello, se describen y proponen algunos problemas pertenecientes a la clase de complejidad NP-completos, de entre los que se seleccionan dos (3SAT y TSP) y se procede a implementar y ejecutar un algoritmo cuántico que resuelva cada uno de estos problemas.

Este trabajo constituye un resumen bibliográfico de numerosas fuentes de información en la materia de la computación cuántica, y, al mismo tiempo, un trabajo experimental en el que se trata de superar los algoritmos clásicos mediante versiones cuánticas.

The limits of quantum computing resolving NP problems

Jorge, Gangoso Klöck (student)

Keywords:Quantum Computing, NP, Quantum Fourier Transform, Quantum Phase Estimation, Grover's Algorithm, Qbit, 3SAT, TSP

Abstract

With the arrival of quantum computing and the first quantum cloud computing services the idea behind this work arises. It intends to compare the efficiency of some of the quantum algorithms proposed in the last decade, as well as trying to determine the limits on this emerging technology.

With this goal in mind, a few problems belonging to the NP-complete complexity class are described. Two of them (3SAT and TSP) are selected to be implemented and executed using a quantum algorithm capable of solving each of these problems.

This project makes a bibliographic summary from multiple sources of information about quantum computing, and, at the same time, it is an experimental essay where we try to surpass classic algorithms with their quantum counterparts.

Yo, **Jorge Gangoso Klöck (alumno)**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 49398653N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jorge Gangoso Klöck (alumno)

Granada a 7 de Septiembre de 2023.

D. **Jesús García Miranda (tutor1)**, Profesor del Área de Álgebra del Departamento Álgebra de la Universidad de Granada.

D. **Antonio M. Lallena Rojo (tutor2)**, Profesor del Área de Física atómica, molecular y Nuclear del Departamento Física atómica, molecular y nuclear de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Los límites de la computación cuántica para la resolución de problemas NP***, ha sido realizado bajo su supervisión por **Jorge Gangoso Klöck (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 7 de Septiembre de 2023.

Los directores:

Jesús García Miranda (tutor1) **Antonio M. Lallena Rojo (tutor2)**

Agradecimientos

Este trabajo está dedicado a todas las personas cercanas a mí. Cada uno de ellos es relevante para que cada día tenga la fuerza necesaria para seguir avanzando y creciendo. Gracias a todos.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Un ordenador es un dispositivo físico que permite realizar tareas de procesamiento de información ejecutando algoritmos. Una tarea de procesamiento de información en computadores depende de tareas físicas como el paso de una determinada corriente eléctrica por un componente electrónico. [34]

Esta relación es la que provoca que al desarrollarse nueva teoría física suela revisarse también la teoría informática ya que la segunda representa una abstracción de la primera, hasta el punto que cuando se realiza programación mediante lenguajes de alto nivel no se tienen en cuenta los transistores que estarán involucrados en el proceso de ejecución del programa. [45]

Por lo tanto, tras el desarrollo de la mecánica cuántica y sus ramas de especialización surge la computación cuántica a partir de la idea de procesar la información haciendo uso de sistemas cuánticos. El procesamiento de información cuántico es el resultado de utilizar la realidad física que la teoría cuántica nos presenta sobre problemas que se creían imposibles o inviables. Los dispositivos que pueden realizar estas tareas de procesamiento cuántico de información reciben el nombre de ordenadores cuánticos. [34]

En primer lugar, daremos un breve repaso histórico del contexto que da pie a esta tecnología emergente (ver apartado 1.1), seguido de la motivación tras este trabajo y su interés científico y académico (ver apartado 1.2).

En segundo lugar se proporcionarán una serie de conocimientos previos en el capítulo 2 que son de grán importancia para entender los procedimientos que se emplean en este trabajo y la utilidad del mismo. El tercer paso será detallar los elementos tanto Hardware como Software requeridos para poder replicar los experimentos realizados en este proyecto (capítulo 3). Tras esto se explica la planificación seguida durante el proyecto desde la idea hasta la redacción (capítulo 4), y comienzan los apartados donde se trata con conocimiento especializado de la materia.

En el capítulo 5 se explican los problemas planteados para su posible resolución, junto con posibles soluciones propuestas en el campo de la computación cuántica.

Si el lector está interesado únicamente en la información relacionada con el manejo de la computación cuántica y los experimentos puede saltar al capítulo 6. En dicho capítulo se detallan los conocimientos necesarios para construir y comprender un circuito o algoritmo cuántico. Algunos de los temas tratados en este capítulo son los cúbits (ver apartado 6.1), las esferas de Bloch (ver apartado6.1) y las puertas cuánticas (ver apartado 6.3).

En el capítulo 7 se realiza ya un trato específico a los algoritmos de *Grover* y de *estimación de fase cuántica* para centrarse en su funcionamiento e implementación tanto general como específica para la resolución de los problemas 3-satisfacibilidad (ver apartado 5.1) y viajante de comercio (ver apartado 5.4).

Los resultados de estas implementaciones y algunos detalles de ejecución pueden verse en el capítulo 8 que va seguido de las conclusiones y trabajos futuros (capítulo 9).

Al final del documento se encuentra la bibliografía (capítulo 10) que contiene las referencias empleadas en el desarrollo del trabajo.

1.1. Contexto histórico

La primera revolución cuántica comienza con el deseo de una explicación consistente al espectro de la *radiación de cuerpo negro* tanto a bajas como a altas frecuencias. En búsqueda de esto, M. Plank introduce en 1900 la cuantización del intercambio de energía entre luz y materia [37]. A. Einstein da un paso más allá y en 1905 propone la cuantización de la propia luz como medio para comprender el *efecto fotoeléctrico*[18]. Las propiedades que dedujo fueron puestas a prueba por R. A. Millikan en 1914 [31]. En la misma época, pruebas convincentes de la existencia de moléculas - hecho puesto en duda hasta el principio del siglo XX - fueron proporcionadas por varias observaciones entre las cuales se encuentra la explicación de Einstein sobre el movimiento browniano [19].

Esto, junto a otros numerosos experimentos, convenció a físicos y filósofos de la granuralidad de la materia y la cuantización de la energía en el mundo microscópico, dando pie al desarrollo de la mecánica cuántica.

Este desarrollo de la mecánica cuántica en el comienzo del siglo XX permitió la comprensión de la estabilidad de la materia, las propiedades térmicas y mecánicas de los materiales, la interacción entre radiación y materia, y otras numerosas propiedades del mundo microscópico que habían

sido imposibles de explicar mediante la física clásica.

Este *desarrollo conceptual* fue seguido de un *desarrollo tecnológico*. Es gracias a los conocimientos sobre la estructura y propiedades de la materia adquiridos mediante la mecánica cuántica que los físicos e ingenieros fueron capaces de desarrollar el transistor, en 1948, y el láser, a finales de los años cincuenta, dos tecnologías clave en el transporte de información y en otros muchos campos científicos y comerciales.

Sus aplicaciones hoy en día pueden observarse en todas partes, los transistores fueron el orígen de los actuales circuitos integrados, y el láser se emplea en lectores de códigos de barras, lectores y reproductores de CD, herramientas médicas, etc. Uno de los usos quizás más importantes del láser es en telecomunicaciones, donde proporcionó un aumento dramático al flujo de información que podía transmitirse.

Años después, en 1964, John Bell presenta un artículo [4] donde trata de "completar" la mecánica cuántica tras cierto debate surgido a partir de otro artículo presentado por Einstein, Podolsky y Rosen (EPR) donde se pone en duda la completitud del modelo físico cuántico. Para ello propone el uso de "variables ocultas" que completaran las explicaciones de ciertas reproducciones cuyos resultados no casaban con las predicciones hechas mediante la mecánica cuántica cuando el entrelazamiento de partículas entraba en juego. Sin embargo, esta tentativa de solución, que permitía explicar algunos de estos casos, no era válido para *todos* los casos. Este resultado sería conocido en adelante como el *Teorema de Bell* que muestra la incompatibilidad entre la mecánica cuántica y las teorías de variables ocultas locales, y que las correlaciones que pueda predecir cualquier modelo de variables ocultas locales están limitadas por unas desigualdades - hoy en día llamadas las *desigualdades de Bell* - que son violadas por algunas predicciones cuánticas. Este teorema dio una base con la que pudo zanjarse experimentalmente el debate previo.

Más cercano al tema que concierne a este trabajo, en 1970, algunos físicos inventaron métodos para manipular y observar objetos elementales básicos, como un único fotón [26], electrón o ion. Fueron capaces de atrapar partículas cargadas durante horas (incluso días y meses) empleando campos eléctricos y magnéticos que contenían la partícula en una cámara vacía, alejada de cualquier pared material.

La existencia de las desigualdades de Bell, que establecen una frontera clara entre el comportamiento clásico y el cuántico, y su violación experimental, son resultados conceptuales importantes que nos hacen reconocer el carácter extraordinario del entrelazamiento cuántico. Sin embargo, se ha descubierto que este entrelazamiento también ofrece posibilidades totalmente revolucionarias en el dominio del tratamiento y transmisión de información, se trata de la segunda revolución cuántica. Este campo se llamaría *informa-*

ción cuántica y trata de implementar soluciones radicalmente novedosas que prometen aplicaciones sorprendentes. Los dos ejemplos principales dentro de este campo son la criptografía cuántica [32], ya operativa, y la computación cuántica [39][33], aún en una etapa muy temprana de su desarollo. [5]

A pesar del interés y relevancia de la criptografía cuántica y su alta relación con la especialidad de "Computación y Sistemas Inteligentes (CSI)" del grado de Ingeniería Informática, nos centraremos únicamente en la computación cuántica ya que es la rama que se emplea en este proyecto.

A comienzos de los años ochenta, algunas asunciones fundamentales en la teoría de la información comenzaron a ser desafiadas por nuemrosos físicos que proponían que, si se tuviera un ordenador cuántico, sería posible implementar algoritmos radicalmente nuevos para resolver ciertas tareas. Hay múltiples personas que pueden ser nombradas en este escenario como Landauer, Feynman o Deutsch, pero el mayor evento sucedió en 1994 cuando P. Shor [41] demostró que un ordenador cuántico debería permitir la factorización de un número grande, en un tiempo mucho menor de lo que un ordenador clásico es capaz. Las implicaciones de esta demostración chocan directamente con temas de la informática teórica como son las clases de complejidad de los problemas, que explicaremos en el capítulo 5, y la tésis de Church-Turing [12].

Es en este contexto, con la promesa de algoritmos capaces de mejorar los tiempos *superpolinomiales* que poseemos actualmente para ciertos problemas, y el nacimiento de la computación cuántica como herramienta de uso generalizado que se encuentra este proyecto.

1.2. Motivación

La idea de este trabajo surge a partir de la premisa de que las clases de complejidad algorítmica, de las que se hablará en el capítulo 2 no tienen por qué coincidir cuando tratamos con ordenadores clásicos o cuánticos. Está demostrado que existen algoritmos como el ya mencionado *algoritmo de factorización Schor* [41], con altas implicaciones en algunas ramas de la computación, en este ejemplo la criptografía, en que un algoritmo cuántico representa una mejora teórica considerable a su equivalente clásico.

Con esto en mente se plantea la posibilidad de emplear dichos algoritmos para resolver otros problemas complejos mediante computadores cuánticos, empleando sus propiedades únicas para obtener mejoras teóricas y quizás empíricas respecto a los computadores clásicos.

De obtenerse resultados satisfactorios en este campo se obtendría el primer paso hacia un cambio sin precedentes en el campo de la informática, y, con la extensión de los sistemas informáticos en el mundo y sus innumerables aplicaciones en todos los campos, podría suponer un cambio en el funcionamiento actual de nuestra sociedad.

Es por esto que en este proyecto se ha decidido comprobar el estado actual de la computación cuántica, tanto a nivel de accesibilidad de los recursos, como al desarrollo algorítmico y hardware de esta tecnología supuestamente capaz de romper algunos paradigmas actuales en informática.

En resumen, a lo largo de este proyecto se dará respuesta a cómo funciona esta tecnología emergente que es la computación cuántica. Para ello se comenzará explicando unos conocimientos generales en el ámbito de la computación y la física cuántica, y, posteriormente, se explicarán los algoritmos que han sido implementados y ejecutados junto a los resultados obtenidos.

Capítulo 2

Conocimientos Previos

2.1. NP-completitud

Es necesario comenzar definiendo algunos conceptos clave que serán utilizados reiteradamente, estos conceptos son los de problemas **P**, **NP** y **NP-completos**. Cada una de estas representa una *clase de complejidad algorítmica*. Esto significa que la dificultad de resolver estos problemas mediante un algoritmo puede ser descrita y clasificada en diferentes clases en función de lo que llamaremos *función de complejidad*, que no es más que una función matemática que describe la cantidad de operaciones necesarias para resolver un determinado problema en función de una variable N que representa el *tamaño del problema*.

Ejemplo 2.1 *Un problema consiste en, dado un vector de números, determinar el número de mayor valor. Si se ejecuta el problema con un vector de cinco números, diremos que el tamaño del problema, N será igual a* 5*. Si en otra ejecución se tiene un vector de diez números, el tamaño del problema N será igual a* 10*.*

Una vez comprendido el concepto de la complejidad de un problema en función del tamaño de entrada, veremos algunas de las diferentes clases de complejidad que existen.

Definición 2.1 (Problema P) *. Llamamos **P** a los problemas que pueden ser resueltos con "soluciones eficientes", es decir, aquellos que pueden ser ejecutados en un tiempo acotado por un polinomio del tamaño del problema, es decir, su función de complejidad es un polinomio cuya variable es N. La letra **P** proviene del inglés "Polynomial Time". [21]*

Aunque no nos centraremos en esta clase de problemas es necesaria su comprensión para explicar las siguientes clases de problemas. Pongamos el

siguiente ejemplo para explicar mejor qué implica que un problema pertenezca a la clase P.

Ejemplo 2.2 *Partimos del ejemplo anterior en el que se tiene un vector de números desordenados v, del cual queremos conocer el valor más alto que contiene. Para hacer dicha tarea, se debe iterar sobre cada elemento del vector, comparar su valor con el mayor valor almacenado, y, en caso de ser mayor que este, sustituir el valor almacenado por el nuevo valor más alto encontrado. Es decir, para cada elemento, realizaremos una o dos operaciones, o lo que es lo mismo, en promedio, realizaremos* 1.5 *operaciones por cada elemento en el vector.*

Por tanto, si el vector tiene 10 *elementos, en promedio realizaremos* $1.5 * 10 = 15$ *operaciones,* 10 *en el caso de que ningún elemento sea mayor que el inicial (que se supone almacenado previamente como mayor valor, en caso contrario añadir una operación a la cuenta), y* 20 *en el caso de que el vector se encuentre ordenado de manera ascendente.*

Esto puede generalizarse para el caso en que el vector tenga N elementos, realizando en el mejor caso N operaciones, en el peor $2 * N$ operaciones, y en promedio $1.5 * N$ operaciones. Es sencillo observar que en todos los casos la cantidad de operaciones viene definida por el tamaño del vector N multiplicado por una constante que en nuestro caso oscila entre 1 *y* 2*. Si tomamos N como una variable, podemos decir que la función que define el número de operaciones necesarias, es decir, su complejidad, viene dada por un polinomio cuya variable es el tamaño del problema: $f(N) = c * N, c \in [1, 2]$. Los problemas cuyas funciones de complejidad que pueden expresarse mediante un polinomio de cualquier grado donde la variable es el tamaño de entrada son a lo que llamamos problemas P. Un problema cuya función de complejidad sea de la forma $f(N) = 2^N$ no formaría parte de la clase P.*

A continuación se define lo que significa un problema NP. Para la definición emplearemos el término máquina de Turing, y, posteriormente, hablaremos de las máquinas de Turing no-deterministas.

Definición 2.2 (Máquina de Turing) *. Una Máquina de Turing es un concepto matemático, consistente en un mecanismo teórico formado por un cabezal de lectura capaz de leer y escribir determinados símbolos sobre una cinta que puede desplazar a izquierda y derecha para cambiar de posición.*

Las máquinas de Turing son lo que conocemos como un *sistema de cómputo* y se emplean para decidir si hay un algoritmo para resolver un problema matemático dado. [40]

En la descripción original de la máquina de Turing propuesta en 1936 por A. Turing [46], se dice que este sistema de cómputo es como una versión

inferior a un ser humano, capaz de encontrarse únicamente en un número finito de estados $q_0, q_1, ..., q_n$ a las que llama *"m-configuraciones"*. A esta máquina se le proporcionaría una *cinta*, que cumpliría un rol semejante al papel que emplean las personas para hacer cálculos, la cual estaría segmentada en *celdas* sobre cada cual se podría almacenar un único *símbolo*. A continuación, sobre esta máquina se definen una serie de transiciones, a las que denotaremos con el símbolo δ que determinan el comportamiento de la máquina en función de su *m-configuración* y del símbolo leído sobre la cinta.

Lo que acabamos de describir es conocido como una máquina de Turing determinista de una cinta (MTD) y pòsee un cabezal con el que puede leer los símbolos en la cinta de entrada y escribir en la misma cualquier símbolo conocido por la máquina. Este conjunto de símbolos denotado por el símbolo Γ recibe el nombre de *alfabeto de trabajo*.

La totalidad de estados o configuraciones en que puede encontrarse la MTD recibe el símbolo Q, y entre estos diferenciamos dos, a los que denotamos como q_Y y q_N que representan un estado de aceptación y de rechazo respectivamente. Esto nos indica que si la MTD alcanza en algún momento el estado q_Y, el contenido de la entrada se acepta como válido y el proceso finaliza satisfactoriamente. Por el contrario, si la MTD alcanza el estado q_N, el contenido de la cinta es rechazado y el proceso finaliza también.

Dependiendo de la programación de la MTD, el significado tras estos dos estados y la utilidad de las transiciones varía, dando lugar a un determinado programa con una función específica. [30]

q	0	1	#
q_0	$(q_1, \#, +1)$	$(q_2, \#, +1)$	$(q_Y, \#, -1)$
q_1	$(q_1, 0, +1)$	$(q_1, 1, +1)$	$(q_3, \#, -1)$
q_2	$(q_2, 0, +1)$	$(q_2, 1, +1)$	$(q_4, \#, -1)$
q_3	$(q_5, \#, -1)$	$(q_N, \#, -1)$	$(q_Y, \#, -1)$
q_4	$(q_N, \#, -1)$	$(q_5, \#, -1)$	$(q_Y, \#, -1)$
q_5	$(q_5, 0, -1)$	$(q_5, 1, -1)$	$(q_0, \#, +1)$

Cuadro 2.1: Ejemplo de programa en MTD. $M = (\Gamma, Q, \delta)$.

Lo que se observa en la tabla 2.1 es una posible descripción del conjunto de transiciones de una MTD cuya finalidad es reconocer si el conjunto de símbolos de la *cinta* de entrada forman un palíndromo. Este programa ha sido generado por mí y no es la única configuración posible de una MTD que acepte las mismas entradas.

En la parte superior de la tabla se encuentran los símbolos que puede leer y escribir la MTD y que conforman su alfabeto de trabajo: $\Gamma = (0, 1, \#)$. A la izquierda tenemos los estados o m-configuraciones en los que puede encontrarse la máquina de Turing: $Q = (q_0, q_1, q_2, q_3, q_4, q_5)$. Finalmente en

el interior de las celdas tenemos las transiciones que permiten a la máquina determinar qué acción realizar ante cierta entrada encontrándose en un determinado estado: $\delta(q, s) \forall q \in Q, s \in \Gamma$.

Por tanto, una transición concreta viene definida por un estado inicial (la fila en la que se encuentra), un símbolo leído (la columna en que se encuentra) y una tripleta que define el comportamiento de la máquina ante esta situación. Veamos como sería el funcionamiento de la máquina mediante una transición.

Ejemplo 2.3 *La transición $\delta(q_0, 0) = (q_1, \#, +1)$ determina el siguiente contexto: la máquina se encuentra en el estado q_0 y en la cinta donde se encuentra el cabezal de lectura/escritura hay escrito un 0, la máquina pasará al estado q_1, escribirá en dicha celda el símbolo blanco (#) y se moverá hacia la derecha (+1).*

Podemos ver una traza de nuestro programa en la figura 2.1. Las celdas azules representan blancos, y las que contienen un valor en su interior representan aquellas con elementos pertenecientes a Γ. La flecha inferior indica la posición del cabezal en cada momento y en la parte superior se indica el estado actual de la MTD, comenzando en q_0. Al final de la ejecución la MTD se encuentra en el estado de aceptación q_Y por lo que se reconoce la entrada como válida lo cual puede apreciarse debido a la aparición de una línea en verde que dicta "accepted". En este ejemplo, debido a la programación concreta de la MTD, que se alcance un estado de aceptación quiere decir que la entrada introducida (la cadena 10101) es un palíndromo.

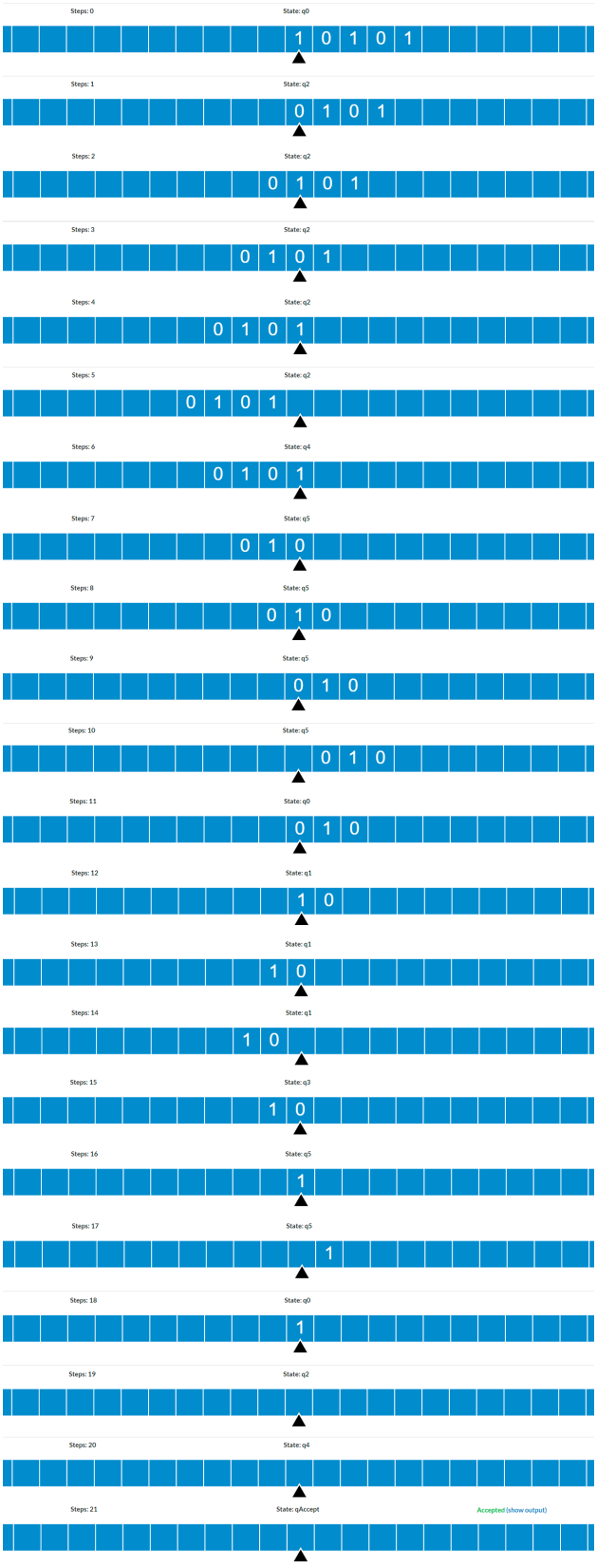


Figura 2.1: Traza de un programa en MTD para reconocer palíndromos binarios

A diferencia de una MTD, una máquina de Turing no determinista (MTND) tiene un numero finito de *transiciones* entre las que elegir en cada paso. Si alguno de los posibles caminos que se ramifican de esas decisiones alcanza un estado de aceptación, la entrada se considera aceptada. Se puede pensar en este proceso como el de una ejecución en paralelo de todos los posibles caminos hasta que bien uno finalice o no queden *transiciones* posibles.[47]

Definición 2.3 (Problema NP) . *Llamamos **NP**, o, "Non-Deterministic polynomial-time", a los problemas que pueden ser resueltos por una máquina de Turing no-determinista en un numero de pasos acotado por un polinomio del tamaño de entrada. [16]*

Generalmente el proceso de resolución de un problema mediante un sistema de cómputo no-determinista viene separado en dos pasos. El primer paso es una fase de *adivinación*, tras el cual lo que queda en la cinta es una posible solución a nuestro problema. La segunda fase consiste en la *verificación* de la solución propuesta en el primer paso. Para que el problema se considere NP la fase de verificación debe poder realizarse en tiempo polinómico. [30]

Dando un paso más allá de los problemas NP, para dar una definición correcta de NP-completitud es necesario explicar previamente el concepto de reducción.

Dados dos problemas X e Y, decimos que X "se reduce a" Y si un algoritmo para resolver Y también representa un algoritmo para resolver X. Para poder afirmar esto, se necesita un determinado algoritmo de transformación $T()$ que transforme una entrada para el algoritmo X en una entrada para el algoritmo Y, y una solución del algoritmo Y en una solución del algoritmo X. (Puede verse un ejemplo de reducción en el apartado 5.4) Teniendo en cuenta esto:

Definición 2.4 (NP-completitud) . *Un problema X es **completo** dentro de la clase de complejidad **NP** si para todo problema Y, perteneciente a **NP**, dicho problema es reducible a X. Por tanto se pueden entender como los problemas más difíciles de resolver dentro de la clase de complejidad **NP**. [16]*

Estos son los problemas con los que trataremos en este trabajo.

2.2. Mecánica cuántica

La mecánica cuántica como se introduce en el apartado 1.1 surge a partir de la imposibilidad de modelar y explicar algunos fenómenos como el *efecto fotoeléctrico*, las ondas electromagnéticas o la estructura atómica empleando la mecánica clásica.

La mecánica cuántica engloba un sistema de mecánicas basadas en la teoría cuántica, que estudia el comportamiento de partículas de tamaño atómico. [40]

Esta rama de la física es la base subyacente al uso que hacemos de la computación cuántica. Los elementos que emplearemos en computación cuántica, desde un cúbit hasta una puerta Hadamard funcionan a partir de interacciones con sistemas cuánticos. Algunos de estos fenómenos, como el entrlazamiento cuántico, tienen especial interés para el proyecto debido a las aplicaciones en cálculos de procesamiento de información.

2.3. Computación cuántica

Definición 2.5 (Computación cuántica) . *La computación cuántica es el diseño y teoría de los sistemas computacionales que dependen de efectos cuánticos para su funcionamiento. [40]*

Esto puede realizarse empleando componentes muy pequeños, de tamaño atómico o molecular, para almacenar o procesar información.

La computación cuántica une los conceptos de teoría de la información, informática y mecánica cuántica [43], englobando así una serie de términos, teoría, librerías y funciones propias que no comparte con ninguna otra rama de computación. Es posible emplear dichas herramientas sin comprender plenamente la teoría tras la mecánica cuántica, pero es aconsejable poseer al menos una base para una mejor comprensión de las actividades que se puedan realizar. Como abstracción que supone la computación al procesamiento de información [45], puede ser difícil en ocasiones ligar las aplicaciones que se crean mediante la computación cuántica con las interacciones de la mecánica cuántica que hacen realidad estos cálculos. Es por ello que a lo largo del trabajo no se trazarán las relaciones entre mecánica y computación cuántica excepto cuando sea necesario, y partiremos de varias definiciones y operaciones que veremos en el capítulo 6 para construir las implementaciones de los algoritmos.

Desde otra perspectiva, a nivel físico, cualquier sistema cuántico con dos estados que pueden diferenciarse podría emplearse para computación cuántica. Algunos de los que se utilizan actualmente o que podrían llegar a

utilizarse son: [50]

- *Fotones*, o partículas cuánticas de luz. La polarización de estos puede ser vertical, horizontal o una composición de ambos.
- *Iones atrapados*. Un *ion* es un átomo que tiene carga eléctrica (su carga total no es neutra) debido a la ganancia o pérdida de electrones. Estos iones pueden atraparse en las llamadas *trampas iónicas* empleando campos eléctricos. Los niveles de energía de estos iones es lo que se emplea como medida.
- *Átomos fríos*. Átomos neutros pueden ser atrapados a bajas temperaturas usando *trampas magneto-ópticas*, que emplea campos magnéticos y láser para atrapar y enfriar los átomos. Una vez más se emplean los niveles de energía del átomo.
- *Resonancia magnética nuclear*. Los núcleos de los átomos y moléculas también tienen polarización, que suele llamarse al igual que con los fotones *spin*. El *spin* puede medirse y emplearse en computación cuántica utilizando las frecuencias de precesión de estos núcleos al verse expuestos a campos magnéticos fuertes.
- *Puntos cuánticos*. Se trata de "átomos artificiales" formados de la unión de un electrón con un dispositivo semiconductor. Estos elementos poseen un equivalente al *spin* que se puede emplear para la computación cuántica.
- *Superconductores*. En un circuito superconductor, la electricidad fluye con resistencia cero. El flujo magnético a través del inductor y la carga de un condensador causan una energía potencial armónica con niveles de energía discretos y equidistantes. A esto se añade una *unión de Josephson* que es una capa que produce un distanciamento no equidistante de los niveles de energía y permite distingirlos. Dos de estos niveles que puedan diferenciarse pueden utilizarse como base para computación cuántica.

Capítulo 3

Especificación y requisitos

Este trabajo depende esencialmente de la ejecución remota de circuitos y algoritmos que requieren un hardware específico (computadores cuánticos o simuladores de estos) para operar. Por esto, son necesarios algunos preparativos previos a realizar los experimentos. En este apartado se explican cuáles son los pasos a seguir para recrear los experimentos realizados en este trabajo y qué conocimientos son necesarios para manipular las herramientas que se utilizan.

En primer lugar, se detallarán las especificaciones del ordenador en el que se han llevado a cabo todos los experimentos y la redacción del trabajo. Como aviso previo, el ordenador cuyas especificaciones se dan a continuación no asume apenas carga de trabajo ya que una gran parte de los cálculos se realizan de forma remota como se explicará más adelante.

En segundo lugar se explicarán los componentes software empleados durante el desarrollo del proyecto, su proceso de instalación y un breve repaso sobre su funcionamiento.

3.1. Hardware

Estas son las especificaciones del ordenador en el que se ha realizado el proyecto:

Componente	PC
S.O.	Microsoft Windows 10 Pro
Versión S.O.	10.0.19045 compilación 19045
Arquitectura	PC basado en x64
Placa Base	Asus PRIME B350-PLUS
Versión BIOS	American Megatrends Inc. 4801
Procesador	AMD Ryzen 5 1600X Six-Core Processor 3.60 GHz
Memoria RAM	2x Kingston DDR4 8GB
Tarjeta Gráfica	NVIDIA GeForce GTX 1060 3GB

Cuadro 3.1: Especificaciones del PC empleado para el trabajo

Más información sobre el hardware que ha llevado a cabo los cálculos en la nube puede encontrarse en el capítulo 8.

A continuación pasaremos a mencionar al Software necesario para replicar los experimentos y la forma en que pueden instalarse o configurarse.

3.2. Software

3.2.1. Qiskit

Qiskit es un kit de desarrollo software open-source creado por IBM para trabajar con copmutación cuántica a nivel de circuitos, pulsos y algoritmos. Emplearemos la versión 0.43.1 de Qiskit base. Este kit de desarrollo incorpora algunos componentes adicionales para distintas funciones concretas como simuladores o algoritmos previamente implementados. Los paquetes instalados y sus versiones son:

- Qiskit-terra: '0.24.1'. Es el componente principal de Qiskit, incluye las instrucciones básicas para construir y trabajar con circuitos cuánticos. También contiene un compilador que permite trabajar con ordenadores cuánticos y una interfaz para ejecutar programas en distintas arquitecturas.
- Qiskit-aer: '0.12.0'. Es un simulador de altas prestaciones que incorpora modelos de ruido realistas.
- Qiskit-imbq-provider: '0.20.2'. Es un paquete que incluía las funciones de conexión con el servidor de IBM a través de la clave API. Este paquete queda en desuso tras una actualización de Qiskit en la que sus funcionalidades han sido migradas a otros paquetes.

Para instalar Qiskit y sus paquetes en las versiones mencionadas en este proyecto se ha procedido mediante un software llamado Anaconda ya que no sólo facilita la instalación de Qiskit, también ofrece una distribución de Python que necesitaremos para este protecto.

Anaconda encapsula una distribución abierta de python junto a otros lenguajes de programación, y otorga un entorno en el que es sencillo instalar librerías auxiliares como Qiskit para utilizar junto a los cuadernos "Jupyter Notebook", una interfaz de intérprete apoyado sobre navegador web que permite organizar código y texto formateado en pequeños bloques. Para instalar Anaconda basta con acceder a la web oficial (https://www.anaconda.com/), descargar el instalador automático que correponda para la arquitectura y sistema operativo que se utilice (en nuestro caso Windows x64) y ejecutar el software de instalación.

Una vez instalado Anaconda tenemos acceso a uno de sus componentes llamado *Anaconda Prompt*. Una consola de comandos propia de Anaconda mediante la que se pueden instalar paquetes y distribuciones de todos los lenguajes de programación compatibles. Se emplea dicha consola de comandos para instalar el paquete auxiliar para computación cuántica Qiskit.

Es posible emplear Qiskit en muchos otros entornos de programación pero este ha sido escogido por la sencillez y claridad visual que otorga Jupyter Notebook y por ser un entorno que ha sido previamente utilizado durante el grado de Ingeniería Informática.

3.2.2. IBM Quantum

Para tener acceso al servicio de Cloud Quantum Computing de IBM necesitamos registrarnos en el servicio. Comenzamos este proceso creando un IBMid, introduciendo los datos personales y una contraseña de acceso. Posteriormente se debe seleccionar un sistema de autentificación de dos factores como puede ser un correo electrónico, un dispositivo móvil o una aplicación de autentificación basada en tiempo. Tras esto se solicitan algunos datos personales más como la institución a la que pertenece el usuario y el motivo por el que solicita emplear los servicios de IBM.

Al finalizar el proceso de creación de cuenta se nos concede acceso a un menú personal (ver figura 3.1) desde donde podremos administrar nuestra cuenta y que contiene algunos aspectos de gran relevancia para el proyecto como el token o llave API. La llave API es necesaria para emplear algunos servicios de qiskit como el acceso a los simuladores de IBM y a sus computadores cuánticos. Esta llave API obtenida en el menú personal de "IBM quantum experience" se introduce en nuestro código a modo de identificación.

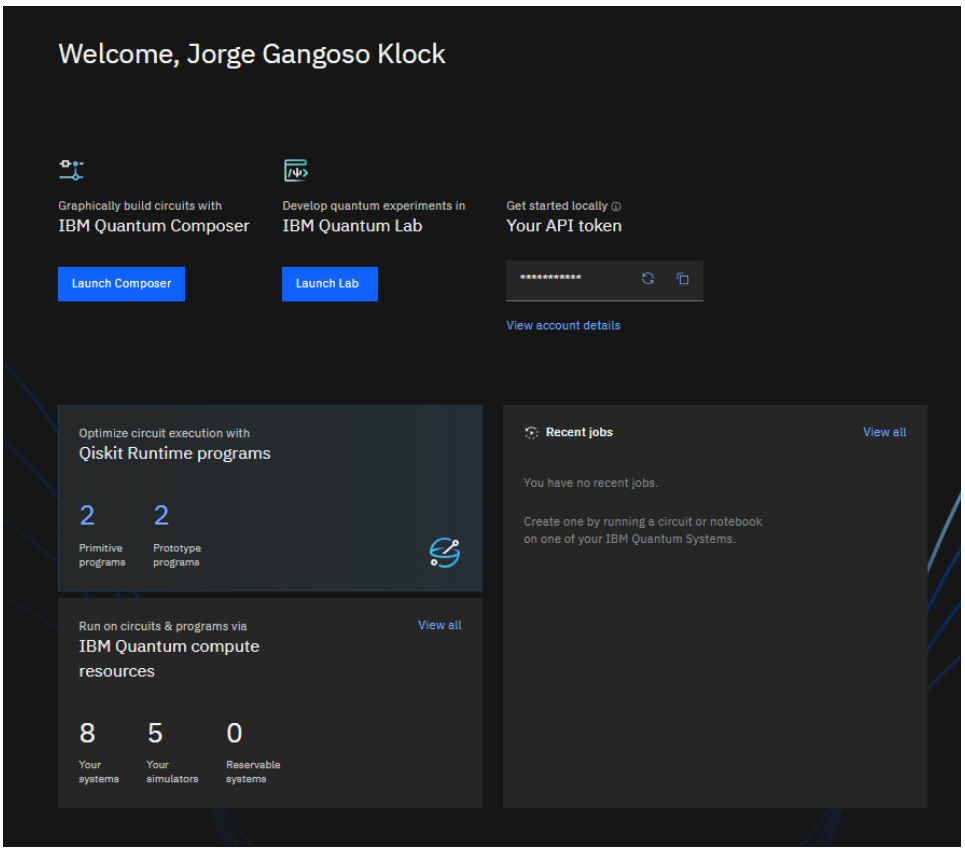


Figura 3.1: Menú principal de IBMQ

Una de las opciones incluidas en este entorno de desarrollo y que será utilizado con frecuencia a lo largo del trabajo es el llamado IBM Quantum Composer. (ver figura 3.2) Un constructor de circuitos visual, que permite crear tus circuitos arrastrando los operadores desde un panel en el extremo izquierdo de la pantalla hasta el esquema del circuito.

Figura 3.2: Editor de circuitos de IBMQ con un circuito de teleportación de información de ejemplo

En la parte izquierda se encuentran las puertas cuánticas (ver apartado 6.3) predeterminadas, así como aquellas que defina el usuario como se observa en la figura 3.3.

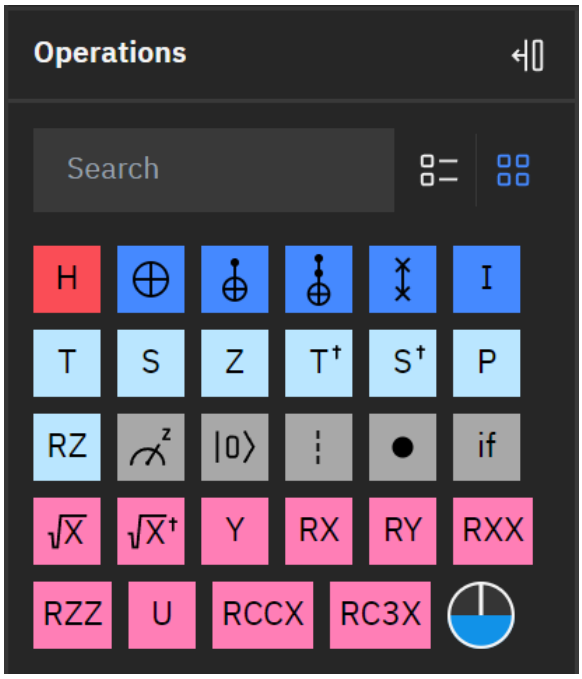
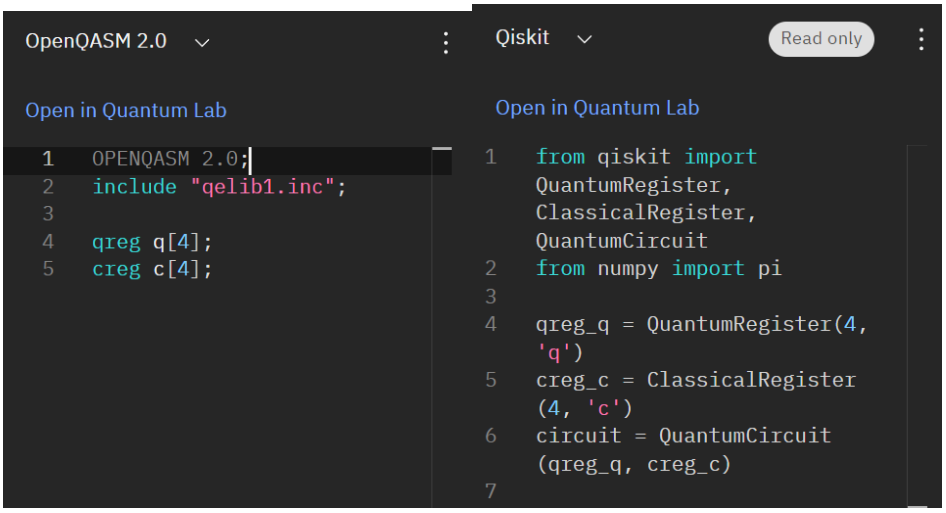


Figura 3.3: Panel de operadores en el editor de circuitos de IBMQ.

A la derecha hay un editor de código donde puede especificarse el circuito. Ya que el código y el circuito se encuentran sincronizados en todo momento, pueden observarse los cambios producidos en el código al añadir o quitar un operador del circuito y de la misma manera puede verse como se ve alterado el circuito al añadir o quitar código. Este código puede escribirse y leerse en openQASM 2.0 (ver apartado 3.2.4) como se observa en la imágen de la izquierda (figura 3.4a) y leerse en Qiskit como se aprecia en la imágen de la derecha (figura 3.4b).



(a) OpenQASM 2.0 **(b)** Qiskit

Figura 3.4: Código de un circuito sin operadores con un registro cuántico de 4 cúbits y un registro clásico de 4 bits en openQASM 2.0. ((a)) y Qiskit ((b))

Empleando esta interfaz de código podemos crear operadores propios a conveniencia, incorporando parámetros, subrutinas y variables como se muestra en la figura 3.5.

OpenQASM 2.0

Open in Quantum Lab

```
OPENQASM 2.0;
include "qelib1.inc";

// Controlled Ugate
gate uni(a, b, c, d) x, y, z {
  cu1(c - a) x, y;
u1(a) x;
cu1(b - a) x, z;
ccx x, y, z;
cu1((d - c + a - b) / 2) x, z;
ccx x, y, z;
cu1((d - c + a - b) / 2) x, y;
cu1((d - c + a - b) / 2) x, z;
}

// C-U = tensor of U1,U2,U3,U4
gate bigU(a, b, c, d, e, f, g, h, i, j, k, l) m, n, o, p, q, r, s, t, u {
  uni(0, a, b, c) m, n, o;
uni(d, 0, e, f) m, p, q;
uni(g, h, 0, i) m, r, s;
uni(j, k, l, 0) m, t, u;
}

```

Figura 3.5: Código en OpenQASM donde se definen dos puertas cuánticas personalizadas

Finalmente, incluye un predictor de los estados de los cúbits (ver apartado 6.1) en tiempo real como se observa en la figura 3.6, que, mediante esferas de Bloch (ver apartado 6.1), vectores de estado o probabilidades, te muestra los posibles resultados de una ejecución de tu circuito. Para funcionar requiere que el circuito contenga menos de seis cúbits

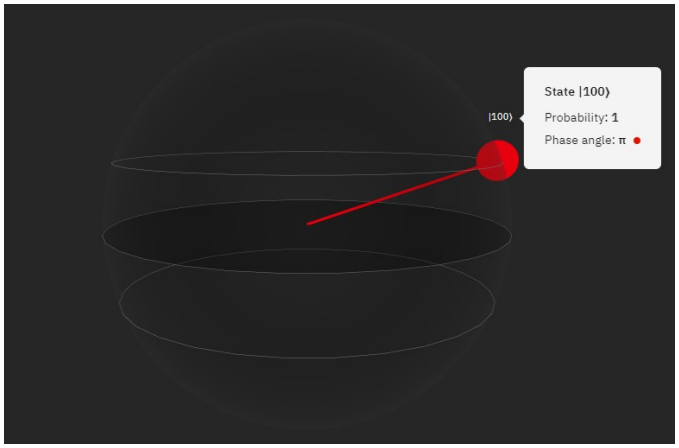


Figura 3.6: Predicción de estado de un circuito con tres cúbits mostrado en IBM-Quantum Composer

3.2.3. Python

El lenguaje de programación principal empleado para este trabajo ha sido Python, ya que es el lenguaje compatible con Qiskit que como ya hemos comentado previamente es una herramienta vital de este proyecto.

Para realizar los experimentos se ha redactado el código necesario para describir y ejecutar los circuitos en la interfaz de Anaconda llamada Jupyter Notebooks que dispone de componentes de sencillo acceso y claridad visual para emplear Python y conectarnos mediante Qiskit a los computadores cuánticos de IBMQ.

Para vincular la sesión de Jupyter con nuestra cuenta IBMid deberemos instalar el paquete qiskit_ibm_provider y emplear la función *save_account(API_Key)* para conectarnos mediante nuestra cuenta a los servicios IBMQ. A continuación, se describen las operaciones y el circuito para, posteriormente, ejecutarlo.

La ejecución puede hacerse en un simulador (en nuestro caso: *'ibmq_qasm_simulator'*) o en un ordenador cuántico real. Hacer esto último requiere obtener una lista de los dispositivos 'backend' disponibles como se muestra en la figura y emplear la función

```
1    execute(circuit, backend)
```

pasando como parámetros el circuito o algoritmo a ejecutar y el dispositivo en el que se desea solicitar la ejecución. Tras hacer esto el trabajo se añade a una cola de ejecuciones como se detallará más en profundidad en el apartado de pruebas.

```
1 from qiskit_ibm_provider import IBMProvider
2 provider = IBMProvider()
3
4 # display current supported backends
5 print(provider.backends())
```

```
[<IBMBackend('ibmq_lima')>, <IBMBackend('simulator_mps')>, <IBMBackend('simulator_statevector')>, <IBMBackend('simulator_stabil
izer')>, <IBMBackend('ibmq_manila')>, <IBMBackend('ibmq_quito')>, <IBMBackend('ibmq_belem')>, <IBMBackend('simulator_extended_s
tabilizer')>, <IBMBackend('ibm_nairobi')>, <IBMBackend('ibm_lagos')>, <IBMBackend('ibm_perth')>, <IBMBackend('ibmq_jakarta')>,
<IBMBackend('ibmq_qasm_simulator')>]
```

Figura 3.7: Listado de backends disponibles mostrados mediante la función backends()

3.2.4. OpenQASM 2.0

OpenQASM es un lenguaje de programación diseñado específicamente para el desarrollo de circuitos y algoritmos cuánticos. Está pensado como una herramienta que cumpla el paso intermedio entre los compiladores de alto nivel y el hardware cuántico. Su especificiación completa puede ser accedida públicamente en ([14]).

En este trabajo se ha empleado este idioma a la hora de generar circuitos mediante el compositor de IBMQ alternando con el seleccionador de operadores básicos (figura 3.3). La especificación de un circuito en código proporciona mucha más capacidad de expresión y pueden definirse rutinas que mediante el editor visual no es posible, pero también constituye un trabajo más difícil y riguroso ya que debe respetarse la sintaxis y gramática propias del lenguaje de programación, por ello una aproximación híbrida empleando cada uno de los editores según la situación se presente más favorable para uno u otro ha sido la decisión para los experimentos llevados a cabo en IBMQ.

Capítulo 4

Planificación

Para la realización del proyecto se propusieron inicialmente una serie de pasos a seguir que se detallarán a continuación. Al tratarse de un tema complejo se evitó cualquier estimación temporal para los distintos apartados y se llevaron a cabo uno tras otro de forma secuencial.

En primer lugar se determinarían los problemas NP a tratar. Debido a la cantidad de documentación y bibliografía existentes al respecto se eligieron los problemas NP-Completos más comunes (ver apartado 5 para su posterior análisis.

En segundo lugar, se llevó a cabo un estudio intensivo del temario, con algunas pinceladas de mecánica cuántica pero haciendo especial énfasis en computación cuántica. Para ello se empleó como principal fuente de información el libro *Introduction to Classical and Quantum computing* [50]. Este libro ha proporcionado la base necesaria para comprender los algoritmos que se proponen más adelante, y, como tal, se hace referencia al mismo en numerosas ocasiones a lo largo del proyecto.

A continuación, se estudió la posibilidad de usar el tiempo como medida comparativa entre los algoritmos cuánticos y clásicos. Para ello se llevó a cabo una prueba con un circuito sencillo. La explicación de dicha prueba y los resultados pueden verse al inicio del capítulo 8. Se adelanta que el tiempo de ejecución de dicha prueba en el *backend* de IBM fue suficientemente alto (2 segundos) como para descartar la posibilidad de comparar de forma empírica algoritmos cuánticos y clásicos. Para ello se prevee que será necesario un avance tecnológico que permita a los ordenadores cuánticos trabajar a una velocidad similar a los ordenadores actuales. Debido a esto, los análisis comparativos serán mayormente teóricos, añadiendo datos de tiempo de ejecución a modo informativo.

El siguiente paso sería recopilar información a cerca de algoritmos cuánticos existentes para la resolución de los problemas previamente filtrados. Da-

dos estos algoritmos es necesario comprenderlos para determinar cuáles son susceptibles de ser implementados y ejecutados en este trabajo. Este paso consiste en numerosos pasos pequeños ya que comprender un algoritmo cuántico implica aprender conceptos y técnicas innovadoras, en su mayoría incluyendo conceptos matemáticos y cuánticos difíciles de comprender. Para alcanzar la comprensión de los algoritmos más complejos se han realizado algunas pruebas, simulaciones y ejecuciones paralelas al proyecto con valor docente pero sin relación directa con el tema que se está tratando a partir de algoritmos más sencillos de comprender e implementar. Estos experimentos, como se repetirá en el apartado 8 pueden ser accedidos en la carpeta de códigos fuente en forma de cuaderno "jupyter notebooks".

Tras este paso se determinaron dos algoritmos, el *algoritmo de Grover* (para la resolución del problema de la 3-satisfacibilidad) y el *algoritmo de estimación de fase cuántico* (para la resolución del problema del viajante de comercio) como las opciones más interesantes dentro del rango de posibles algoritmos cuánticos.

Una vez escogidos los algoritmos se deben implementar, y ejecutar. Se debe ser cuidadoso en este punto del proyecto ya que una mala implementación o lectura de los resultados puede influir drásticamente en las conclusiones del estudio.

Es recomendable, a modo de buenas prácticas, ir documentando y almacenando toda la información que se genera conforme se avanza en el proyecto para que ésta se encuentre disponible cuando se requiera hacer uso de la misma.

El paso final consiste en estructurar de forma adecuada la información y generar el documento final incluyendo un análisis de los resultados obtenidos y las conclusiones extraídas de estos resultados.

Capítulo 5

Análisis

Dentro del conjunto de problemas NP encontramos el subconjunto de problemas apodado NP-completos. Como ha sido comentado previamente, en dicho conjunto encontramos problemas a los cuales podemos reducir cualquier otro problema perteneciente a NP. Los siguientes seis problemas forman una lista que pueden ser considerada como el núcleo del conjunto NP-completo, ya que lo que se listan a continuación son los problemas que se utilizan de forma más recurrente a la hora de hacer reducciones para demostrar la NP-completitud de numerosos problemas, y otros tantos problemas pueden ser directamente transformados en alguno de estos seis problemas o en varios. [30] Como apunte, generalmente se escriben los nombres de los problemas estudiados en complejidad algorítmica en mayúsculas para denotar que se habla de un problema.

- 3-SATISFACIBILIDAD (3-SAT)
- COINCIDENCIA TRIDIMENSIONAL (3DM)
- RECUBRIMIENTO POR VÉRTICES (VC)
- CLIQUE
- CIRCUITO HAMILTONIANO (HC)
- PARTICIÓN

Existen propuestas de algoritmos cuánticos para numerosos problemas NP-completos incluidos los recién listados. Se va a realizar una breve explicación de cada problema así como del enfoque cuántico que se está utilizando actualmente para tratar de resolver cada uno de forma más eficiente.

5.1. 3-SATISFACIBILIDAD

Para explicar este problema comenzaremos por explicar el problema de satisfacibilidad booleana (SAT). Se trata de un problema que se engloba en la lógica proposicional, en que tenemos un conjunto de variables $v \in U$ que pueden ser interpretadas como verdaderas o falsas, y unas clásulas c que se obtienen como disyunción de literales (variables o sus negados). Dependiendo de la interpretación tomada para las variables estas cláusulas pueden resultar en conjunto verdaderas o falsas.

El problema SAT consiste en, partiendo de un conjunto de estas cláusulas $C = [c_1, c_2, ..., c_n]$, comprobar si existe una interpretación de U (el conjunto de variables) que hace verdaderas todas las cláusulas, en cuyo caso se dice que el conjunto es satisfacible y por tanto esa instancia del problema SAT tiene solución. Si dicha interpretación no existe, el conjunto es insatisfacible y la instancia del problema SAT no tiene solución.

El problema 3-SATISFACIBILIDAD (3-SAT) es una versión reducida del problema de satisfacibilidad booleana en el que todas las instancias tienen exactamente tres literales por cada cláusula.

Por tanto la definición formal de 3-SAT es la siguiente: Dado un conjunto de cláusulas $C = [c_1, c_2, ..., c_n]$ sobre un conjunto finito de variables U de tal manera que $|c_i| = 3, \forall i \in 1, ..., n$. ¿Existe una interpretación de U que hace verdaderas todas las cláusulas en C? [30]

Actualmente se realizan competiciones anuales para determinar los mejores algoritmos para resolver el problema de la satisfacibilidad booleana, lo que demuestra hasta cierto punto el grado de dificultad y la relevancia de este problema.

Entre las propuestas cuánticas para la resolución de este problema se encuentran:

- El uso de circuitos cuánticos, haciendo uso de la llamada puerta *Fredkin* cuyo funcionamiento se describe en el apartado 6.5.3. [29]
- Utilizando Quantum Register Machines (QRM). Los QRM son sistemas de cómputo completos basados en las máquinas de registros clásicas, y, como tal, incluyen operaciones propias como *goto* o *if X then Y*.
- Mediante Quantum UREM P systems, que son construcciones algo más complejas, similares a las máquinas de Turing que como estos se definen mediante un alfabeto y una serie de reglas semejantes a las transiciones.
- Utilizando el algoritmo de Grover para búsqueda no-estructurada.

Consiste en emplear un oráculo de fase junto a un algoritmo de amplificación de amplitud (Amplitude Amplification) para encontrar soluciones que satisfacen las cláusulas del problema. Esta propuesta es una de las que serán implementadas.

5.2. COINCIDENCIA TRIDIMENSIONAL

El problema de la COINCIDENCIA TRIDIMENSIONAL ("3DM" del inglés *3-dimensional matching*), es una generalización del clásico "problema del matrimonio estable": dados n hombres , n mujeres y una lista de las relaciones hombre-mujer que se consideran aceptables. ¿Es posible crear parejas evitando la poligamia y de forma que todos reciben una pareja de entre las listadas? De forma análoga el 3DM está compuesto de tres conjuntos W, X e Y que se corresponden con tres géneros diferentes, y una lista M que contiene las posibles combinaciones que serían aceptables para todos los implicados. Añadir un matrimonio no contenido en M implica una solución no válida para el problema. Por tanto, cada tripleta perteneciente a M se corresponde con un matrimonio tripartito aceptable. Mientras que 3DM es NP-completo el problema original del "matrimonio estable" puede ser resuelto en tiempo polinómico [30].

Hay algunos artículos recientes sobre algoritmos empleando teoría de grafos para resolver este problema y la relación que existe entre ambos. Sin embargo, no parece haber recibido atención desde el ámbito de la computación cuántica, por lo que no nos centraremos en este problema.

5.3. VC y CLIQUE

Estos dos problemas así como el problema del CIRCUITO HAMILTONIANO son problemas de grafos. Un grafo G es un conjunto de objetos a los que llamaremos *vértices* V que pueden estar conectados entre sí mediante lo que llamamos *aristas*. Una arista siempre conecta dos vértices $u, v \in V$, y para un determinado grafo, la existencia de todas las aristas que lo forman viene dada en un conjunto que llamaremos E. Por tanto todo grafo se determina como un conjunto de vértices y aristas que las unen $G = \{V, E\}$.

Para explicar el problema del recubrimiento de vértices ("VC" del inglés *vertex cover*) así como el del clique es aconsejable dar primero una definición del concepto llamado conjunto independiente.

Definición 5.1 (Conjunto Independiente) . *Un conjunto independiente de un grafo $G = \{V, E\}$ es un subconjunto $V' \subseteq V$ tal que, para todo $u, v \in V'$, la arista $\{u, v\}$ no se encuentra en E.*

De forma parecida, un recubrimiento por vértices es un subconjunto de vértices $V' \subseteq V$ de un grafo G de forma que toda arista perteneciente a E incide en al menos un vértice de V'.

Y finalmente un clique consiste en un subconjunto de vértices $V' \subseteq V$ tal que, para toda pareja de vértices $u, v \in V'$, la arista $\{u, v\}$ que los une *si* se encuentra en E.

Debido a la semejanza de estos tres problemas, partiendo de la definición formal de conjunto independiente podemos definir los otros dos problemas que no son más que una transformación del mismo.

Teorema 5.1 *Para cualquier grafo $G = (V, E)$ y subconjunto $V' \subseteq V$, las siguientes afirmaciones son equivalentes:*

(a) V' es un recubrimiento por vértices de G

(b) $V \setminus V'$ es un conjunto independiente de G

(c) $V \setminus V'$ es un clique en el complemento G^c de G, donde $G^c = (V, E^c)$ con $E^c = \{\{u, v\} : u, v \in V \text{ y } \{u, v\} \notin E\}$

Estos problemas tienen actualmente numerosos algorítmos heurísticos que tratan de obtener soluciones cercanas a la óptima, debido a la dificultad de encontrar la solución óptima global.[53]

Entre estos encontramos el algoritmo de enfriamiento simulado [24], algoritmo genético [15], algoritmo de ADN [22], [52], algoritmo de optimización por enjambre de partículas [36], [2], [10], SMA metaheurístico [38], algoritmo de balance de corriente activa [27], algoritmo voraz [51], [8], algoritmo de optimización jerárquica Bayesiana [35], algoritmo de optimización de reacción química [23], etc.

No obstante, en el ámbito de las soluciones aproximadas encontramos el algoritmo cuántico apodado 'Quantum approximate optimization algorithm' (QAOA). Este algoritmo presenta una aceleración exponencial frente a los algoritmos clásicos especialmente para la resolución de problemas de optimización combinatoria [53]. Este algoritmo, propuesto por Edward Fache en 2014, es un algoritmo heurístico que hibrida computación clásica y cuántica para resolver los ya mencionados problemas de optimización combinatoria [20].

La aplicación de este algoritmo a lo largo del proceso de resolución del problema se entremezcla con una versión clásica, e implica conocer excepcionalmente bien los métodos ya existentes para el cálculo del corte máximo [13] y de la cobertura exacta [11]. La solución al problema mediante QAOA está detallada en [53] junto a pseudocódigo, diagramas de flujo, justificaciones matemáticas y esquemas de los circuitos cuánticos. Además puede

obtenerse el código original escrito en Python mediante la librería pyQPanda a través de un enlace a la plataforma GitHub.

5.4. CIRCUITO HAMILTONIANO

Para explicar el problema del circuito o ciclo hamiltoniano debemos comenzar por dar una definición de circuito simple.

Definición 5.2 (Circuito simple) . *Un circuito simple, o ciclo, en un grafo* $G = (V, E)$ *es una secuencia* $< v_1, v_2, ..., v_k >$ *de vértices diferentes pertenecientes a* V *tal que* $\{v_i, v_{i+1}\} \in E$ *para* $1 \leq i < k$ *y también* $\{v_k, v_1\} \in E$

Dada la definición de circuito simple se obtiene la de ciclo hamiltoniano como sigue:

Definición 5.3 (Ciclo Hamiltoniano) . *Un ciclo hamiltoniano en un grafo* $G = (V, E)$ *es un circuito simple que contiene todos los vértices de* G.

El problema del CICLO HAMILTONIANO se define como sigue: Dado un grafo $G = (V, E)$, ¿Contiene G un ciclo hamiltoniano? Existe una conversión sencilla en tiempo polinómico entre el problema del CICLO HAMILTONIANO y el VIAJANTE DE COMERCIO (TSP) que se describe haciendo corresponder a cada vértice en V una ciudad en C (conjunto de ciudades del problema del VIAJANTE DE COMERCIO), y para cada dos ciudades, si existe una arista que las une en E, el coste asociado para el viajante es 1, si no están conectadas, el coste es 2. De tal manera, si establecemos el coste máximo del viaje en $m : |V| = m$, la solución al TSP será también solución del ciclo hamiltoniano. [30]

Las soluciones propuestas para el viajante de comercio son numerosas. Aún hoy en día, una búsqueda rápida en google scholar muestra más de 6.900 resultados, incluyendo artículos con propuestas de soluciones eficientes, catálogos de algoritmos, aplicaciones, etc...

De entre tantas soluciones no son pocas las que exploran el uso de la computación cuántica para la resolución de este popular problema. De entre ellos destacan los algoritmos adiabáticos y estimación de fase. Este último presenta una propuesta muy original empleando los eigenvalues, o autovalores, para determinar los costes de las diferentes rutas de una manera muy eficiente. [42]

5.5. PARTICIÓN

Finalmente tenemos el problema de la PARTICIÓN (en inglés PARTITION). Este problema es comúnmente usado para demostrar la NP-completitud de otros problemas que involucran parámetros numéricos, como distancias, pesos, costes, capacidades y otros. El problema PARTICIÓN se formaliza de la siguiente manera: Dado un conjunto finito A y un "tamaño" $s(a) \in Z^+$ para todo $a \in A$, encontrar un subconjunto $A' \subseteq A$ de forma que se cumple [30]

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a) \tag{5.1}$$

.

De manera informal el problema consistiría en decidir si es posible dividir el conjunto en dos partes de forma que la suma de los elementos en ambas sea igual.

Para este problema destaca el algoritmo de optimización de enfriamiento cuántico (adiabatic quantum optimization), el cual aparece como una de las herramientas más populares a la hora de resolver de forma eficiente problemas NP-completos empleando la computación cuántica. Este problema concluye el apartado de análisis y da pie al apartado de diseño, donde se explicarán los conocimientos teóricos necesarios para comprender y diseñar circuitos cuánticos.

Capítulo 6

Diseño

Para diseñar los circuitos cuánticos capaces de resolver nuestros problemas deberemos hacer uso de componentes exclusivos de la computación cuántica. Estos componentes se apoyan principalmente del Álgebra lineal y su función es almacenar o procesar la información con la que estemos trabajando.

6.1. Cúbits, superposición y medición

El primer componente del que vamos a hablar son los cúbits, que definimos a continuación.

Definición 6.1 (Cúbit) . *Un bit cuántico o cúbit (en inglés qubit o quantum bit) es una unidad de información [40]. Un cúbit puede encontrarse en los estados* 0, 1 *o cualquier superposición de éstos.*

Siguiendo la *notación de Dirac* escribimos estos valores entre una línea vertical y un ángulo, a esta combinación la llamamos *ket*: $|0\rangle, |1\rangle$.

Si tuvieramos un bit clásico $|0\rangle$ y $|1\rangle$ serían los únicos estados, sin embargo, las leyes de la mecánica cuántica permiten a un cúbit encontrarse en una combinación de estos estados llamado *superposición.* Una superposición es una combinación lineal de los estados $|0\rangle$ y $|1\rangle$ de forma que la suma de las norma al cuadrado de los coeficientes de ambos sumen 1. Por tanto, dado un estado en superposición cualquiera $|\psi\rangle$, lo escribiremos como:

$$|\psi\rangle = \alpha|0\rangle + |\beta|1\rangle. \tag{6.1}$$

Esta forma de representar los estados será de gran relevancia cuando analicemos a continuación la relación entre los coeficientes que forman la superposición y las probabilidades asociadas a los estados base.

A menudo se visualizan los dos valores $|0\rangle$ y $|1\rangle$ del cúbit como los polos norte y sur de una esfera de radio 1 llamada *esfera de Bloch*, y, los valores que surgen como combinación lineal de estos, o estados en superposición, forman la totalidad de puntos posibles en la superficie de la esfera.

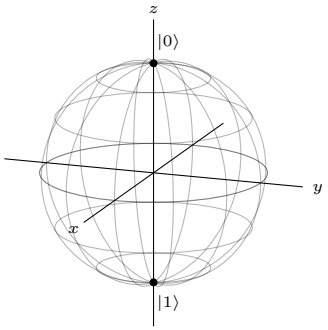


Figura 6.1: Esfera de Bloch con los estados $|0\rangle$ y $|1\rangle$

Ya que un cúbit puede encontrarse en superposición, los coeficientes asociados a los estado $|0\rangle$ y $|1\rangle$ determinan cuán "cercano" es el estado a uno de estos estados base.

Ejemplo 6.1 *Un estado en superposición con la misma cantidad de* $|0\rangle$ *y de* $|1\rangle$ *sería el estado:*

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{6.2}$$

La $\sqrt{2}$ *en el denominador es necesaria para mantener el módulo 1.*

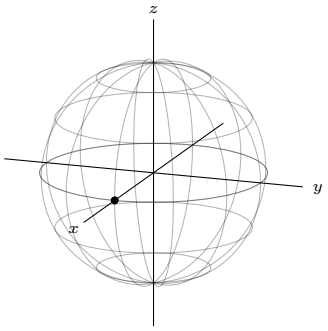


Figura 6.2: Esfera de Bloch con el estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

Existen infinitos estados con la misma proporción de los estados $|0\rangle$ y $|1\rangle$, cada uno de ellos representado como un punto en el ecuador de la esfera. La diferencia entre estos es la llamada *fase relativa* del estado, la cual puede modificarse alterando el valor de la parte imaginaria de una de las componentes.

Cuatro estados que se emplean de forma habitual y que presentan estas características son:

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \\ |i\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \\ |-i\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle). \end{aligned} \tag{6.3}$$

Se puede intuir rápidamente que no todos los estados deben encontrarse necesariamente en un polo o en el ecuador de la esfera de bloch. Un estado puede encontrarse en cualquier punto de dicha esfera, dando lugar a estados que amplifiquen las probabilidades de uno de los dos estados base de un cúbit.[50]

Estos estados se representan mediante coeficientes en forma de números complejos asociados a dos estados que formen una base. Normalmente se emplean los estados $|0\rangle$ y $|1\rangle$, pero también forman una base $|+\rangle$ y $|-\rangle$, $|i\rangle$ y $|-i\rangle$ o cualquier pareja de estados que sean extremos opuestos en la esfera de Bloch.

Ejemplo 6.2 *Siguiendo esto, un ejemplo de estado en superposición podría ser el estado*

$$\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle. \tag{6.4}$$

Y un estado medido en una base distinta a la formada por los estados $|0\rangle$ *y* $|1\rangle$ *podríamos extraerlo de la propia definición de los estados* $|+\rangle$ *y* $|-\rangle$*.*

$$|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \tag{6.5}$$

A parte de la ya mencionada fase relativa existe la *fase global*, que consiste en una determinada fase $e^{i\theta}$ aplicada a ambas componentes de un estado en superposición. De forma que un estado con fase global sería de la forma:

$$|\psi\rangle = e^{i\theta}(\alpha|0\rangle + \beta|1\rangle). \tag{6.6}$$

Ya que, a priori, α y β son números complejos, podemos escribirlos en forma polar como:

$$z = re^{i\theta}, \tag{6.7}$$

lo que nos permite extraer factor común la fase de, por ejemplo, el coeficiente α y añadirlo a la fase global, quedando así un estado en que α es un número real y positivo, y β puede ser complejo. Esto realmente no cambia nada más que la forma de expresar los estados pero a continuación veremos una cualidad de las fases globales que harán de esta nomenclatura un sistema de normalización que facilitará mucho la lectura de estados, y que emplearemos durante todo el proyecto.

Los estados en superposición colapsan en el momento en el que se realiza una medición sobre el cúbit, con una cierta probabilidad como observamos en el ejemplo 6.3. Este fenómeno, comentado debido a sus consecuencias en el experimento de la doble rendija y banalizado mediante el juego mental del "gato de Shrödinger" trata de explicar la interacción entre el propio observador y los estados cuánticos, el hecho de que la medición de un sistema debe colapsar en un valor propio del operador que se emplea para medir el sistema..[3]

En el caso de los cúbits esta medición se realiza habitualmente sobre el eje Z de la mostrada esfera de Bloch, y el valor resultante de la medición debe ser bien un 0, bien un 1, como si de un bit clásico se tratara. Sin emargo, no debemos olvidar que el estado en que se encuentra el cúbit viene definido mediante una superposición de dos estados, y la probabilidad de medir uno u otro estado se puede calcular mediante la norma al cuadrado de cada amplitud. [50]

Ejemplo 6.3 *Tenemos un cúbit en el estado* $\frac{1+i\sqrt{3}}{3}|0\rangle + \frac{2-i}{3}|1\rangle$*. Calculamos las probabilidades de obtener* $|0\rangle$ *o* $|1\rangle$ *cuando el cúbit sea medido. Esto se realiza mediante la norma al cuadrado de las amplitudes.*

La amplitud de $|0\rangle$ *es* $\frac{1+i\sqrt{3}}{3}$*, por tanto:*

$$P(|0\rangle) = \left|\frac{1+i\sqrt{3}}{3}\right|^2 = \frac{1+i\sqrt{3}}{3}\frac{1-i\sqrt{3}}{3} = \frac{1+i\sqrt{3}-i\sqrt{3}-3i^2}{9} = \frac{4}{9} \quad (6.8)$$

La amplitud de $|1\rangle$ *es* $\frac{2-i}{3}$*, por tanto:*

$$P(|1\rangle) = \left|\frac{2-i}{3}\right|^2 = \frac{2-i}{3}\frac{2+i}{3} = \frac{4-i+i-i^2}{9} = \frac{5}{9} \quad (6.9)$$

Observamos cómo nuestro cúbit tiene más probabilidades de colapsar en el estado $|1\rangle$ *que en el estado* $|0\rangle$*, y, por la aparición de la parte compleja en las amplitudes, podemos asumir también que tendrá una determinada fase relativa.*

Volviendo al tema de la fase global, calculamos ahora la probabilidad de

este mismo estado aplicándole una fase global $e^{i\theta}$ y obtenemos lo siguiente:

$$\begin{aligned} |\psi\rangle &= e^{i\theta}(\frac{1+i\sqrt{3}}{3}|0\rangle + \frac{2-i}{3}|1\rangle) \\ P(|0\rangle) &= \left|e^{i\theta}\frac{1+i\sqrt{3}}{3}\right|^2 = (e^{i\theta}e^{-i\theta})\frac{1+i\sqrt{3}}{3}\frac{1-i\sqrt{3}}{3} \\ &= (1)\frac{1+i\sqrt{3}-i\sqrt{3}-3i^2}{9} = \frac{4}{9}. \end{aligned} \tag{6.10}$$

Como se ha podido comprobar la fase global es físicamente irrelevante, ya que no altera las probabilidades de medir los estados que formen la base en que se está midiendo.[50] Emplearemos este hecho y el desglose de un estado en superposición explicado con anterioridad en que podemos tomar de coeficientes un número real y un número complejo para explicar la normalización de estados.

Un estado se dice que está *normalizado* si su probabilidad total es 1, como debe serlo. En ocasiones requerimos de una *constante de normalización* para obtener esto. Supongamos por ejemplo un cúbit en el estado

$$A(\sqrt{2}|0\rangle + i|1\rangle). \tag{6.11}$$

Debemos encontrar el valor de A que asegura que la probabilidad total sea 1. Para ello calculamos [50]

$$\begin{aligned} 1 &= (A\sqrt{2})(A\sqrt{2}) * + (Ai)(Ai) * \\ &= 2\,|A|^2 + |A|^2 \\ &= 3\,|A|^2 \\ |A|^2 &= \frac{1}{3}. \end{aligned} \tag{6.12}$$

De entre las posibles soluciones a la última igualdad, podemos tomar la solución real

$$A = \frac{1}{\sqrt{3}} \tag{6.13}$$

gracias a que la fase global no afecta al cálculo de probabilidades. Por tanto el estado normalizado sería

$$\frac{1}{\sqrt{3}}(\sqrt{2}|0\rangle + i|1\rangle). \tag{6.14}$$

La utilización de esta normalización va más allá de la claridad visual. Empleando esto podemos aprender a situar un estado cualquiera en la esfera de Bloch. Vamos a ver paso a paso como, a partir de un estado cualquiera,

podemos emplear la normalización que acabamos de ver para situar el estado en la esfera de Bloch.

Ejemplo 6.4 *Comenzamos el ejemplo con un cúbit en un estado inicial*

$$\frac{3+i\sqrt{3}}{4}|0\rangle - \frac{1}{2}|1\rangle. \tag{6.15}$$

En primer lugar se observa que el coeficiente del estado $|0\rangle$ *es complejo por lo que se procede a convertir en real siguiendo las pautas de normalización dadas. Para ello comenzamos convirtiendo el número complejo a forma polar. Ya que* $(3+i\sqrt{3})/4 = (\sqrt{3}/2)e^{i\pi/6}$*, el estado se puede escribir como*

$$\frac{\sqrt{3}}{2}e^{i\pi/6}|0\rangle - \frac{1}{2}|1\rangle. \tag{6.16}$$

Si sacamos factor común $e^{i\pi/6}$ *tenemos:*

$$e^{i\pi/6}\Big(\frac{\sqrt{3}}{2}|0\rangle - e^{-i\pi/6}\frac{1}{2}|1\rangle\Big) \equiv \frac{\sqrt{3}}{2}|0\rangle - e^{-i\pi/6}\frac{1}{2}|1\rangle. \tag{6.17}$$

La equivalencia viene dada por la demostración previa de que las fases globales no afectan a los estados. Para cambiar el signo negativo por un signo positivo podemos emplear la igualdad $e^{i\pi} = -1$*:*

$$\frac{\sqrt{3}}{2}|0\rangle + e^{i\pi}e^{-i\pi/6}\frac{1}{2}|1\rangle = \frac{\sqrt{3}}{2}|0\rangle + e^{i5\pi/6}\frac{1}{2}|1\rangle. \tag{6.18}$$

A continuación, ya que α *es real y* β *es complejo, podemos parametrizar un estado cualquiera* $|\psi\rangle$ *como*

$$\alpha = \cos\Big(\frac{\theta}{2}\Big), \qquad \beta = e^{i\phi}\sin\Big(\frac{\theta}{2}\Big) \tag{6.19}$$

ya que las operaciones seno y coseno mantienen la normalización de probabilidad total en 1*. Si comparamos* α *y* β *con nuestro estado obtenemos que*

$$\cos\Big(\frac{\theta}{2}\Big) = \frac{\sqrt{3}}{2}, \qquad e^{i\phi} = e^{i5\pi/6}, \qquad \sin\Big(\frac{\theta}{2}\Big) = \frac{1}{2}. \tag{6.20}$$

Resolviendo las ecuaciones primera o última para θ *mediante el arcocoseno y el arcoseno y resolviendo la segunda ecuación para* ϕ *tenemos:*

$$\theta = \frac{\pi}{3}, \qquad \phi = \frac{5\pi}{6}. \tag{6.21}$$

O lo que es lo mismo, podemos expresar nuestro estado de la forma

$$\cos\Big(\frac{\pi/3}{2}\Big)|0\rangle + e^{i5\pi/6}\sin\Big(\frac{\pi/3}{2}\Big)|1\rangle. \tag{6.22}$$

Teniendo estos dos ángulos podemos representar cualquier estado sobre la esfera de Bloch siendo θ el llamado ángulo polar y mide el ángulo respecto al polo norte hacia abajo, y ϕ el llamado acimud, que mide el ángulo desde del eje x hasta el plano xy como puede verse en la figura 6.3.[50]

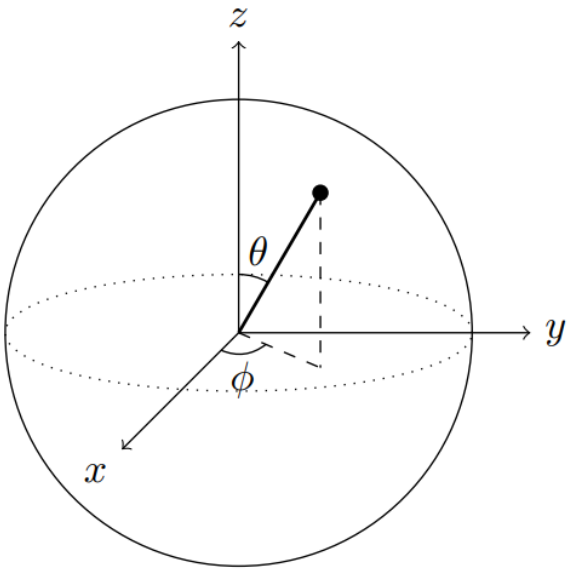
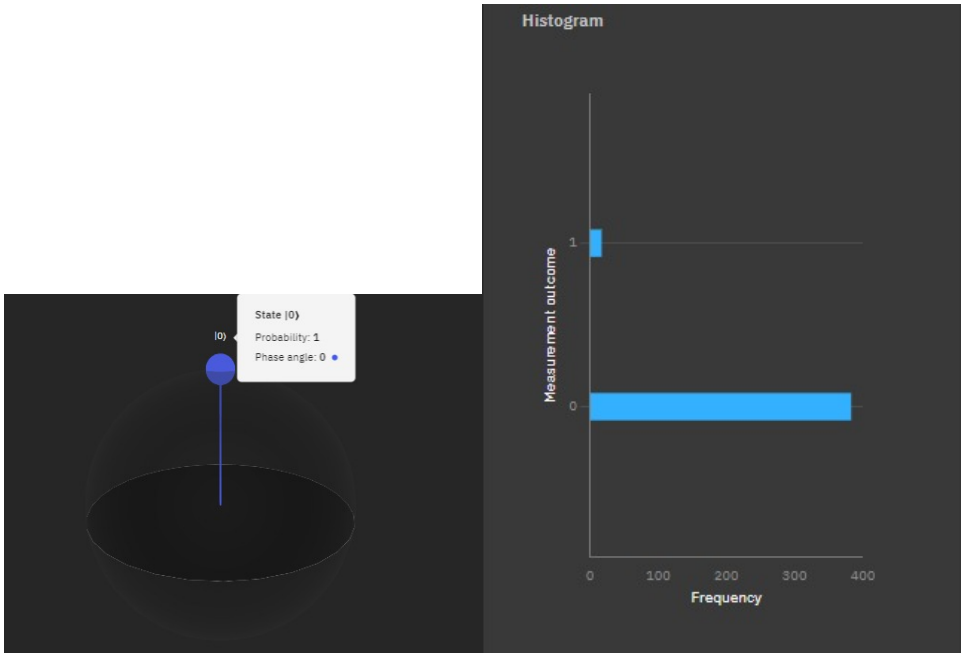


Figura 6.3: Representación de los ángulos θ y ϕ en la esfera de Bloch. Imágen extraída de [50].

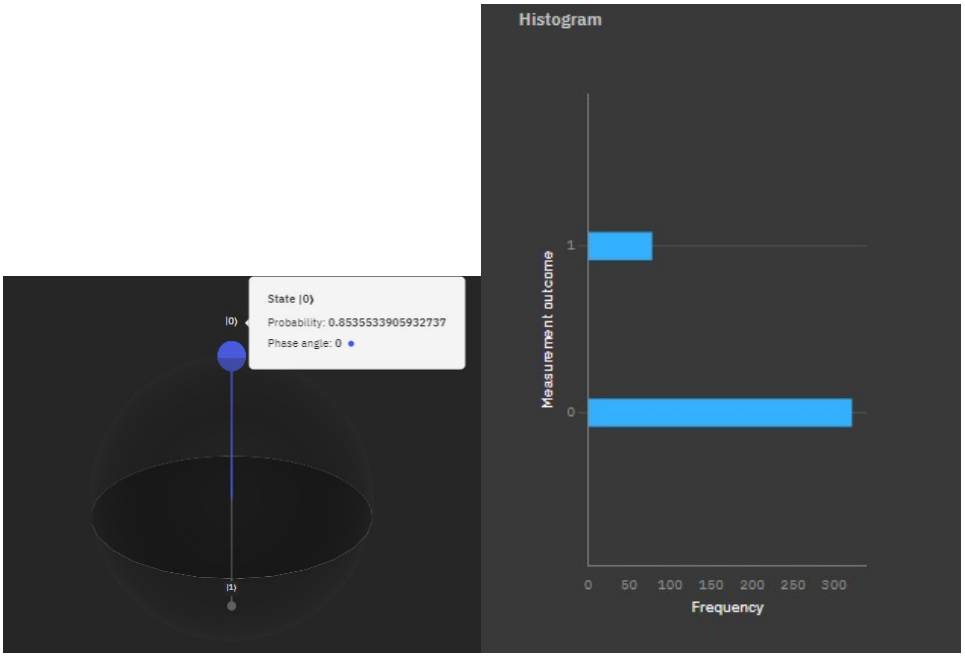
A continuación se muestra un pequeño experimento que demuestra lo anteriormente explicado sobre mediciones. En la figura 6.4 se observa el estado $|0\rangle$ totalmente puro, mientras que en la figura 6.5 se trata de un estado en superposición que se encontraría a medio camino en el meridiano que atraviesa los ejes z e y, debido a una rotación de $\frac{\pi}{4}$ radianes sobre el eje x.



(a) Cúbit inicializado **(b)** Resultados de la ejecución

Figura 6.4: Cúbit en el estado $|0\rangle$

Nótese en la figura 6.4b como las veces en las que el resultado de la ejecución ha devuelto 1 no son inexistentes. Al trabajar con ordenadores cuánticos reales estos se encuentran sujetos a unos porcentajes de error muy superiores a los que podríamos encontrar en cualquier ordenador clásico, debido a la dificultad de realizar las operaciones a nivel físico.



(a) Cúbit inicializado **(b)** Resultados de la ejecución

Figura 6.5: Cúbit al que se le ha aplicado una rotación de $\frac{\pi}{4}$ sobre el eje x

Vemos en la figura 6.5b mediciones del valor 1 con una mayor frecuencia que en el ejemplo mostrado en la figura 6.4. Mientras que el cúbit se encuentra en un estado en superposición, al realizar la medición sólo podemos obtener los valores 0 y 1. Las mediciones que realizaremos en este trabajo serán equivalentes a realizar una operación que provoque el colapso del estado cuántico en un autovalor del eje Z, dando lugar a los valores binarios clásicos $|0\rangle$ y $|1\rangle$. Aunque, como hemos comentado previamente, un cúbit puede medirse en cualquier base que queramos definir, y, evidentemente, dependiendo de la base escogida y el estado del cúbit, las probabilidades de obtener como resultado de la medición uno u otro estado de los que forman nuestra base serán diferentes.

6.2. Álgebra lineal y ortonormalidad

Empleando álgebra lineal podemos expresar los estados de un cúbit mediante matrices con algunas propiedades de gran interés para la computación. En primer lugar, una representación de estado más sencilla de almacenar y operar para un ordenador clásico sería la representación matricial en forma de *vector columna*.

Definición 6.2 (Vector columna) . *Un vector columna es una matriz que únicamente posee una columna. La traspuesta de un vector columna es un vector fila, un tipo de matriz formada por una sola fila.*

Empleando vectores columna podemos representar un estado dado $|\psi\rangle$ como combinación lineal de amplitudes en una base determinada.

Ejemplo 6.5 *Un estado $|\psi\rangle$ en forma de vector columna.*

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\0\end{pmatrix} + \begin{pmatrix}0\\\beta\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix}. \tag{6.23}$$

Generalmente, en computación cuántica, además de la forma en vector columna emplearemos la representación en forma de vector fila. En esta representación sin embargo no utilizaremos la traspuesta directa del estado original, emplearemos la *traspuesta conjugada*, que se obtiene tomando el conjugado complejo de cada elemento en la matriz traspuesta. Se suele denotar mediante el símbolo † (apodado "dagger" en inglés)[50][49]:

$$\begin{pmatrix}\alpha\\\beta\end{pmatrix}^{\dagger} = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}. \tag{6.24}$$

Cuando empleamos esta forma en computación cuántica, siguiendo la notación *Bra-Ket* denotaremos el estado conjugado traspuesto mediante un ángulo y una barra vertical apodado *bra*:

$$\langle\psi| = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}. \tag{6.25}$$

Puede verse la semejanza respecto al ya explicado *ket* que es el sistema que hemos estado empleando para denotar los estados más comunes. Cambiar entre estas dos representaciones se le llama "tomar el dual". De esta forma el dual de un *Bra* es el *Ket* resultante de calcular la conjugada traspuesta y viceversa. [50] Esto se emplea con una finalidad concreta llamado *producto interior*.

Definición 6.3 (Producto interior) . *El producto interior de dos estados, en computación cuántica, se define como la operación de multiplicar* $\langle\psi|$ *por* $|\phi\rangle$*:*

$$\langle\psi||\phi\rangle = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}\begin{pmatrix}\gamma \\ \delta\end{pmatrix} \tag{6.26}$$

Generalmente se unen las barras verticales del centro dejando la expresión del producto interior como $\langle\psi|\phi\rangle$.

Teorema 6.1 *Dos estados son ortonormales si su producto interior es igual a* 0 *y cada estado se encuentra individualmente normalizado (el producto interior consigo mismo es igual a* 1*).*

Conectando con lo que se explicó con anterioridad sobre estados opuestos en la esfera de Bloch, dos estados cualesquiera opuestos en la esfera son ortonormales y forman una base sobre la que podemos realizar mediciones.

Podemos calcular las amplitudes de un estado $|\psi\rangle$ aplicando el producto interior con respecto a la base que se quieren obtener los resultados. Así, si queremos calcular las amplitudes de $|\psi\rangle$ en la base $\{|+\rangle, |-\rangle\}$ calcularíamos $\langle+|\psi\rangle$ y $\langle-|\psi\rangle$ respectivamente. Computando la norma al cuadrado de dichas amplitudes tendríamos las probabilidades de medir cada uno de los dos estados que forman la base, tal y como se estudió anteriormente.[50]

De manera análoga a como hemos definido el producto interior se define el *producto exterior*.

Definición 6.4 (Producto exterior) . *El producto exterior de dos estados, en computación cuántica, se define como la operación de multiplicar* $|\psi\rangle$ *por* $\langle\phi|$*:*

$$|\psi\rangle\langle\phi| = \begin{pmatrix}\alpha \\ \beta\end{pmatrix}\begin{pmatrix}\gamma^* & \delta^*\end{pmatrix} = \left(\begin{pmatrix}\alpha \\ \beta\end{pmatrix}\gamma^*\begin{pmatrix}\alpha \\ \beta\end{pmatrix}\delta^*\right) = \left(\begin{pmatrix}\alpha\gamma^* & \alpha\delta^* \\ \beta\gamma^* & \beta\delta^*\end{pmatrix}\right) \tag{6.27}$$

Como se puede ver en la ecuación 6.27, el resultado del producto exterior es una matriz. Estos productos escalares podremos emplearlos para construir puertas cuánticas, que, como veremos a continuación, tienen una estrecha relación con las matrices.

6.3. Puertas cuánticas para 1 cúbit

Una puerta cuántica, o "quantum gate", transforma el estado de un cúbit a otro estado. Actúan de una manera similar a las puertas lógicas en la electrónica clásica. Como tal, la transformación que produce se puede denotar de diferentes maneras.

Generalmente, se emplean matrices unitarias para describir su comportamiento. El hecho de que se empleen matrices unitarias nos da mucha información sobre las características que deben tener estos operadores, pero es necesario tener algunas nociones iniciales de álgebra para comprenderlas.

Definición 6.5 (Matriz unitaria) . *Una matriz U es unitaria si, y sólo si, su inversa es igual al conjugado de su traspuesta. Es decir, $U^{-1} = U^{\dagger}$*

Además, del hecho de que U sea unitaria, derivan las siguientes afirmaciones:

- 1. $U^{\dagger}$ es unitaria.
- 2. U^{-1} es unitaria.
- 3. $U^{-1} = U^{\dagger}$ (que es el criterio para ser U unitaria).
- 4. $U^{\dagger}U = \mathbb{I}$.
- 5. $|det(U)| = 1$.
- 6. Las columnas de U forman un conjunto de vectores ortonormal.
- 7. Para una columna dada, $\sum_{i=1}^{2^n} |U_{ij}|^2 = 1$.
- 8. Para una fila dada, $\sum_{j=1}^{2^n} |U_{ij}|^2 = 1$.

De estas características, las más relevantes para la computación cuántica, y, concretamente, para las puertas cuánticas son las siguientes tres. En primer lugar, ya que $U^{\dagger}U = \mathbb{I}$, tenemos garantía de que siempre podemos deshacer el cambio provocado por una puerta cuántica (son lógicamente reversibles), lo cual será de gran utilidad más adelante en algoritmos como el *algoritmo cuántico de estimación de fase*. En segundo lugar, las propiedades 7 y 8, nos garantizan que si partimos de un estado normalizado (con probabilidad total igual a uno), tras aplicar una puerta cuántica el resultado será un estado normalizado también. Finalmente, ya que es la magnitud $|det(U)|$ lo que está restringido a la unidad, implica que la restricción sobre el determinante se puede satisfacer con $det(U) = \pm 1$ o $\pm i$. Lo que nos indica que los elementos de una matriz unitaria pueden ser elementos complejos[48],

que, como hemos visto anteriormente, aparecen recurrentemente tratando con computación cuantica.

Podemos por tanto exponer el siguiente teorema sobre las puertas cuánticas.

Teorema 6.2 *Toda puerta cuántica es reversible y puede ser descrita mediante una matriz unitaria, y toda matriz unitaria representa una puerta cuántica. Al ser unitarias se demuestra por definición que al actuar con dichas matrices sobre los estados, se mantienen dichos estados normalizados. [50]*

A continuación vamos a ver algunas de las puertas cuánticas más habituales, su definición, y como utilizarlas empleando Qiskit. Comenzando con puertas que actúan sobre un único cúbit, tras lo cual analizaremos puertas que actúan mediante dos cúbits y finalmente explicaremos brevemente el concepto de puertas cuánticas universales junto a las puertas cuánticas que actúan sobre tres cúbits.

Al final del apartado explicaremos también lo que es un oráculo ya que su esencia es casi idéntica a una puerta cuántica y su utilización es clave en el diseño de circuitos cuánticos.

6.3.1. Puerta identidad

La puerta identidad es una puerta que actúa sobre un cúbit dejándolo en su mismo estado, a todos los efectos, no produce ningún cambio. [50] La matriz que representa su funcionamiento es

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{6.28}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.id(qbit)
```

Partiendo de un único registro cuántico q inicializado en el estado $|0\rangle$ obtenemos el mismo estado como se observa en la figura 6.6.

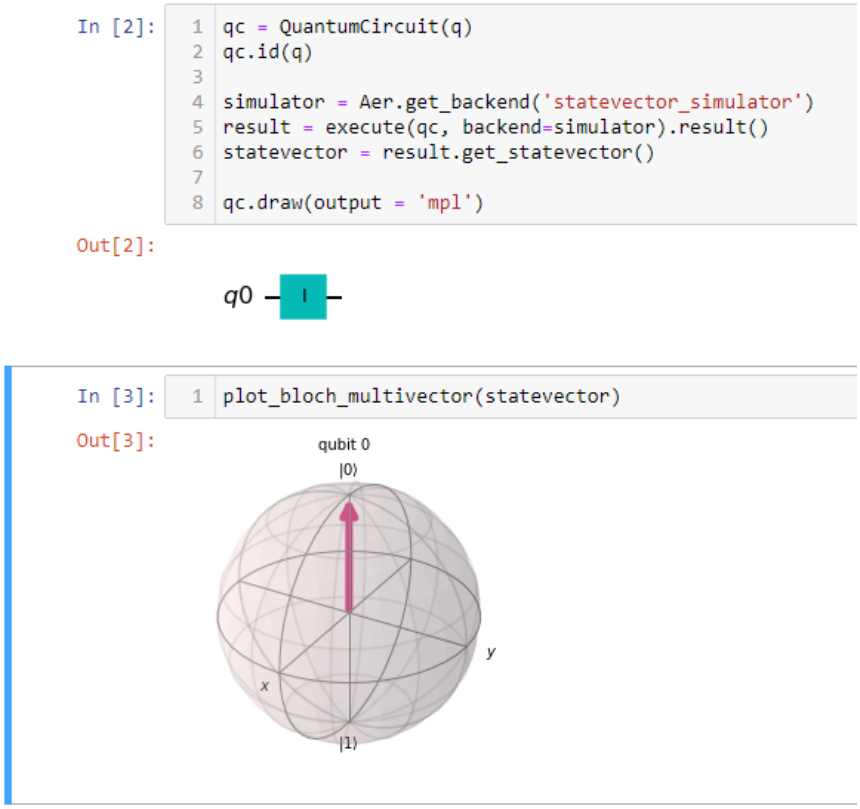


Figura 6.6: Resultado de aplicar la puerta ID a un cúbit inicializado en $|0\rangle$

6.3.2. Puertas Pauli

Una puerta Pauli es semejante a la puerta lógica *NOT*. Sin embargo, existen tres puertas de Pauli diferentes, cada una realizando una operación de negación diferente.

Puerta Pauli-X

La puerta Pauli-X, o, *bit-flip gate*, cambia el estado $|0\rangle$ por $|1\rangle$ y viceversa. Se puede representar en una esfera de Bloch como una rotación de 180° sobre el eje x. [50] La matriz que representa su funcionamiento es

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{6.29}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
1 QuantumCircuit.x(qbit)
```

Partiendo de un único registro cuántico q inicializado en el estado $|0\rangle$ obtenemos el estado $|1\rangle$ tal y como se muestra en la figura 6.7. Esta rotación también nos permitiría intercambiar los estados $|i\rangle$ y $|-i\rangle$.

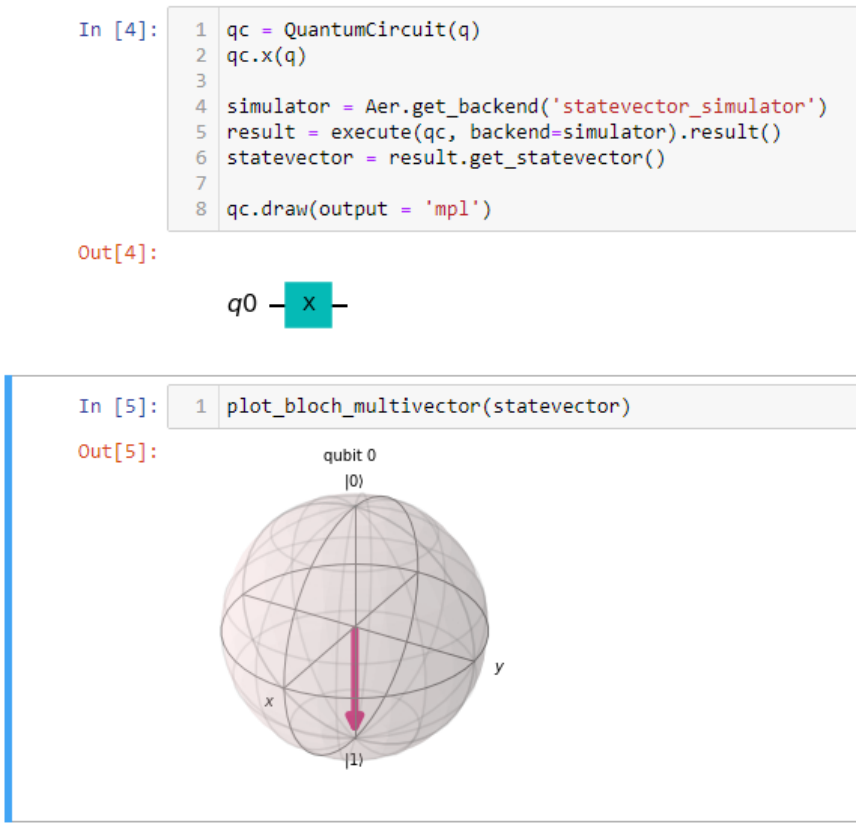


Figura 6.7: Aplicación de la puerta Pauli-X a un cúbit inicializado en el estado $|0\rangle$

Puerta Pauli-Y

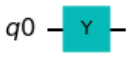
La puerta Pauli-Y, o, *bit and phase-flip gate*, cambia el estado $|0\rangle$ por $i|1\rangle$ y $|1\rangle$ por $-i|0\rangle$. Se puede representar en una esfera de Bloch como una rotación de 180° sobre el eje y. [50] La matriz que representa su funcionamiento es

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{6.30}$$

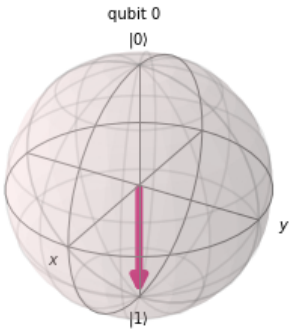
Puede ser utilizado en Qiskit mediante la función [44]

```
1 QuantumCircuit.y(qbit)
```

Esta rotación provoca que no sólamente alteremos el valor de estado del cúbit (ver figura 6.8a), también la *fase relativa* del estado se ve invertida (ver figura 6.8b) permitiendo pasar del estado $|+\rangle$ a $|-\rangle$ y al contrario.




```
1 plot_bloch_multivector(statevector)
```




```
1 backend = BasicAer.get_backend('unitary_simulator')
2 job = backend.run(transpile(qc, backend))
3 job.result().get_unitary(qc, decimals=3)
```

```
array([[ 0.+0.j, -0.-1.j],
       [ 0.+1.j,  0.+0.j]])
```

(a) Circuito y resultado en esfera de Bloch. **(b)** Resultado en forma de matriz unitaria.

Figura 6.8: Aplicación de la puerta Pauli-Y a un cúbit inicializado en el estado $|0\rangle$.

Puerta Pauli-Z

La puerta Pauli-Z, o, *phase-flip gate*, mantiene el estado $|0\rangle$ pero transforma $|1\rangle$ en $-|1\rangle$. Se puede representar en la esfera de Bloch como una rotación de 180° sobre el eje z. [50]

La matriz que representa su funcionamiento es

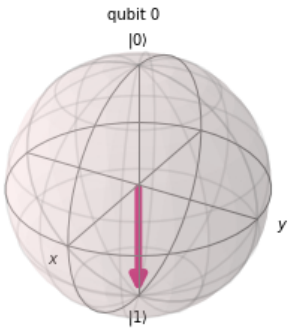
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{6.31}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.z(qbit)
```

Esta rotación invierte la fase del cúbit, permitiendo pasar del estado $|1\rangle$ a $-|1\rangle$. Para ver este efecto podemos inicializar un cúbit en el estado $|1\rangle$ aplicándole una puerta Pauli-X y posteriormente aplicando la puerta Pauli-Z.


```
plot_bloch_multivector(statevector)
```




```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(qc, backend))
job.result().get_unitary(qc, decimals=3)
```

```
array([[ 0.+0.j,  1.+0.j],
       [-1.-0.j,  0.+0.j]])
```

(a) Circuito y resultado en esfera de Bloch. **(b)** Resultado en forma de matriz unitaria.

Figura 6.9: Aplicación de la puerta Pauli-Z a un cúbit inicializado en el estado $|1\rangle$.

6.3.3. Puertas S y S†

La puerta S es una raíz cuadrada de la puerta Pauli-Z explicada anteriormente. Realiza una rotación del estado del cúbit de 90° sobre el eje Z, y, por tanto, aplicarla dos veces consecutivas es equivalente a aplicar la puerta Pauli-Z. La puerta S† es su conjugada, y realiza la misma operación pero rotando −90°. [50]

La matrices que representan sus respectivos funcionamientos son

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i*\frac{\pi}{2}} \end{pmatrix} \qquad S^{\dagger} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i*\frac{\pi}{2}} \end{pmatrix} \tag{6.32}$$

Pueden ser utilizadas en Qiskit mediante las funciones [44]

```
QuantumCircuit.s(qbit)
QuantumCircuit.sdg(qbit)
```

Al igual que la puerta Pauli-Z no produce cambios en los estados $|0\rangle$ y $|1\rangle$ así que observamos su efecto en el valor de matriz unitario del cúbit (figura 6.10).

```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(qc, backend))
job.result().get_unitary(qc, decimals=3)
```

```
array([[1.+0.j, 0.+0.j],
       [0.+0.j, 0.+1.j]])
```

(a) Resultado de la puerta S

```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(qc, backend))
job.result().get_unitary(qc, decimals=3)
```

```
array([[1.+0.j, 0.+0.j],
       [0.+0.j, 0.-1.j]])
```

(b) Resultado de la puerta S†

Figura 6.10: Aplicación de las puertas S y S† a un cúbit inicializado en el estado $|0\rangle$.

6.3.4. Puertas T and T†

Las puertas T y T† son una raíz cuadrada de la puerta S (y su conjugada). Por tanto realizan una rotación de 45° sobre el eje Z.

Las matrices que definen su comportamiento son:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \qquad T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix} \tag{6.33}$$

Pueden ser utilizadas en Qiskit mediante las funciones [44]

```
QuantumCircuit.t(qbit)
QuantumCircuit.tdg(qbit)
```

Los resultados son paralelos a los de las puertas S y S† a excepción de la última componente de la matriz unitaria que pasa de ser i o $-i$, a ser, $0.707 + 0.707i$ o $0.707 - 0.707i$. Dos aplicaciones consecutivas de la puerta T resultan en una aplicación de la puerta S, y cuatro aplicaciones equivalen a una puerta Pauli-Z.

De forma práctica a la hora de diseñar circuitos cuánticos mediante un lenguaje de alto nivel las puertas S y T no suelen emplearse a menudo, debido a la existencia de la puerta U que comentaremos algo más adelante. Sin embargo la existencia de estas puertas tiene una cierta importancia a nivel Hardware y es por ello que se mencionan.

6.3.5. Puerta Hadamard

La puerta Hadamard transforma el estado $|0\rangle$ en $|+\rangle$, $|1\rangle$ en $|-\rangle$ y $|i\rangle$ en $|-i\rangle$. Visualmente consiste en una rotación sobre el eje $x+z$ de 180°.[50] Como sabemos, los estados $|+\rangle$ y $|-\rangle$ son estados en los que la probabilidad de medir los estados base es igual, esto provoca que la puerta Hadamard sea utilizada frecuentemente para obtener cúbits en estados de equiprobabilidad.

La matriz que define el comportamiento de esta puerta es

$$Z = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{6.34}$$

Puede ser utilizada en Qiskit mediante la función [44]

```
1 QuantumCircuit.h(qbit)
```

El estado observado en la figura 6.11 es por tanto un estado "aleatorio", donde al realizarse mediciones se obtendrán como resultados el estado $|0\rangle$ y $|1\rangle$ con la misma probabilidad. Aquí comienza a dilucidarse la relación con el no-determinismo que se explicó anteriormente. Si tenemos un sistema cuya entrada son n cúbits que son, al menos parcialmente, 0 y 1 al mismo tiempo, sería semejante a ejecutar el sistema con las 2^n posibles entradas que pueden darse con n cúbits.

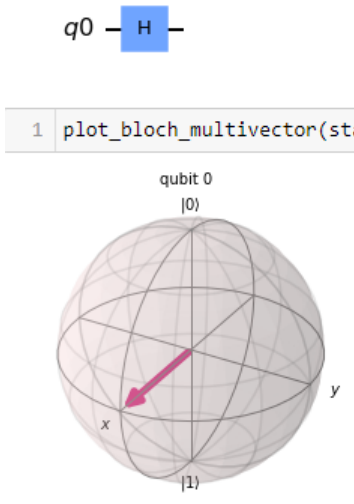


Figura 6.11: Aplicación de la puerta Hadamard a un cúbit inicializado en el estado $|0\rangle$

6.3.6. Puerta U

La puerta U representa una rotación aplicada a un cúbit con los ángulos de Euler. Ya que las rotaciones mantienen la probabilidad total del estado en 1, cualquier rotación es considerada una puerta cuántica, y, como tal, podemos crear una puerta cuántica genérica que nos permita representar cualquier rotación posible.

Es evidente que esta puerta que estamos por definir tendrá parámetros, concretamente tres, uno por cada eje de rotación de Euler, a los que llamaremos θ, ϕ y λ.[44]

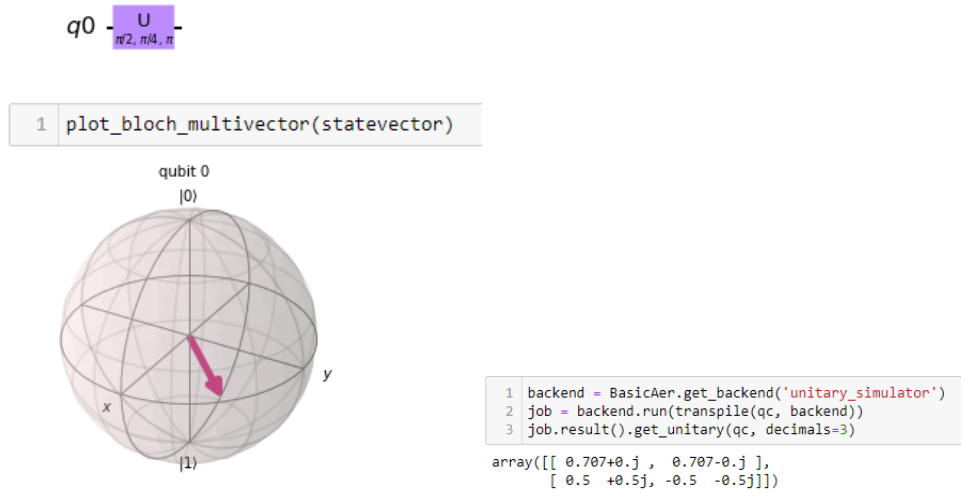
La existencia de estos parámetros hace imposible la existencia de dicha puerta como circuito físico (aunque algunos investigadores ya se encuentran buscando solución a esto [17]), pero a nivel lógico y teórico puede ser utilizado para reducir considerablemente la cantidad de puertas cuánticas que empleamos en nuestros circuitos.

La matriz que define la puerta U es

$$U(\theta,\phi,\lambda) = \begin{pmatrix} cos(\frac{\theta}{2}) & -e^{i\lambda}sin(\frac{\theta}{2}) \\ e^{i\phi}sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)}sin(\frac{\theta}{2}) \end{pmatrix} \tag{6.35}$$

Puede ser utilizada en Qiskit mediante la función [44]

```
1 QuantumCircuit.u(theta, phi, lambda, qbit)
```



(a) Esfera de Bloch **(b)** Matriz unitaria

Figura 6.12: Resultados de la aplicación de una puerta U($\frac{\pi}{2}, \frac{\pi}{4}, \pi$) sobre un cúbit inicializado en el estado $|0\rangle$

6.3.7. Puerta P

La puerta cuántica P actúa de forma semejante a la *U-gate* pero precisa de un único parámetro que es traducido en una fase que se aplica sobre el cúbit.

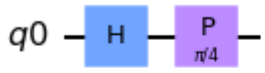
La matriz que define la puerta P es

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}. \tag{6.36}$$

Puede ser utilizada en Qiskit mediante la función

```
1 QuantumCircuit.p(lambda, qbit)
```

[44]




```
1 plot_bloch_multivector(statevector)
```

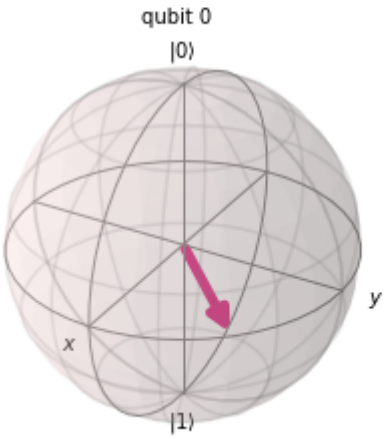


Figura 6.13: Aplicación de la puerta P a un cúbit en el estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

6.4. Puertas cuánticas para 2 cúbits

Para hablar sobre el proceso de alterar estados cuando tenemos sistemas formados por múltiples cúbits debemos hacer primero un inciso en cuanto a representación y cálculo de dichos estados compuestos.

Cuando tenemos múltiples cúbits formando un mismo sistema, escribimos sus estados como un *producto tensorial*, denotado mediante el símbolo $\otimes$. Habitualmente se comprime la notación añadiendo los estados de cada cúbit dentro de un mismo *Bra*, quedando la base Z para dos cúbits como $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, y cualquier estado general de dos cúbits será una superposición de estos cuatro estados base:

$$c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle. \tag{6.37}$$

El cálculo de las probabilidades se hace de forma idéntica al cálculo para un solo cúbit y este proceso se puede extrapolar a cualquier número de cúbits. Es sencillo observar que la cantidad de amplitudes de probabilidad que tenemos en el sistema crece exponencialmente conforme se añaden cúbits al sistema, lo cual se estipula que puede ser uno de los motivos por los que los ordenadores clásicos tendrán dificultades para simular ordenadores cuánticos. [50]

El producto tensorial se calcula mediante el producto de Kronecker([9]) que consiste en multiplicar cada elemento de una matriz por la totalidad de la matriz a la que multiplica.

Siguiendo esto y aplicando el producto tensorial, tenemos que, dados dos operadores, A y B, que actúan cada uno sobre un cúbit, el operador conjunto $A \otimes B$ que actúa sobre ambos cúbits se define mediante la matriz:

$$A \otimes B = \begin{pmatrix} A_{00}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{01}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \\ A_{10}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{11}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \end{pmatrix}. \tag{6.38}$$

De forma análoga, los vectores base para un sistema de dos cúbits se forman mediante el producto tensorial de las posibles combinaciones de estados base que forman la base vectorial de un sistema con un sólo cúbit:

$$\begin{aligned} |00\rangle &= \begin{pmatrix} 1\begin{pmatrix}1\\0\end{pmatrix} \\ 0\begin{pmatrix}1\\0\end{pmatrix} \end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad |01\rangle = \begin{pmatrix} 1\begin{pmatrix}0\\1\end{pmatrix} \\ 0\begin{pmatrix}0\\1\end{pmatrix} \end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \\ |10\rangle &= \begin{pmatrix} 0\begin{pmatrix}1\\0\end{pmatrix} \\ 1\begin{pmatrix}1\\0\end{pmatrix} \end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \quad |11\rangle = \begin{pmatrix} 0\begin{pmatrix}0\\1\end{pmatrix} \\ 1\begin{pmatrix}0\\1\end{pmatrix} \end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}. \end{aligned} \tag{6.39}$$

[44]

De ahora en adelante en el trabajo emplearemos la notación comprimida en su mayoría ya que es más sencilla y suele aparecer de esta manera en bibliografía y artículos.

Se estudiarán a continuación las puertas cuánticas que operan sobre dos cúbits. Éstas puertas, a excepción de la puerta SWAP, son comunmente llamadas *controlled gates*, y consisten en un cúbit que actúa como control (determina cuando se aplica la puerta), y otro cúbit que es el receptor de la operación y que ve su estado alterado.

Esto hace que sea importante determinar qué cúbit es el que actúa como control y cual el que actúa como receptor, ya que la función definida, y, por tanto, la matriz que representa la puerta cuántica, es ligeramente diferente según si el control es el cúbit más significativo ("Most significant bit" MSB) o es el cúbit menos significativo ("Least significant bit" LSB). Empleando Qiskit siempre será el primer cúbit añadido como parámetro el control, y el segundo, el cúbit objetivo.

6.4.1. Puertas Pauli controladas

Tal y como vimos en el apartado 5.3.2, existen tres tipos de puertas Pauli, una por cada eje (x, y, z), cada una de las versiones controladas realiza la misma operación que su puerta Pauli simple asociada, pero únicamente cuando el valor del cúbit de control sea $|1\rangle$.

Pauli-X controlada

La puerta Pauli-X controlada, generalmente llamada en inglés con los términos *Controlled-X* y *Controlled-NOT*, realiza la operación Pauli-X cuando el cúbit control está en el estado $|1\rangle$.

Una forma de representar la operación que realiza es mediante la siguiente función [50]:

$$CNOT|a\rangle|b\rangle = |a\rangle|a \oplus b\rangle \tag{6.40}$$

La matriz que representa su funcionamiento cuando el cúbit control es el MSB es

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.41}$$

Mientras que si el cúbit control es el LSB es

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{6.42}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
1 QuantumCircuit.cx(control qbit, target qbit)
```

Partiendo de un registro cuántico q inicializado en el estado $|0\rangle$ y un registro que emplearemos como control inicializado mediante una puerta Pauli-X en el estado $|1\rangle$ obtenemos la negación cúbit objetivo sin alterar el cúbit control (ver figura 6.14)

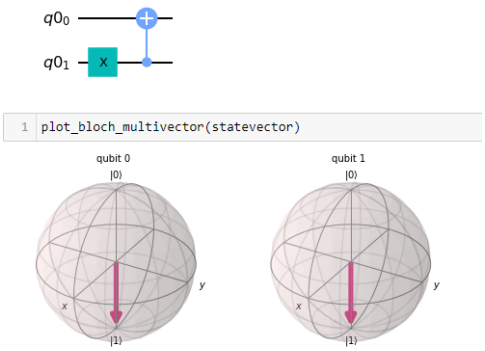


Figura 6.14: Aplicación de la puerta controlada Pauli-X

Pauli-Y controlada

La puerta Pauli-Y controlada, también llamada *Controlled-Y*, realiza la operación Pauli-Y cuando el cúbit control está en el estado $|1\rangle$.

La matriz que representa su funcionamiento cuando el cúbit control es el MSB es

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix} \tag{6.43}$$

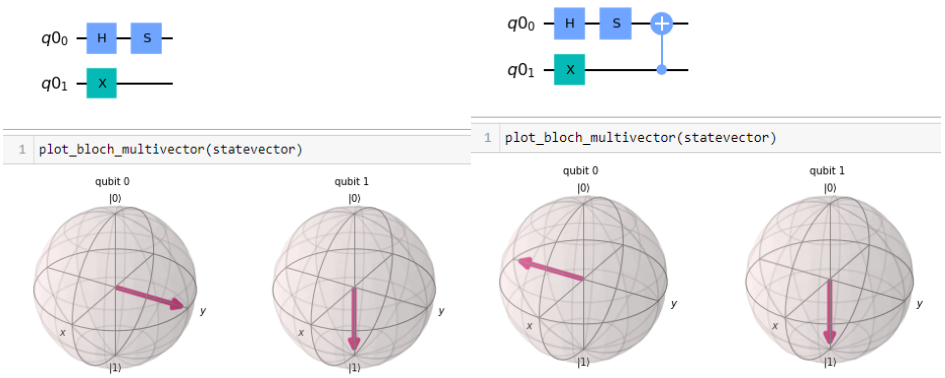
Mientras que si el cúbit control es el LSB es

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & 1 & 0 \\ 0 & i & 0 & 0 \end{pmatrix} \tag{6.44}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
1 QuantumCircuit.cy(control qbit, target qbit)
```

Para ver correctamente los efectos de esta puerta se han preparado dos escenarios, el primero aplicandole al cúbit objetivo una puerta Hadamard y una puerta S (figura 6.15a), de forma que se encuentra en el estado $|i\rangle$, el cúbit control es irrelevante en este escenario. En el segundo, aplicamos la puerta *Controlled-Y*, y observamos como el cúbit objetivo pasa del estado $|i\rangle$ a $|-i\rangle$ (figura 6.15b).



(a) Preparación de los estados **(b)** Inclusión de la puerta en el circuito

Figura 6.15: Aplicación de la puerta controlada Pauli-Y

Pauli-Z controlada

La puerta Pauli-Y controlada, también llamada *Controlled-Z* y *Controlled Phase-Flip*, realiza la operación Pauli-Z, es decir, una inversión de fase, cuando el cúbit control está en el estado $|1\rangle$.

La matriz que representa su funcionamiento es igual siendo el bit de control el LSB y el MSB.

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{6.45}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.cz(control qbit, target qbit)
```

Para visualizar algo mejor el funcionamiento de esta puerta cuántica veremos los siguientes dos casos.

Con los estados base tenemos que la operación C_Z provoca los siguientes cambios:

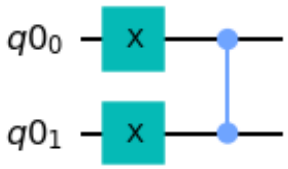
$$\begin{aligned} C_Z|00\rangle &= |00\rangle \\ C_Z|01\rangle &= |01\rangle \\ C_Z|10\rangle &= |10\rangle \\ C_Z|11\rangle &= -|11\rangle \end{aligned} \tag{6.46}$$

Con un sistema de dos cúbits en estado de superposición tendremos:

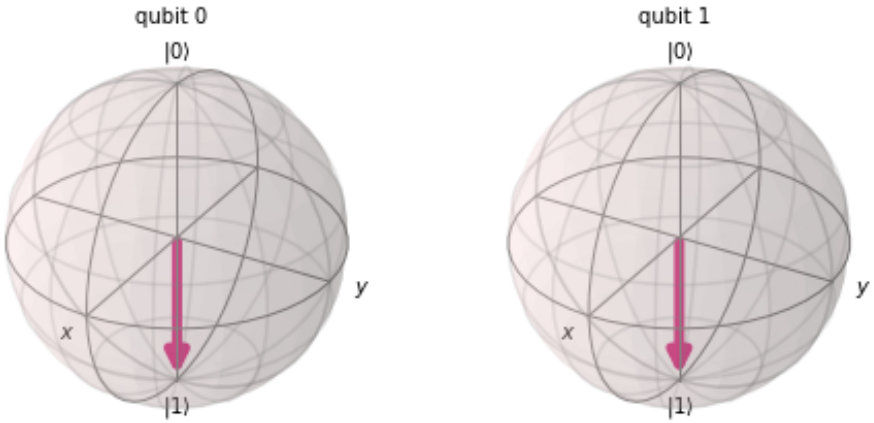
$$\begin{aligned} |\psi\rangle &= \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \\ C_Z|\psi\rangle &= \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle - \delta|11\rangle, \end{aligned} \tag{6.47}$$

Donde α, β, γ y δ son las amplitudes correspondientes a cada estado.

En la figura 6.16 podemos observar como un sistema de dos cúbits inicializado en el estado $|11\rangle$ pasa a estar en el estado $-|11\rangle$ a través de la representación en forma de matriz unitaria de nuestro sistema.




```
1 plot_bloch_multivector(statevector)
```




```
1 backend = BasicAer.get_backend('unitary_simulator')
2 job = backend.run(transpile(qc, backend))
3 job.result().get_unitary(qc, decimals=3)
```

```
array([[ 0.+0.j,  0.+0.j,  0.+0.j,  1.+0.j],
       [ 0.+0.j,  0.+0.j,  1.+0.j,  0.+0.j],
       [ 0.+0.j,  1.+0.j,  0.+0.j,  0.+0.j],
       [-1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j]])
```

Figura 6.16: Aplicación de la puerta controlada Pauli-Z

Debido a la estructura particular de esta puerta, simularla para más cúbits implica tener que realizar una equivalencia de circuitos con otras puertas cuánticas, mientras que con otras puertas la aplicación sobre más de dos cúbits se puede realizar de forma secuencial y directa.

6.4.2. Puerta SWAP

La puerta SWAP intercambia los estados de los dos cúbits implicados, es decir, realiza la función:

$$SWAP|a\rangle|b\rangle = |b\rangle|a\rangle. \tag{6.48}$$

Empleando la puerta SWAP no se puede llegar a un estado entrelazado, ya que, actuando sobre estados en superposición, únicamente intercambia de lugar las amplitudes asociadas a los estados base $|01\rangle$ y $|10\rangle$.

La matriz que representa su funcionamiento es

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6.49}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
1 QuantumCircuit.swap(qbit_0, qbit_1)
```

En la figura 6.17 podemos observar como un sistema de dos cúbits inicializado en el estado $|01\rangle$, mediante una puerta Pauli-X en el cúbit 1, pasa a estar en el estado $|10\rangle$ tras aplicarle la puerta SWAP.

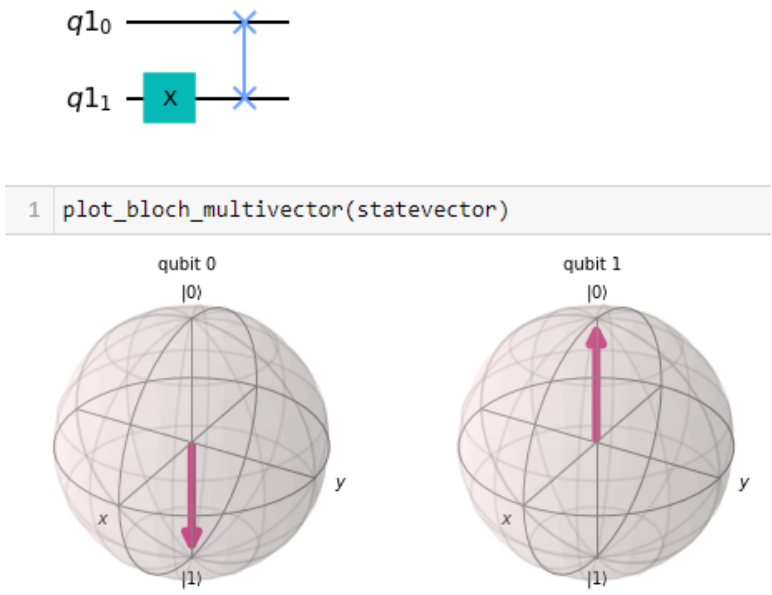


Figura 6.17: Aplicación de la puerta SWAP

6.4.3. Puerta Hadamard controlada

Esta puerta cuántica aplica una puerta de Hadamard sobre un cúbit objetivo si el cúbit control se encuentra en el estado $|1\rangle$.

Vemos a continuación su forma matricial cuando el bit de control es el LSB:

$$C_H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{6.50}$$

Podemos asumir según lo aprendido, que su forma matricial cuando el bit de control es el MSB será:

$$C_H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{6.51}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.ch(qbit_0, qbit_1)
```

Para observar su funcionamiento preparamos un circuito sencillo inicializado al estado $|01\rangle$ y utilizamos el cúbit menos significativo como control para aplicar la puerta hadamard (ver figura 6.18)

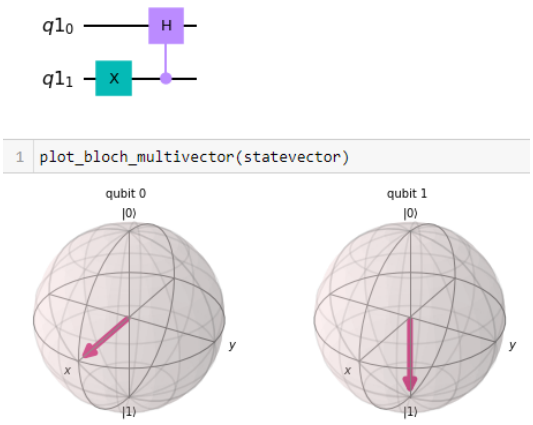


Figura 6.18: Aplicación de la puerta Hadamard controlada

6.4.4. Puertas de rotación controladas

Dentro de esta categoría se encuentran puertas cuánticas que realizan rotaciones de forma condicional apoyándose una vez más en la idea del cúbit control. Qiskit nos da acceso a tres puertas dentro de esta categoría, siendo: *Controlled rotation-Z*, *Controlled phase-rotation* y *Controlled U-rotation*. Ya que sus funcionalidades son idénticas a las equivalentes estudiadas en el apartado de puertas cuánticas para un sólo cúbit (*Rotation-Z*, *P-Gate* y *U-Gate* respectivamente) abreviaremos este apartado indicando únicamente sus formas matriciales, y únicamente cuando el LSB es el bit de control, además de la función asociada en Qiskit.

Rotación-Z controlada

Realiza una rotación en el eje Z cuando el bit de control está en el estado $|1\rangle$. Requiere un parámetro que indica la cantidad de rotación. Su forma matricial es [44]:

$$C_{R_z}(\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\lambda/2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda/2} \end{pmatrix}. \tag{6.52}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.crz(lambda, control qbit, target qbit)
```

Rotación de fase controlada

Aplica una rotación de fase si ambos cúbits están el sistema formado por dos cúbits se encuentra en el estado $|11\rangle$. Requiere un parámetro que indica la cantidad de rotación. Su forma matricial es[44]:

$$C_p(\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{pmatrix}. \tag{6.53}$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.cp(lambda, control qbit, target qbit)
```

Rotación U controlada

Realiza una puerta U cuando el bit de control está en el estado $|1\rangle$. Requiere tres parámetros que coinciden con los que tiene la propia U-Gate. Su forma matricial es [44]:

$$C_u(\theta,\phi,\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i(\phi+\lambda)/2}cos(\theta/2) & 0 & -e^{-i(\phi-\lambda)/2}sin(\theta/2) \\ 0 & 0 & 1 & 0 \\ 0 & e^{i(\phi-\lambda)/2}sin(\theta/2) & 0 & e^{i(\phi+\lambda)/2}cos(\theta/2) \end{pmatrix}. \quad (6.54)$$

Puede ser utilizado en Qiskit mediante la función [44]

```
QuantumCircuit.cu(theta, phi, lambda, control qbit, target qbit)
```

Esta puerta es muy empleada en circuitos cuánticos complejos, y suele ser la base para formar otros componentes lógicos más complejos a los que llamaremos oráculos y que estudiaremos algo más adelante.

6.5. Puertas Cuánticas para 3 cúbits

Hemos visto como todas las puertas cuánticas tienen una forma matricial asociada con cualidades necesarias como su unitaridad, que conlleva la reversibilidad de la puerta. También hemos visto como se pueden crear puertas cuánticas mediante el producto exterior.

En esencia es posible crear puertas cuánticas para n cúbits a voluntad, pero sólo existen dos puertas cuánticas ya implementadas en Qiskit para 3 cúbits, y ninguna para sistemas de mayor tamaño.

Estas dos puertas tienen funcionamientos que ya nos resultan familiares de los apartados anteriores, pero, para explicar la importancia de la primera de ellas, la *Toffoli Gate*, haremos una breve introducción relacionado con la computación clásica.

6.5.1. Conjuntos completos y la clase BQP

En electrónica clásica existe el término *conjunto de puertas lógicas completo.*

Definición 6.6 (Conjunto completo) *. Se dice que un conjunto de puertas lógicas es completo, si empleando únicamente puertas lógicas de ese conjunto se puede representar cualquier función lógica. [50]*

Por otra parte, como discutimos en la introducción, existen numerosas clases de complejidad para separar los problemas a nivel computacional, y la clase P estaba entre ellos, definida a grandes rasgos como la clase que contiene los problemas que se pueden resolver de forma eficiente por un ordenador clásico. Aquí vamos a añadir una clase de complejidad adicional que no se ha mencionado anteriormente pero que tiene especial interés para nosotros que es la clase BQP.

La clase BQP proviene de su análogo clásico, la clase BPP (*bounded-error probabilistic polynomial time*), una clase que contiene problemas que pueden resolverse en tiempo polinómico pero empleando unas máquinas de Turing llamadas "probabilísticas", y con un, acotado, margen de error. En la versión cuántica por supuesto intervienen los computadores cuánticos, y el nombre completo es *bounded-error quantum polynomial time*.

De forma general, la clase BQTime($T(n)$) se define como los problemas que pueden ser resueltos con probabilidad $\frac{2}{3}$ por una máquina de Turing cuántica (QTM) cuyo tiempo de ejecución para una entrada de tamaño n está acotada por $T(n)$.

Esta clase de problemas es propuesta como la candidata a contener todos los problemas que pueden ser computados de forma eficiente mediante un ordenador cuántico. [7]

Vamos a estudiar la puerta Toffoli y con esto esclarecer la importancia de los conjuntos completos y la clase BQP.

6.5.2. Puerta Toffoli

La puerta Toffoli, o *controlled-controlled-NOT*, es una puerta que aparece a menudo en computación cuántica. Es una puerta que opera sobre tres cúbits.

Su funcionamiento es sencillo, realiza una negación sobre el cúbit objetivo si los dos cúbits restantes, que actúan como control, están en el estado $|1\rangle$.

En forma de función la puerta Toffoli puede definirse como sigue:

$$\text{Toffoli}|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|ab \oplus c\rangle. \tag{6.55}$$

Y en forma de matriz se representa mediante una matriz de dimensiones 2^3 x 2^3, ya que estamos operando en un sistema de tres cúbits.

$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{6.56}$$

El funcionamiento lógico es prácticamente idéntico al de la puerta Pauli-X controlada. Sin embargo esta puerta tiene una propiedad de gran interés para la computación teórica.

La puerta Toffoli es un conjunto lógico completo para la computación clásica, es decir, cualquier circuito clásico puede implementarse únicamente mediante puertas Toffoli. Esto implica de forma directa e indiscutible, que cualquier algoritmo clásico eficiente, puede convertirse en un algoritmo formado por puertas Toffoli y ejecutado por un ordenador cuántico. Un paso más allá en el razonamiento y obtenemos que empleando un ordenador cuántico podemos computar de forma eficiente cualquier cosa que un ordenador clásico pueda.

Hablando en términos de complejidad algorítmica, la conclusión es que $P \subseteq BQP$.[50] Esto es algo verdaderamente relevante y presenta la pregunta de si llegará el día en que todos los ordenadores personales sean parcial o totalmente formados por cúbits.

Sin embargo para nuestro trabajo este resultado no es más que una cota inferior para cuán eficientes pueden llegar a ser los ordenadores cuánticos. Sería una posibilidad que todos los problemas NP pudieran ser resueltos en tiempo polinómico por un ordenador cuántico, aunque esta idea se encuentra actualmente rechazada por un artículo publicado en 1996.[6]

La puerta Toffoli puede ser utilizada en Qiskit mediante la función [44]

```
QuantumCircuit.ccx(control qbit_0, control qbit_1, target
    qbit)
```

6.5.3. Puerta Fredkin

La puerta Fredkin, o *controlled swap gate*, realiza la operación de la puerta SWAP si el LSB es $|1\rangle$. La función que representa es:

$$|abc\rangle \rightarrow \begin{cases} |abc\rangle \text{ si } c = 0 \\ |bac\rangle \text{ si } c = 1 \end{cases} \tag{6.57}$$

En forma matricial sería:

$$C_{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6.58}$$

Esta puerta se emplea en Qiskit mediante la función [44]

```
QuantumCircuit.cswap(control qbit, target qbit_0, target
    qbit_1)
```

Como vimos en el apartado de análisis el funcionamiento de esta puerta ha suscitado el interés de algunos experimentos que emplean la puerta Fredkin como núcleo del funcionamiento de circuitos cuánticos complejos.

Al igual que la puerta Toffoli también constituye un conjunto lógico completo y, por tanto, puede ser utilizada para construir cualquier circuito lógico clásico.

6.6. Oráculos

Habiendo explicado brevemente la noción de cúbit así como la notación *Bra-Ket* podemos comenzar a explicar lo que son algo semejante a las funciones en la programación clásica, y que, generalmente, forman las piezas que componen los circuitos cuánticos.

Definición 6.7 (Oráculo cuántico) . *Un oráculo cuántico es una puerta o conjunto de puertas lógicas, que puede definirse mediante una tabla de verdad y que, siguiendo las reglas de los operadores cuánticos, debe ser reversible. [50]*

Suelen aparecer en la bibliografía los oráculos como cajas negras en la que, dada una entrada, recibimos una salida determinada. Este proceso de introducir información en el oráculo y obtener un resultado es equivalente a una "llamada" a una función en programación clásica, por esto, emplearemos de ahora en adelante el mismo término para hacer referencia al proceso de ejecutar el oráculo con una determinada entrada.

Los oráculos suelen formar el núcleo central de todo algoritmo cuántico, y las llamadas a estos se utilizan para determinar la complejidad de los mismos.

Dado un algoritmo cuántico, consistente en un único oráculo, diremos que la complejidad del algoritmo es del órden $O(N)$ si para una entrada de tamaño N al algoritmo, este requiere N llamadas al oráculo para finalizar. A esta notación se le conoce como *Notación de Landau*, para más información sobre la notación consultar el siguiente artículo sobre notaciones[28].

Para crear un oráculo reversible a partir de una determinada función, se toma la salida de dicha función y se realiza la operación de O-exclusivo (XOR, representado mediante el símbolo $\oplus$) cuya tabla de verdad es la que vemos en la tabla 6.1 con un cúbit adicional.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Cuadro 6.1: Tabla de verdad XOR

Podemos verificar que esto provoca que cualquier oráculo sea reversible mediante el siguiente razonamiento. Un determinado oráculo cuántico U_f realizará la operación:

$$|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle. \tag{6.59}$$

Si a continuación introducimos $|y \oplus f(x)\rangle$ en el oráculo junto a $|x\rangle$ de nuevo obtenemos:

$$|x\rangle|y \oplus f(x)\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x) \oplus f(x)\rangle. \tag{6.60}$$

Siguiendo la tabla de verdad de la operación XOR tenemos que:

$$\begin{aligned} &f(x) \oplus f(x) = 0, \forall x \in \{0,1\} \\ &y \oplus 0 = y, \forall y \in \{0,1\} \end{aligned} \tag{6.61}$$

Por lo que el estado final sería $|x\rangle|y\rangle$, lo cual confirma la reversibilidad del oráculo, sin importar la naturaleza del mismo. Además, siguiendo este mismo razonamiento, si inicializamos $|y\rangle$ al estado $|0\rangle$ la salida del oráculo es $|x\rangle|f(x)\rangle$. Al cúbit adicional $|y\rangle$ se le suele llamar *cúbit respuesta*, y al cúbit $|x\rangle$ lo llamamos *cúbit de entrada*.

6.6.1. Oráculo de fase

El oráculo de fase es un tipo particular de oráculo cuántico. Mientras que un oráculo estandar mantiene el cúbit de entrada $|x\rangle$ intacto y aplica la operación $|y \oplus f(x)\rangle$ al cúbit respuesta, un oráculo de fase tiene por objetivo mantener el cúbit respuesta en su estado original y aplicar una determinada fase al cúbit de entrada.

Este proceso se consigue inicializando el cúbit respuesta en el estado $|-\rangle$, lo cual provoca el siguiente comportamiento al realizar una llamada a nuestro oráculo U_f con la entrada $|x\rangle|-\rangle$:

$$\begin{aligned} |x\rangle|-\rangle &= |x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(|x\rangle|0\rangle - |x\rangle|1\rangle) \\ &\xrightarrow{U_f} \frac{1}{\sqrt{2}}(|x\rangle|0 \oplus f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle) \\ &= \begin{cases} \frac{1}{\sqrt{2}}(|x\rangle|0\rangle - |x\rangle|1\rangle) \text{ si } f(x) = 0 \\ \frac{1}{\sqrt{2}}(|x\rangle|1\rangle - |x\rangle|0\rangle) \text{ si } f(x) = 1 \end{cases} \\ &= \begin{cases} |x\rangle|-\rangle \text{ si } f(x) = 0 \\ -|x\rangle|-\rangle \text{ si } f(x) = 1 \end{cases} \\ &= (-1)^{f(x)}|x\rangle|-\rangle. \end{aligned} \tag{6.62}$$

Como podemos ver a lo largo del proceso se le ha aplicado una fase de $(-1)^{f(x)}$ al cúbit entrada y el cúbit respuesta se mantiene en $|-\rangle$.[50]

Este oráculo será una pieza clave para implementar el Algoritmo de Grover.

Capítulo 7

Implementación

7.1. Algoritmo de Grover

El algoritmo de Grover supone una alternativa cuántica a la solución clásica para el problema de la búsqueda no estructurada. Este problema consiste en, dada una base de datos, encontrar un elemento objetivo sabiendo que la base de datos no está ordenada mediante ningún criterio conocido. Generalmente, mediante algoritmos clásicos, esta tarea es de complejidad $O(N)$, ya que, suponiendo que cada elemento se identifica mediante una cadena de bits, buscar en todas las entradas requiere probar con todas las posibles entradas $N = 2^n$. Sin embargo las posibilidades de que el elemento a buscar se encuentre en la primera posición o en la última se consideran iguales, por tanto de media se realizarán $N/2$ comprobaciones, por tanto siguiendo la notación de Landau introducida anteriormente tenemos que efectivamente tiene complejidad $O(N)$.

El algoritmo de Grover supone una mejora teórica sobre la búsqueda clásica, y se afirma que el orden de complejidad para resolver la búsqueda no estructurada mediante el algoritmo de Grover es $O(\sqrt{N})$. Se procede a explicar su funcionamiento.

7.1.1. Algoritmo de Grover - Inicialización

Se parte inicialmente de un conjunto de n cúbits inicializados en el estado $|+\rangle$ y un cúbit respuesta que ignoraremos por el momento. A este estado inicial formado por los n cúbits lo llamaremos $|s\rangle$. Esto supone una superposición uniforme de todas las posibles cadenas de n-bits:

$$\begin{aligned} &|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} |+\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n} \\ &= \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle = |s\rangle \end{aligned} \tag{7.1}$$

Para comprender estas operaciones es importante recordar que $N = 2^n$ y que crear n estados $|+\rangle$ implica aplicar n puertas de Hadamard, una en cada cúbit.

Este estado inicial, al ser una superposición uniforme, contiene de forma equiprobable todas las 2^n posibles cadenas de n bits, incluida la que se quiere encontrar, cadena a la que llamaremos $|w\rangle$. Se puede por tanto separar mediante esta distribución la cadena deseada y las restantes:

$$\begin{aligned} |s\rangle &= \frac{1}{\sqrt{N}}(|w\rangle + \sum_{i\neq w} |i\rangle) \\ &= \frac{1}{\sqrt{N}}|w\rangle + \frac{1}{\sqrt{N}} \sum_{i\neq w} |i\rangle) \\ &= \frac{1}{\sqrt{N}}|w\rangle + \sqrt{\frac{N-1}{N}} \underbrace{\frac{1}{\sqrt{N-1}} \sum_{i\neq w} |i\rangle}_{|r\rangle} \\ &= \frac{1}{\sqrt{N}}|w\rangle + \sqrt{\frac{N-1}{N}}|r\rangle \\ &= \sin\theta|w\rangle + \cos\theta|r\rangle \end{aligned} \tag{7.2}$$

donde $|r\rangle$ se define como el conjunto de cadenas en superposición uniforme que no contienen a $|w\rangle$ y θ se define como:

$$\sin\theta = \frac{1}{\sqrt{N}}, \quad \cos\theta = \sqrt{\frac{N-1}{N}}. \tag{7.3}$$

Acabada la inicialización, el estado del sistema se puede dibujar en un plano de coordenadas con $|r\rangle$ y $|w\rangle$ como sus ejes x e y, obteniendo lo que

se observa en la figura 7.1.

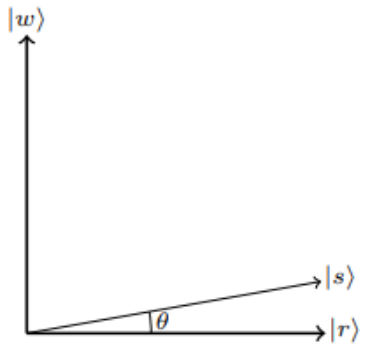


Figura 7.1: Estado inicial del algoritmo de Grover

7.1.2. Algoritmo de Grover - Estructura

Una vez inicializado el estado $|s\rangle$ procedemos a llamar a nuestro oráculo U_f. Este oráculo es un oráculo de fase formado por un oráculo que determina $f(x) = 1$ si y sólo si $x = w$, y el cúbit inicializado a $|-\rangle$ que nos permite la aplicación de fase. Aplicando nuestro oráculo al estado deberemos obtener el siguiente resultado:

$$\begin{aligned} |s\rangle &= \sin\theta|w\rangle + \cos\theta|r\rangle \\ &\xrightarrow{U_f} (-1)^{f(x)}\sin\theta|w\rangle + (-1)^{f(x)}\cos\theta|r\rangle \\ &= (-1)^{f(w)}\sin\theta|w\rangle + (-1)^{f(r)}\cos\theta|r\rangle \\ &= (-1)^{1}\sin\theta|w\rangle + (-1)^{0}\cos\theta|r\rangle \\ &= -\sin\theta|w\rangle + \cos\theta|r\rangle \end{aligned} \tag{7.4}$$

Como puede verse la amplitud de $|w\rangle$ se invierte, dando lugar a un estado simétrico a $|s\rangle$ respecto a $|r\rangle$ (figura 7.2).

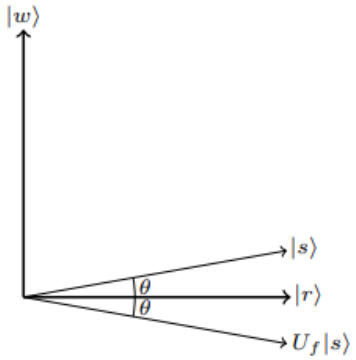


Figura 7.2: Estado del algoritmo de Grover tras el oráculo

El siguiente paso es complicado, e involucra una puerta cuántica cuya finalidad es reflejar un determinado estado en torno al estado $|s\rangle$. Llamaremos a esta puerta R_S. Esta puerta no emplea el cúbit respuesta $|-\rangle$ y el objetivo es el de realizar la operación:

$$\begin{aligned} R_S|s\rangle &= |s\rangle \\ R_S|s^\perp\rangle &= -|s^\perp\rangle \end{aligned}, \tag{7.5}$$

donde $|s^\perp\rangle$ es un estado ortogonal a $|s\rangle$. La explicación y demostración de dicha puerta requiere un nivel superior de conocimientos. Por ello se realiza una simplificación y se asume que dicha puerta existe y su función es la ya mencionada. Aplicando la puerta R_S sobre el estado $U_f|s\rangle$, obtenemos la siguiente situación (figura 7.3):

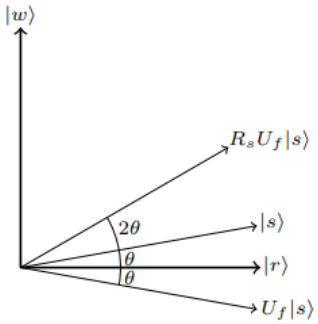


Figura 7.3: Estado del algoritmo de Grover tras la aplicación de R_S

Se observa cómo el resultado de aplicar $R_S U_f|s\rangle = 2\theta|s\rangle$. Si se continúan utilizando estas dos operaciones se realiza un giro con un angulo de 2θ por cada una (ver figura 7.4). El proceso consiste en emplear la operación tantas

veces como sea necesario para acercar $|s\rangle$ hasta un estado suficientemente cercano a $|w\rangle$.

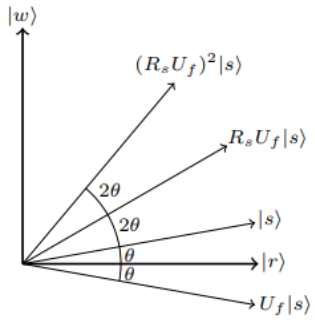


Figura 7.4: Estado del algoritmo de Grover tras dos aplicaciones de R_SU_f

Se procede a calcular la cantidad necesaria de aplicaciones a las que llamaremos t. Sabemos que $|r\rangle$ y $|w\rangle$ son ortogonales por lo que forman un ángulo de $\frac{\pi}{2}$ radianes, y $|s\rangle$ se encuentra desplazado θ respecto a $|r\rangle$. Por tanto:

$$\begin{aligned} \theta + t(2\theta) &= \frac{\pi}{2} \\ t((2\theta) &= \frac{\pi}{2} - \theta \\ t &= \frac{\pi}{4\theta} - \frac{1}{2}. \end{aligned} \tag{7.6}$$

Si asumimos que N es un número grande y recordando que $\sin\theta = 1/\sqrt{N}$, tenemos que:

$$\theta = \sin^{-1}(\frac{1}{\sqrt{N}}) \approx \frac{1}{\sqrt{N}}, \tag{7.7}$$

y, por tanto,

$$t \approx \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \approx \frac{\pi}{4}\sqrt{N}. \tag{7.8}$$

Como se puede ver esto se corresponde con una complejidad $O(\sqrt{N})$, lo cual supone una aceleración cuadrática respecto al algoritmo clásico con $O(N)$.

Es posible que el ángulo final no sea $\pi/2$ por lo que la probabilidad de obtener $|w\rangle$ puede no ser 1. Esto no supone un problema ya que suponiendo un valor grande de N, θ es pequeño, por lo que el estado final estará desplazado por un margen pequeño también. Además, hay técnicas que se pueden

emplear para ajustar el algoritmo de forma que el último paso realice una rotación diferente de forma que el resultado final coincida exactamente con $|w\rangle$.[50]

7.1.3. Algoritmo de Grover - Circuito

Para la demostración de un circuito de algoritmo de Grover simple para una búsqueda no estructurada se ha optado por una implementación en el entorno IBM Quantum (Para más información ver el apartado *Especificación y Requisitos*).

Para ello ejecutamos el *composer* y procedemos con la inicialización del estado $|s\rangle$ (Hasta la primera barrera en la figura 7.5) Empleamos puertas de Hadamard para los estados $|+\rangle$ y una combinación de Pauli-X y Hadamard para el estado $|-\rangle$.

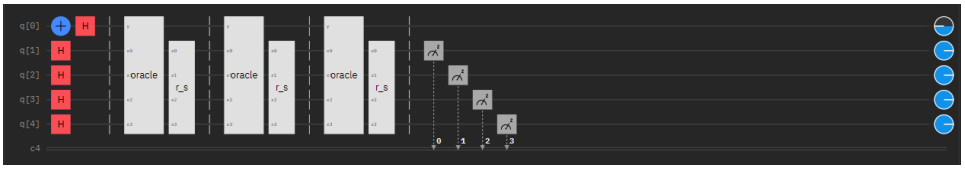


Figura 7.5: Circuito del Algoritmo de Grover

En nuestro ejemplo tenemos que $n = 4$, $N = n^2 = 16$, y, por tanto, $t \approx \frac{\pi}{4}\sqrt{N} = \pi$. Por lo que se aplicará la sucesión de operaciones $R_S U_f$ tres veces (Entre la primera y la última barrera de la figura 7.5) para tratar de acercarnos lo máximo posible al estado objetivo $|w\rangle$. Una vez realizado esto sólo resta medir los valores de todos los cúbits excepto el *answer qubit*.

Para la construcción del oráculo se debe elegir el estado ganador que determinará $f(x) = 1$. En nuestro caso hemos optado por el estado $|1111\rangle$, para lo cual creamos una puerta condicional Pauli-X donde el cúbit objetivo es el cúbit respuesta y los condicionantes son todos los cúbits restantes. Este oráculo es un oráculo de fase como se ha explicado con anterioridad.

Para la construcción de la puerta que debe reflejar respecto a $|s\rangle$, R_S hacemos uso de una puerta auxiliar R_0 que debe construirse de la siguiente manera: Para cada cúbit en el circuito (a excepción del cúbit respuesta) aplicamos una puerta Pauli-X. A continuación, debemos aplicar una puerta condicional Pauli-Z sobre el cúbit de mayor peso, donde los condicionantes son el resto de cúbits. Para realizar esto en IBMQ, debido a que no existe dicha puerta hacemos uso de un circuito equivalente que consiste en aplicar una Pauli-X condicional sobre el cúbit de mayor peso rodeando de puertas Hadamard este cúbit (ver figura 7.6a). Finalmente aplicamos Pauli-Z, Pauli-X, Pauli-Z, sobre el cúbit de menor peso, y una puerta Pauli-X sobre el resto.

Una vez construida esta puerta, la construcción de la puerta R_S es sencilla, consiste en aplicar la puerta R_0 rodeada de puertas de Hadamard en ambos lados y para todos los cúbits como se aprecia en la figura 7.6b.

Pueden verse los resultados obtenidos en una ejecución para el problema

```
gate r_0 x0, x1, x2, x3 {
x x0;
x x1;
x x2;
x x3;

h x3;
c3x x0, x1, x2, x3;
h x3;

z x0;
x x1;
x x2;
x x3;
x x0;
z x0;
}
```

(a) Puerta R_0

```
gate r_s x0, x1, x2, x3 {
  h x0;
h x1;
h x2;
h x3;
r_0 x0, x1, x2, x3;
h x0;
h x1;
h x2;
h x3;
}
```

(b) Puerta R_S

Figura 7.6: Puertas personalizadas del algoritmo de Grover

de la búsqueda no estructurada en el apartado 8.1.

7.1.4. Algoritmo de Grover - Aplicación en 3SAT

Se ha estudiado con detenimiento el algoritmo de Grover en un caso de uso sencillo como es el de la búsqueda en bases de datos no estructuradas. A continuación, vamos a aplicar este algoritmo para resolver uno de los problemas NP-completos mencionados con anterioridad.

Se trata del problema del 3SAT, la simplificación del problema SAT en la que todas las cláusulas tienen exactamente tres literales. Estos tres literales se suelen expresar en forma normal disyuntiva, una estructura compuesta de n variables, $x1, x2, ..., x_n$ donde cada una es de la forma:

$$(y_1 \vee y_2 \vee ... \vee y_n) \tag{7.9}$$

Donde $\vee$ es el operador lógico OR [1]. Concretamente se suele usar una notación llamada DIMACS CNF la cual representa una fórmula en forma normal conjuntiva descrita mediante texto. Hay muchas variaciones del formato por lo que se explicará aquella que va a utilizarse en este experimento.

Un fichero DIMACS comienza con una cabecera de la forma

```
1        p cnf <n_variables> <n_clausulas>
```

donde la cantidad de variables y cláusulas se indican mediante un numero en formato decimal. Posteriormente a esta cabecera aparecen las cláusulas, las cuales vienen codificadas mediante una serie de números, cada número representando un literal, separados por espacios y saltos de línea. Un literal negado tendrá su número precedido de un símbolo − indicando un número

negativo. Cada conjunto de literales que forma una cláusula debe ir seguido de un símbolo 0 representando el final de la cláusula. Cualquier línea que comience con el caracter c se considera comentario.[25]

Siguiendo estas indicaciones la expresión booleana $(x \vee \neg y \vee \neg z)^{(}\neg x \vee z)$ se escribiría como

```
p cnf 3 2
1 -2 -3 0
-1 3 0
```

Con esto se completa el primer paso del algoritmo, obtener un caso de problema 3SAT en formato DIMACS CNF que podemos almacenar en una variable como se realiza en la figura 7.7.

```
input_3sat_instance = '''
c example DIMACS-CNF 3-SAT
p cnf 3 2
1 2 -3 0
-2 3 0
'''
```

Figura 7.7: Fichero DIMACS CNF almacenado en variable

El siguiente paso para aplicar el algoritmo de Grover implica tener acceso a un oráculo de fase que sea capaz de reconocer si $U_f(x1, x2, ..., x3)$ satisface las cláusulas indicadas por el fichero. Es evidente que se necesita un oráculo diferente dependiendo del fichero, y no es posible crear un oráculo que responda la pregunta asociada sin tener previamente dicho fichero.

Calcular manualmente la estructura que tiene que tener el oráculo de fase en función de las cláusulas es una tarea costosa, ya que implica calcular una función $f(x)$ que devuelva 1 si el estado de los cúbits se corresponde con una combinación que satisface las cláusulas y 0 en caso contrario. Para esta tarea Qiskit pone a nuestra disposición una función que genera automáticamente el oráculo necesario teniendo en cuenta un fichero DIMACS de entrada. [44]

```
oracle = PhaseOracle.from_dimacs_file(file_name)
```

Una vez tenemos el fichero DIMACS y el oráculo sólo queda aplicar el algoritmo de Grover, ya que la puerta R_S estudiada con anterioridad sólo es dependiente del número de cúbits. Qiskit pone a nuestra disposición la clase *Grover* dentro del paquete *qiskit.algorithms* con la que podemos realizar una llamada a la función *amplify(problem)* que recibe como parámetro el problema ya definido con el oráculo que obtuvimos anteriormente y ejecuta el

algoritmo. El objeto *problem* debe ser inicializado mediante la función *AmplificationProblem* (en nuestro caso, ya que tratamos de resolver un problema mediante amplificación de fase), que recibe como parámetros el oráculo y la definición de estado objetivo, la cual calcula automáticamente a partir del propio oráculo.

Por supuesto todo este proceso puede hacerse sin emplear las librerías y funciones de Qiskit mediante un circuito como ya hemos demostrado en el apartado anterior, pero este sistema nos permite comprobar diferentes ficheros y obtener resultados mucho más rápido que si tuvieramos que calcular el oráculo necesario para cada fichero y construir el resto del circuito con éste. Podría compararse a las versiones en computación clásica de programación en lenguajes de *alto nivel* y *bajo nivel*.

```
## Grover's Algorithm (Quantum Search) for SAT (SAT-3)
from qiskit import BasicAer
from qiskit.algorithms import Grover
from qiskit.primitives import Sampler
from qiskit.algorithms import AmplificationProblem
from qiskit.circuit.library.phase_oracle import PhaseOracle
from qiskit.tools.visualization import plot_histogram

import tempfile

fp = tempfile.NamedTemporaryFile(mode='w+t', delete=False)
fp.write(input_3sat_instance)
file_name = fp.name
fp.close()
oracle = None
```

```
oracle = PhaseOracle.from_dimacs_file(file_name)
```

```
problem = AmplificationProblem(oracle, is_good_state=oracle.evaluate_bitstring)
grover = Grover(sampler=Sampler())
result = grover.amplify(problem)
```

Figura 7.8: Implementación usando Qiskit del algoritmo de Grover para la resolución del problema 3SAT

El código para el problema queda compacto haciendo uso de las funciones de alto nivel de Qiskit (ver figura 7.8) y los resultados se almacenan en una variable permitiendo hacer uso de ellos de la forma que más interese al usuario. La variable "input_3sat_instance" contiene el texto en formato DIMACS CNF tal y como se observa en la figura 7.7.

Los resultados de la ejecución pueden verse en el apartado 8.2.

7.2. Algoritmo de estimación de fase

Se ha empleado a lo largo del apartado de diseño la representación en forma de matriz unitaria para las puertas cuánticas y vectores para los estados de un sistema cuántico. Esto suele aplicarse a menudo para computar de forma rápida como una puerta cuántica afecta a un estado, ya que haciendo el correspondiente producto matricial obtenemos el resultado de dicha operación.

Hay casos en los que aplicar una operación sobre un vector resulta en el mismo vector, multiplicado a lo sumo por un escalar al que se llama *autovalor* (*eigenvalue* internacionalmente). Estos vectores que poseen dicha propiedad se les concede el nombre de *autovector* (*eigenvector* internacionalmente) de dicha operación.

Ejemplo 7.1 *Si consideramos la puerta Hadamard aplicada sobre el estado* $\begin{pmatrix} 1-\sqrt{2} \\ 1 \end{pmatrix}$ *y calculamos el resultado:*

$$\begin{aligned} H((1-\sqrt{2})|0\rangle + (1)|1\rangle) &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1-\sqrt{2} \\ 1 \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 2-\sqrt{2} \\ -\sqrt{2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{2-\sqrt{2}}{\sqrt{2}} \\ \frac{-\sqrt{2}}{\sqrt{2}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{2}{\sqrt{2}} - 1 \\ -1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{2\sqrt{2}}{\sqrt{2}\sqrt{2}} - 1 \\ -1 \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{2} - 1 \\ -1 \end{pmatrix} \\ &= -\begin{pmatrix} 1-\sqrt{2} \\ 1 \end{pmatrix} \end{aligned} \tag{7.10}$$

Observamos como dicho estado es un autovector de la puerta Hadamard cuyo autovalor es -1*.*

A partir de este concepto surgen tres áreas de desarrollo en el ámbito de los autovectores y autovalores. El primero, estudiar la importancia de los mismos y sus aplicaciones. El segundo, calcular los autovalores y autovectores de una matriz dada. Por último, siendo el ámbito de interés para

este experimento, dada una matriz unitaria U y uno de sus autovectores $|v\rangle$, encontrar o estimar su autovalor.

Es conocido gracias al álgebra lineal que todo autovalor de una matriz unitaria debe ser de la forma $e^{i\theta}$ para un determinado número real θ. Debido a esto, se llama a este problema *estimación de fase*, ya que encontrar el autovalor es equivalente a encontrar la fase θ.[50]

7.2.1. Algoritmo de estimación de fase - Solución clásica

La solución clásica a este problema aplica la propiedad de los autovalores en matrices unitarias para afirmar que

$$U|v\rangle = e^{i\theta}|v\rangle. \tag{7.11}$$

Donde siendo $|v\rangle$ un vector N-dimensional y U una matriz de tamaño $N \times N$, se puede escribir la ecuación anterior como sigue:

$$\begin{pmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ U_{21} & U_{22} & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ U_{N1} & U_{N2} & \cdots & U_{NN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} = e^{i\theta} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} \tag{7.12}$$

Si se multiplica a la izquierda de la igualdad se obtiene

$$\begin{pmatrix} U_{11}v_1 & U_{12}v_2 & \cdots & U_{1N}v_N \\ U_{21}v_1 & U_{22}v_2 & \cdots & U_{2N}v_N \\ \vdots & \vdots & \ddots & \vdots \\ U_{N1}v_1 & U_{N2}v_2 & \cdots & U_{NN}v_N \end{pmatrix} = e^{i\theta} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} \tag{7.13}$$

Donde, haciendo uso de cualquier fila, podemos obtener una ecuación simple de donde extraer el valor de $e^{i\theta}$, por ejemplo si se emplea la primera fila:

$$U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N = e^{i\theta}v_1. \tag{7.14}$$

Y, por tanto, el autovalor es

$$e^{i\theta} = \frac{U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N}{v_1}. \tag{7.15}$$

Esto precisa de N multiplicaciones, $N-1$ sumas, y una división, resultando en un total de $2N = O(N)$ operaciones aritméticas básicas. [50]

Se procede a explicar cómo puede obtenerse una mejora respecto a esta complejidad algorítmica haciendo uso de la computación cuántica.

7.2.2. Algoritmo de estimación de fase - Solución cuántica

Se parte de la base de que poseemos la matriz U, una matriz que representa el funcionamiento de una puerta cuántica de n cúbits, por tanto, U es una matriz de tamaño $N \times N$ donde $N = 2^n$. Se asume que tenemos acceso a n cúbits cuyo estado es el autovector $|v\rangle$:

$$\underbrace{|v\rangle}_{n \text{ cúbits}} . \tag{7.16}$$

Para estimar la fase del autovalor correspondiente con una precisión de m bits se necesitan otros m cúbits adicionales, inicializados en el estado $|0\rangle$:

$$\underbrace{|0...000\rangle}_{m \text{ cúbits}} \underbrace{|v\rangle}_{n \text{ cúbits}} . \tag{7.17}$$

Por tanto se parte de un circuito con $m+n$ cúbits. Haremos referencia a estos dos conjuntos como "registro de autovalor" y "registro de autovector" ya que el conjunto de m cúbits acabará conteniendo la aproximación de m-bits del autovalor, y, los n cúbits representan el autovector $|v\rangle$. [50]

Algoritmo de estimación de fase cuántico - Circuito teórico

Para obtener el autovalor a partir de el estado inincial donde tenemos el registro de autovalor y el registro de autovector debemos aplicar el circuito que se muestra en la figura 7.9.

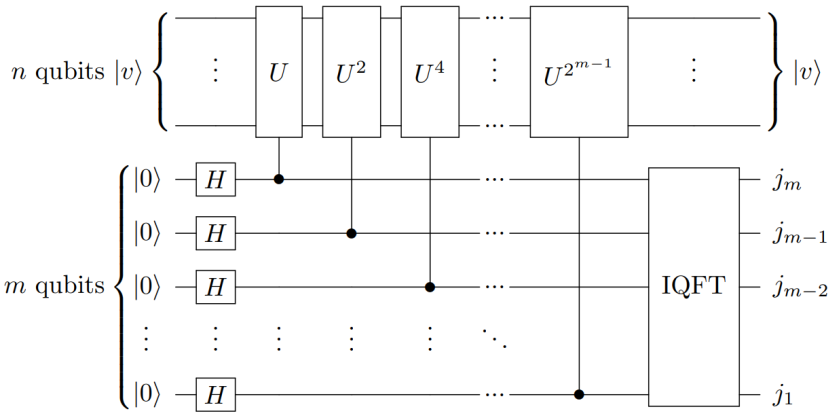


Figura 7.9: Circuito del algoritmo de estimación de fase cuántico

El circuito que puede observarse en la figura 7.9 tiene algunos elementos difíciles de comprender que requieren una explicación dedicada, por lo que vamos a proceder a estudiar, paso a paso, el funcionamiento del mismo.

Algoritmo de estimación de fase cuántico - Estructura

El circuito para resolver la estimación de fase cuántica comienza con la aplicación de puertas de Hadamard en cada cúbit perteneciente al registro de autovalor, produciendo el siguiente cambio:

$$\begin{aligned} |++\cdots+\rangle|v\rangle &= \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)...\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|v\rangle \\ &= \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)...(|0\rangle+|1\rangle)|v\rangle. \end{aligned} \tag{7.18}$$

A continuación, se aplica una puerta U controlada (ver apartado 6.4.4), de forma que el cúbit más significativo del registro de autovalor es el cúbit control, y el registro de autovector es el objetivo. Ya que $U|v\rangle = e^{i\theta}|v\rangle$, esta operación provoca que el vector o estado adquiera una fase de $e^{i\theta}$ cuando el cúbit control está a $|1\rangle$:

$$\frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)...(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)\Big(|0\rangle+e^{i\theta}|1\rangle\Big)|v\rangle. \tag{7.19}$$

Continuamos este proceso aplicando una puerta U^2 controlada, que aplicará dos veces la fase de $e^{i\theta}$, que equivale a $e^{2i\theta}$,cuando el penúltimo cúbit control sea $|1\rangle$:

$$\frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)...(|0\rangle+|1\rangle)\Big(|0\rangle+e^{2i\theta}|1\rangle\Big)\Big(|0\rangle+e^{i\theta}|1\rangle\Big)|v\rangle. \tag{7.20}$$

Se prosigue aplicando la puerta U^4 que produce una fase de $e^{4i\theta}$:

$$\frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)...\Big(|0\rangle+e^{4i\theta}|1\rangle\Big)\Big(|0\rangle+e^{2i\theta}|1\rangle\Big)\Big(|0\rangle+e^{i\theta}|1\rangle\Big)|v\rangle. \tag{7.21}$$

Este procedimiento es finalmente aplicado m veces, donde la última puerta que se utiliza es la puerta $U^{2^{m-1}}$, cuya fase aplicada es $e^{2^{m-1}i\theta}$:

$$\frac{1}{\sqrt{2^m}}\Big(|0\rangle+e^{2^{m-1}i\theta}|1\rangle\Big)...\Big(|0\rangle+e^{4i\theta}|1\rangle\Big)\Big(|0\rangle+e^{2i\theta}|1\rangle\Big)\Big(|0\rangle+e^{i\theta}|1\rangle\Big)|v\rangle. \tag{7.22}$$

Se realiza el siguiente cambio de variables $\theta = 2\pi j$, de manera que si se puede obtener j bastará con multiplicarlo por 2π para obtener θ:

$$\frac{1}{\sqrt{2^m}}\Big(|0\rangle+e^{2\pi i 2^{m-1}j}|1\rangle\Big)...\Big(|0\rangle+e^{2\pi i 4j}|1\rangle\Big)\Big(|0\rangle+e^{2\pi i 2j}|1\rangle\Big)\Big(|0\rangle+e^{2\pi ij}|1\rangle\Big)|v\rangle. \tag{7.23}$$

Ya que $0 \leq \theta < 2\pi$, se obtiene que $0 \leq j < 1$. Se expresa a continuación j como un número binario de m bits $0.j_1j_2...j_m$ que es un número menor que 1, y el estado se escribe como:

$$\begin{aligned}\frac{1}{\sqrt{2^m}}\Big(|0\rangle + e^{2\pi i(j_1j_2...j_{m-1}.j_m)}|1\rangle\Big)...\Big(|0\rangle + e^{2\pi i(j_1j_2.j_3...j_m)}|1\rangle\Big) \\ \times \Big(|0\rangle + e^{2\pi i(j_1.j_2...j_m)}|1\rangle\Big)\Big(|0\rangle + e^{2\pi i(0.j_1...j_m}|1\rangle\Big)|v\rangle.\end{aligned} \tag{7.24}$$

Puesto que $e^{2\pi Ti}$ es una función periódica cuyo valor es 1 para todo $T \in \mathbb{Z}$, todos los componentes a la izquierda del punto decimal pueden ser ignorados ya que son iguales a 1, por lo que podemos expresar el estado mediante la siguiente ecuación.

$$\begin{aligned}\frac{1}{\sqrt{2^m}}\Big(|0\rangle + e^{2\pi i(0.j_m)}|1\rangle\Big)...\Big(|0\rangle + e^{2\pi i(0.j_3...j_m)}|1\rangle\Big) \\ \times \Big(|0\rangle + e^{2\pi i(0.j_2...j_m)}|1\rangle\Big)\Big(|0\rangle + e^{2\pi i(0.j_1...j_m}|1\rangle\Big)|v\rangle.\end{aligned} \tag{7.25}$$

Esto se corresponde con la expresión de la Transformada cuántica de Fourier (QFT) del estado $|j_1j_2...j_m\rangle$, por lo que aplicar la Transformada inversa cuántica de Fourier (IQFT) en el registro de autovalor produce el resultado:

$$|j_1j_2...j_m\rangle|v\rangle. \tag{7.26}$$

Con esto se completa el circuito cuántico para estimación de fase. Se debe recordar que al medir los cúbits obtenemos los valores decimales de j, por lo que debemos recuperar j aplicando:

$$\begin{aligned}j &= 0.j_1j_2...j_m \\ &= \frac{j_1}{2} + \frac{j_2}{4} + ...\frac{j_m}{2^m}.\end{aligned} \tag{7.27}$$

Y con eso se obtiene que la fase del autovalor es $\theta = 2\pi j$, y, en consecuencia, el autovalor es $e^{i\theta}$.

En resumen, para estimar la fase del autovalor con m bits de precisión se requieren m puertas Hadamard, m operaciones U^P controladas, y una

IFQT de m cúbits que requiere $O(m^2)$ puertas. En conjunto, el número total de puertas necesarias es $O(m^2)$. El método clásico emplea $O(N) = O(2^n)$ operaciones aritméticas básicas, por lo que dependiendo del número de bits de precisión m, el método cuántico puede ser más rápido, si bien se asume que podemos crear el estado $|v\rangle$ y realizar operaciones U^P controladas. [50]

Algoritmo de estimación de fase cuántico - Circuito

Procedemos a crear un circuito para ejecutar el algoritmo de estimación de fase. Siguiendo la representación teórica del algoritmo emplearemos los siguientes componentes:

- U: Emplearemos la puerta cuántica Pauli-X como matriz de transformación.
- n: Aplicaremos la puerta a 1 cúbit.
- $N = 2^n$: La matriz U será de dimensiones 2×2.
- $|v\rangle$: El autovector que emplearemos será el estado $|-\rangle$
- m: Estimaremos el autovalor con una precisión de 2 cúbits

Es sencillo calcular teóricamente que $|-\rangle$ es un autovector de la puerta Pauli-X cuyo autovalor es -1, por lo que partimos de un experimento del que sabemos de antemano el resultado esperado, construiremos el circuito según dicta la justificación matemática y compararemos los resultados obtenidos frente a los esperados.

Se comienza por crear la puerta cuántica que generará el estado $|v\rangle$, en este caso estará formada por una puerta Pauli-X seguida de una puerta de Hadamard para obtener el estado $|-\rangle$.

```
1        gate v x {
2          x x;
3          h x;
4        }
```

En segundo lugar se define la nuestra matriz de transformación U y su cuadrado U^2. La segunda puede definirse de forma recursiva realizando dos llamadas a la primera empleando las funciones de OpenQASM.

```
1        gate u_1 x, y {
2          cx x, y;
3        }
4        gate u_2 x, y {
5          u_1 x, y;
6          u_1 x, y;
7        }
```

Para finalizar los preparativos del circuito se crean las puertas cuánticas necesarias para realizar la IQFT, que, siendo $m = 2$, consiste en una única puerta U controlada (ver apartado 6.4.4) que realiza una rotación de $\frac{-\pi}{2}$.

```
gate r2 x, y {
cu1(-pi / 2) x, y;
}
```

Una vez declaradas las puertas necesarias para el experimento se procede a ensamblar el circuito. Las operaciones hasta la primera barrera de la figura 7.10 son la inicialización del circuito, creando nuestro estado $|v\rangle = |-\rangle$ y los m cúbits del registro de autovalor en el estado $|+\rangle$.

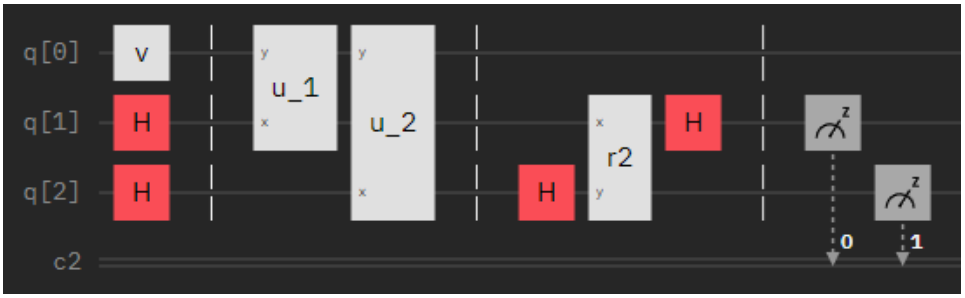


Figura 7.10: Circuito para estimación de fase

Las operaciones entre la primera y segunda barrera de la figura 7.10 son las puertas U y U^2 que aplican la fase definida en el apartado de teoría a los cúbits del registro de autovalor y que suponen la solución a nuestro problema.

Las operaciones entre la segunda y tercera barrera de la figura 7.10 forman el algoritmo de transformación inversa cuántica de Fourier, haciendo uso de la puerta declarada con anterioridad. Al finalizar este paso, los cúbits q[1] y q[2] contienen los valores decimales de j.

El último paso es realizar las mediciones de los cúbits del registro de autovalor como se observa tras la última barrera en la figura 7.10 para obtener mediante los valores decimales el valor completo de j. Si el experimento procede de forma adecuada deberíamos obtener los siguientes valores: $j_1 = 1, j_2 = 0 \rightarrow j = \frac{1}{2}$. Si se tiene en cuenta la definición de j explicada con anterioridad (ver apartado 7.2.2):

$$\theta = 2\pi j \tag{7.28}$$

Por lo que tendremos que $\theta = \pi$ y por tanto el autovalor se calcula como:

$$\text{autovalor} = e^{i\theta} = e^{\pi i} = -1. \tag{7.29}$$

Algoritmo de estimación de fase cuántico - Aplicación en TSP

La aproximación propuesta por [42] plantea resolver el problema del TSP empleando el algoritmo de estimación de fase cuántico. Para ello es necesaria una codificación concreta ya que la aplicación del algoritmo al problema no es banal.

En los algoritmos clásicos se toma de entrada una matriz A tal que $[A]_{ij} = \phi_{ij}$, donde ϕ_{ij} es el coste (o cualquier otra medida cuantitativa) necesario para viajar de i a j. Este coste es el que se pretende minimizar a la hora de resolver el problema del TSP. En la versión cuántica introduciremos dicho coste en forma de fases por dos mostivos principales; en primer lugar, la descripción de costes recién mencionada en forma matricial no tiene por qué ser unitaria, lo que imposibilita trabajar con dicha matriz en un procesador cuántico. En segundo lugar, las fases irán sumándose conforme se realicen multiplicaciones o productos tensoriales de estados, con estas fases como coeficientes, es decir, el coste se irá acumulando en forma de fase, lo cual es requerido para el proceso de búsqueda.

Por tanto, representaremos los costes mediante una matriz B, donde $B_{ij} = e^{i(\phi_i j)}$.

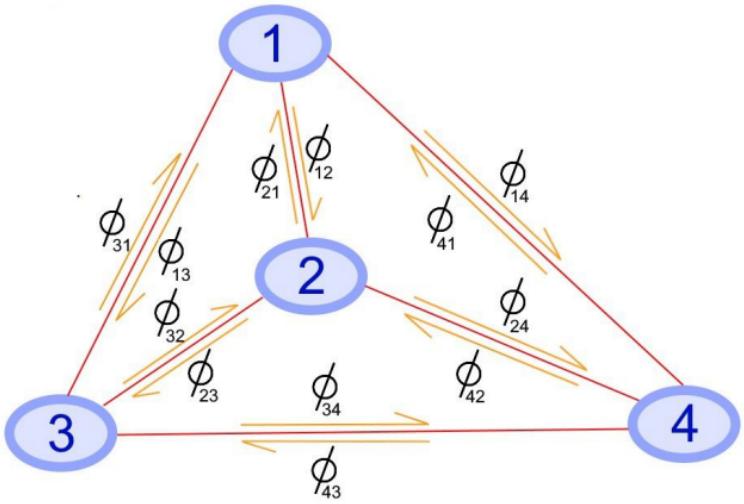


Figura 7.11: Esta figura extraída de [42] ilustra una instancia del problema TSP con cuatro ciudades, cada una nombrada mediante un número del 1 al 4, y los costes de viajar entre ellas están mostradas en las aristas representadas por ϕ_{ij}, se trata de el caso más general de TSP en el que el coste de viajar desde i hasta j no tiene por qué coincidir con el coste de viajar desde j hasta i.

El siguiente paso es construir matrices unitarias U_j a partir de la matriz B como sigue:

$$[U_{ij}]_{kk} = \frac{1}{\sqrt{N}}[B]_{jk}, \tag{7.30}$$

donde N es el número de ciudades y $j, k \in [1, N]$. El resto de elementos de la matriz U_j se inicializan a cero, por lo que se tiene una matriz diagonal unitaria. A continuación se utilizan estos elementos unitarios para formar otra matriz unitaria mediante el producto tensorial de todos los unitarios; $U = U_1 \otimes U_2 \otimes ... \otimes U_N$. Se tiene que U es una matriz diagonal, ya que es el producto tensorial de N matrices diagonales. Esto implica que los autovectores de la matriz U son estados base computacionales $(|0\rangle, |1\rangle, ...)$ cuyos autovalores son los correspondientes elementos de la diagonal.

Esto supone $(N-1)!$ elementos diagonales que serán de interés frente a los N^N elementos totales. Estos elementos tendrán el coste total de los $(N-1)!$ posibles ciclos Hamiltonianos como fases. Es decir, hay $(N-1)!$ autovectores de U cuyo autovalor será el coste total del ciclo Hamiltoniano correspondiente como fase. Se emplea el algoritmo de estimación cuántica para extraer en forma binaria las fases de dichos elementos de interés, y, posteriormente, se puede emplear un algoritmo de búsqueda para obtener el mínimo coste y la ruta correspondiente.

En el ejemplo con 4 ciudades, tenemos $(N-1)! = 6$ posibles ciclos Hamiltonianos. Si se impone una restricción de forma que $\phi_{ij} = \phi_{ji}$ este valor se reduce a la mitad. Por sencillez supondremos el caso específico del viajante de comercio en que los caminos son de coste simétrico.

En el ejemplo del estudio [42] la matriz de costes D, tiene los siguientes valores:

$$D = \frac{1}{2}\begin{pmatrix} 1 & e^{i\phi_{12}} & e^{i\phi_{13}} & e^{i\phi_{14}} \\ e^{i\phi_{21}} & 1 & e^{i\phi_{23}} & e^{i\phi_{24}} \\ e^{i\phi_{31}} & e^{i\phi_{32}} & 1 & e^{i\phi_{34}} \\ e^{i\phi_{41}} & e^{i\phi_{42}} & e^{i\phi_{43}} & 1 \end{pmatrix}, \tag{7.31}$$

Donde $\phi_{12} = \phi_{21} = \pi/2$, $\phi_{13} = \phi_{31} = \pi/8$, $\phi_{14} = \phi_{41} = \pi/4$, $\phi_{23} = \phi_{32} = \pi/4$, $\phi_{24} = \phi_{42} = \pi/4$ y $\phi_{34} = \phi_{43} = \pi/8$. Partiendo de los datos iniciales se procede a calcular cada unitario U_j a través de la siguiente expresión.

$$\begin{pmatrix} |00\rangle\langle 00| \\ |01\rangle\langle 01| \\ |10\rangle\langle 10| \\ |11\rangle\langle 11| \end{pmatrix} \begin{pmatrix} 1 & e^{i\phi_{12}} & e^{i\phi_{13}} & e^{i\phi_{14}} \\ e^{i\phi_{21}} & 1 & e^{i\phi_{23}} & e^{i\phi_{24}} \\ e^{i\phi_{31}} & e^{i\phi_{32}} & 1 & e^{i\phi_{34}} \\ e^{i\phi_{41}} & e^{i\phi_{42}} & e^{i\phi_{43}} & 1 \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix}. \tag{7.32}$$

Estos unitarios U_j pueden representarse en forma matricial como matrices diagonales cuyos coeficientes son las fases asociadas a cada elemento de la diagonal.

Ejemplo 7.2 *Si se toma la matriz unitaria U_1 en la ecuación 7.32, se puede observar que su construcción se realiza mediante el producto de las filas de la primera matriz por la columna de la segunda matriz, dando lugar a*

$$U_1 = 1|00\rangle\langle 00| + e^{i\phi_{21}}|01\rangle\langle 01| + e^{i\phi_{31}}|10\rangle\langle 10| + e^{i\phi_{41}}|11\rangle\langle 11| \tag{7.33}$$

o, en forma matricial,

$$U_1 = \begin{pmatrix} e^{i0} & 0 & 0 & 0 \\ 0 & e^{i\phi_{21}} & 0 & 0 \\ 0 & 0 & e^{i\phi_{31}} & 0 \\ 0 & 0 & 0 & e^{i\phi_{41}} \end{pmatrix}. \tag{7.34}$$

Se emplea dicha notación matricial para poder realizar una descomposición que nos permita crear los operadores U_j como circuitos cuánticos que serán añadidos al circuito final. La descomposición propuesta por [42] es

$$U_j = \begin{pmatrix} e^{ia} & 0 & 0 & 0 \\ 0 & e^{ib} & 0 & 0 \\ 0 & 0 & e^{ic} & 0 \\ 0 & 0 & 0 & e^{id} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i(c-a)} \end{pmatrix} \otimes \begin{pmatrix} e^{ia} & 0 \\ 0 & e^{ib} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i(d+c-a-b)} \end{pmatrix} \tag{7.35}$$

Para el algoritmo de estimación de fase se necesita el circuito correspondiente a la operación $(U_1 \otimes U_2 \otimes U_3 \otimes U_4)$ controlada, lo cual se puede conseguir mediante $(C-U_1 \otimes C-U_2 \otimes C-U_3 \otimes C-U_4)$, donde $C-U_j$ representa la puerta cuántica U_j controlada.

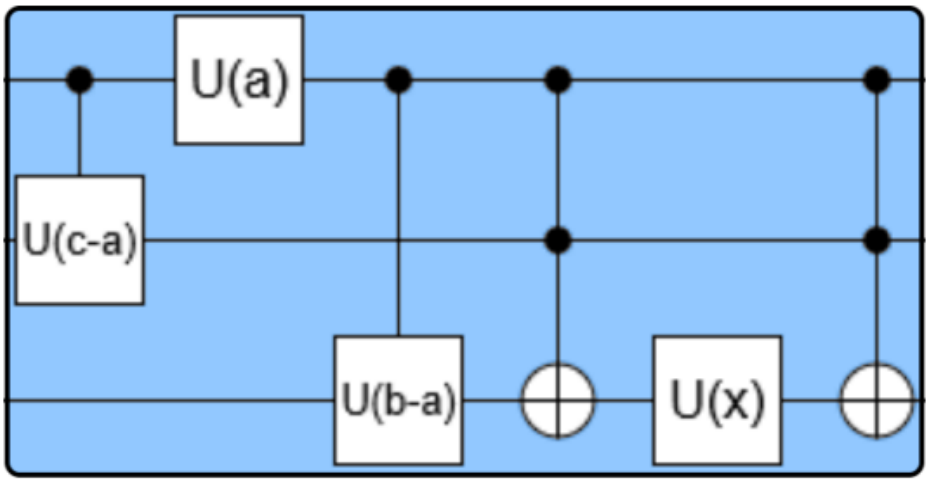


Figura 7.12: Apariencia de un circuito parametrizado para la operación U_j controlada. En la figura el parámetro x se corresponde con $x = (d+c-a-b)/2$

Para construir el circuito que se muestra en la figura 7.12 mediante el compositor de circuitos de IBM emplearemos el siguiente código:

```
gate uni(a, b, c, d) x, y, z {
cu1(c - a) x, y;
u1(a) x;
cu1(b - a) x, z;
ccx x, y, z;
cu1((d - c + a - b) / 2) x, z;
ccx x, y, z;
cu1((d - c + a - b) / 2) x, y;
cu1((d - c + a - b) / 2) x, z;
}
```

Teniendo construída la operación unitaria U_j, construir el producto tensorial de los $N = 4$ unitarios es sencillo empleando las subrutinas de openQASM.

```
gate bigU(a, b, c, d, e, f, g, h, i, j, k, l) m, n, o, p, q, r, s, t,
    u {
uni(0, a, b, c) m, n, o;
uni(d, 0, e, f) m, p, q;
uni(g, h, 0, i) m, r, s;
uni(j, k, l, 0) m, t, u;
}
```

Terminando el proceso, construimos la puerta cuántica específica para este ejemplo añadiendo los valores de las fases dados en la matriz D en la ecuación 7.31, y creamos las puertas cuadráticas necesarias a partir de esta tal y como se vio en el algoritmo de estimación de fase cuántico (ver apartado 7.2.2).

```
// C-U empleando los valores de D
gate finU m, n, o, p, q, r, s, t, u {
  bigU(pi / 2, pi / 8, pi / 4, pi / 2, pi / 4, pi / 4, pi / 8, pi / 4,
       pi / 8, pi / 4, pi / 4, pi / 8) m, n, o, p, q, r, s, t, u;
}

// C-U 2
gate finU2 m, n, o, p, q, r, s, t, u {
finU m, n, o, p, q, r, s, t, u;
finU m, n, o, p, q, r, s, t, u;
}

// C-U 4
gate finU4 m, n, o, p, q, r, s, t, u {
finU2 m, n, o, p, q, r, s, t, u;
finU2 m, n, o, p, q, r, s, t, u;
}

// C-U 8
gate finU8 m, n, o, p, q, r, s, t, u {
finU4 m, n, o, p, q, r, s, t, u;
finU4 m, n, o, p, q, r, s, t, u;
}

// C-U 16
gate finU16 m, n, o, p, q, r, s, t, u {
```

```
26 finU8 m, n, o, p, q, r, s, t, u;
27 finU8 m, n, o, p, q, r, s, t, u;
28 }
29
30 // C-U 32
31 gate finU32 m, n, o, p, q, r, s, t, u {
32 finU16 m, n, o, p, q, r, s, t, u;
33 finU16 m, n, o, p, q, r, s, t, u;
34 }
35
36 // Compresion de todos los C_U en una unica puerta cuantica
37 gate fU a, b, c, d, e, f, n, o, p, q, r, s, t, u {
38 finU f, n, o, p, q, r, s, t, u;
39 finU2 e, n, o, p, q, r, s, t, u;
40 finU4 d, n, o, p, q, r, s, t, u;
41 finU8 c, n, o, p, q, r, s, t, u;
42 finU16 b, n, o, p, q, r, s, t, u;
43 finU32 a, n, o, p, q, r, s, t, u;
44 }
```

Si se aplica finalmente la transformación inversa cuántica de Fourier en el registro de autovalor obtenemos las fases estimadas de nuestro problema, quedándo la estructura final del circuito como se observa en la figura 7.13.

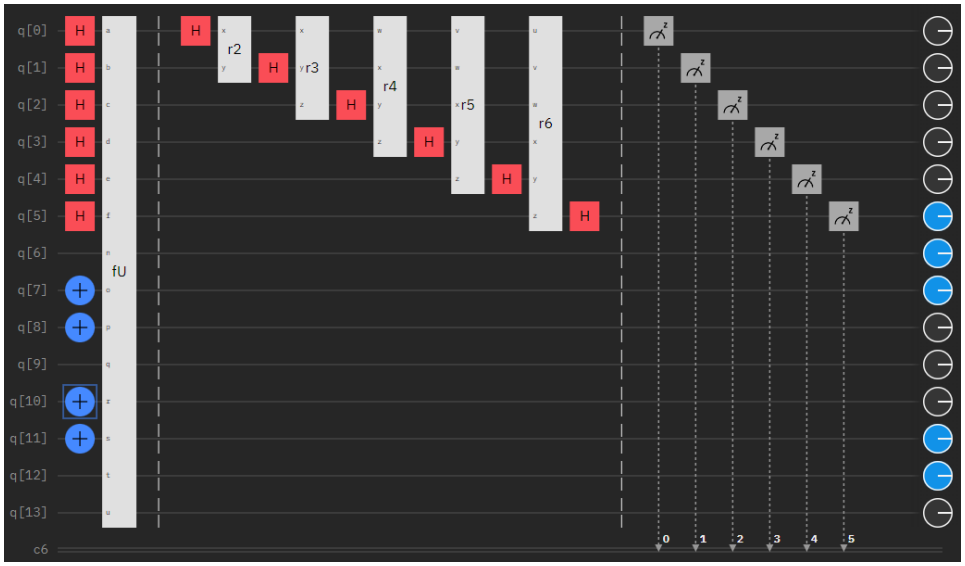


Figura 7.13: Circuito que realiza estimación de fase sobre el autovector $|01101100\rangle$ inicializado mediante puertas Pauli-X al inicio del resgistro de autovector.

Pueden verse los resultados del experimento en el apartado 8.4.

Capítulo 8

Pruebas

En este apartado explicaremos el proceso de ejecución de los distintos circuitos y algoritmos así como los resultados obtenidos en el proceso. Ejecutar programas a través de un backend distinto al propio ordenador en el que se trabaja implica aprender previamente el funcionamiento del mismo, las condiciones que se dan, cómo adquirir acceso al servicio y cómo obtener los resultados. Ya que se trata de un proceso complicado donde no se puede entender plenamente el funcionamiento hasta que se experimenta se ha comenzado practicado el funcionamiento del backend de los computadores cuánticos de IBM generando un circuito cuántico básico formado por dos qubits, uno de ellos en estado de superposición mediante el uso de una puerta lógica de Hadamard, y ambos entrelazados mediante una puerta lógica c-NOT. Ambos cúbits se miden y el resultado se almacena en dos bits clásicos.

Este algoritmo fue simulado mediante el backend de simulación de IBM llamado "qasm-simulator" y, posteriormente, ha sido enviado a un backend cuántico para su ejecución en el backend "ibmq-lima". Inicialmente el trabajo es evaluado y puesto en cola para su ejecución. Podemos acceder al estado de nuestro trabajo en la cola mediante el panel de usuario de IBMQ donde se ofrece una estimación del tiempo de espera hasta que nuestro trabajo sea ejecutado (ver figura 8.1).

En este listado aparecen todos los trabajos enviados, tanto los que han finalizado como los que están pendientes de ejecutarse. Accediendo a cualquiera de ellos podemos ver los detalles del trabajo como la fecha en que se envió, su tiempo en cola y de ejecución, los resultados obtenidos y un acceso directo al circuito en cuestión que ha sido ejecutado.

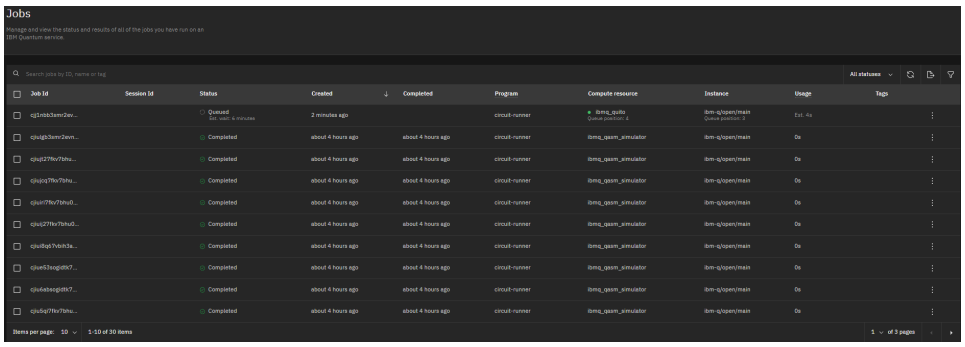
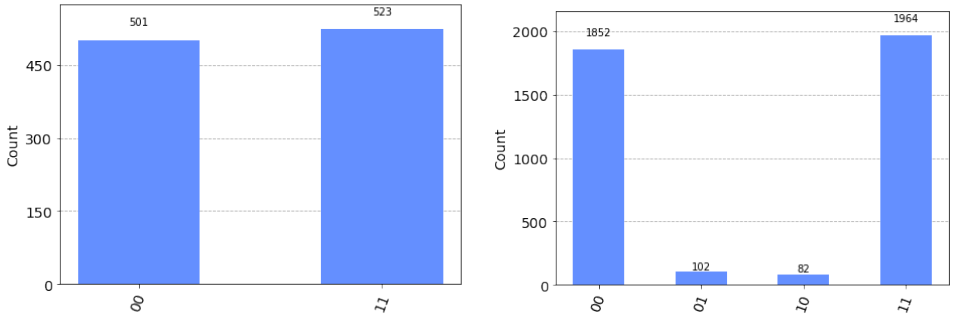


Figura 8.1: Listado de trabajos en IBMQ

El tiempo de espera cuando se envió el trabajo de prueba marcaba dos días. Tras unos minutos estimaba un día de espera y, finalmente, diez minutos después, IBMQ informó que el trabajo estaba completado. Debido a esto se puede asumir que la estimación no se ajusta adecuadamente a los tiempos reales. Si que puede usarse de forma fiable para ver cuantos trabajos hay en cola por delante del nuestro, así como cuantos trabajos tenemos pendientes y sus identificadores.



(a) qasm-simulator **(b)** ibmq-lima

Figura 8.2: Resultados del algoritmo empleando (a): simulación y (b): ejecución

En la figura 8.2 podemos ver como los resultados varían de forma considerable entre la simulación del experimento y la ejecución del mismo mediante un ordenador cuántico real.

La simulación otorga resultados esperados pero estos no emplean mecanismos cuánticos para producir resultados. En su lugar realizan aproximaciones probabilísticas a los resultados reales empleando computación clásica.

La ejecución por el contrario sí emplea mecanismo cuánticos y por tanto podemos observar en la ejecución del algoritmo una tasa de error $\varepsilon = 4.6\,\%$. Sabemos que estos resultados son producto de errores ya que según el funcionamiento teórico del algoritmo, sin importar en qué superposición de estados

$|0\rangle$ y $|1\rangle$ se encuentre el primer cúbit, el segundo debería alterarse mediante un entrelazamiento fuerte cambiando su estado al mismo que el primer cúbit. Cosa que como podemos ver en algunos casos no ocurre.

En resumen, mientras que el simulador proporciona unos resultados que podrían ser generados por un sistema cuántico perfecto, el sistema cuántico real es susceptible a errores en el sistema cuántico.

El tiempo de ejecución del algoritmo en el computador cuántico fue de $2s$ mientras que la simulación tardó $0.004s$. Una diferencia demasiado elevada para un circuito equivalente al programa "Hello World!" de la programación clásica. Este primer experimento es la causa de que se haya descartado la idea original de comparar empíricamente en tiempo los algoritmos cuánticos propuestos frente a sus contrapartes clásicas.

Además de este algoritmo se han implementado otros algoritmos que no aparecerán en la memoria como son el *algoritmo de teleportación de información*, el *algoritmo Bernstein-Vazirani* o el *algoritmo de Deutsch* que pueden verse en el código fuente del cuaderno "jupyter notebooks" y que han sido usados como aprendizaje para comprender los algoritmos que se presentan a continuación.

8.1. Algoritmo de Grover

Presentamos a continuación los resultados obtenidos al ejecutar el algoritmo de Grover (figura 8.3), implementado para realizar la búsqueda no estructurada del valor $|1111\rangle$ tal y como se explicó en el apartado de implementación.

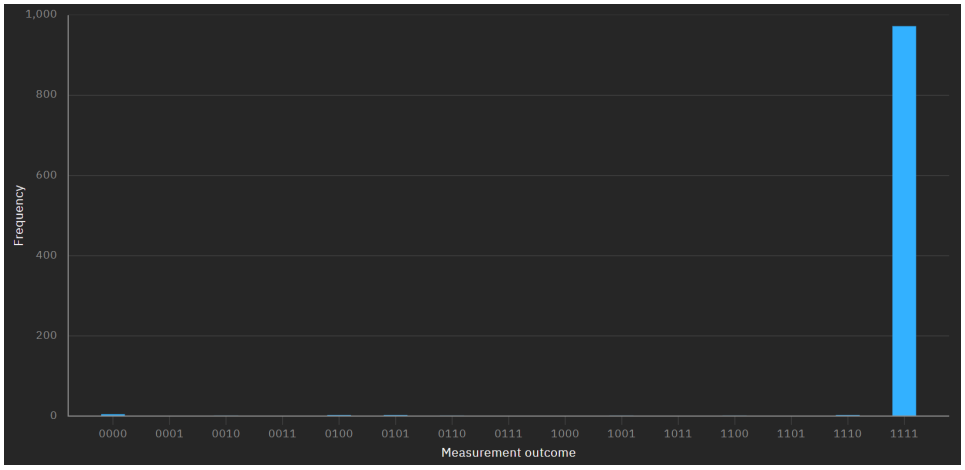


Figura 8.3: Resultados de la búsqueda no estructurada (1000 ejecuciones, ibmq_qasm_simulator)

Al tratarse de la versión sencilla del algoritmo el estado $|w\rangle$ se aproxima pero no con un margen de precisión del 100 %. Los resultados en el simulador son favorables y demuestran que el algoritmo funciona, siendo capaz de reconocer el estado ganador en un 97 % de los casos, recordemos, partiendo de cúbits inicializados en un estado de superposición puro $|+\rangle$.

Pasamos a ver los resultados mediante el backend "ibmq-quito". Este backend posee únicamente cinco cúbits y tiene una capacidad de 2.5K CLOPS (Operaciones de la capa de circuito por segundo) empleando una familia de procesadores que la empresa IBM ha apodado "Falcon", consistente en una pila de varios chips unidos a un "Intermediador", un componente eléctrico que realiza de puente entre varios zócalos de conexión.

8.2. 3SAT - Algoritmo de Grover

Se estudia a continuación el experimento para la resolución del problema NP-completo llamado 3SAT haciendo uso de una generalización del algoritmo de Grover habitualmente llamada *Algoritmo de amplificación de fase*. El funcionamiento es semejante al del propio algoritmo de Grover. Partiendo de un estado inicial uniformemente distribuído se hacen pasar los cúbits por

un oráculo que devuelve $|1\rangle$ en caso de que el estado de entrada sea un caso positivo al problema y $|0\rangle$ en caso contrario. Tras esto se procede con una operación semejante a la puerta R_S explicada junto al algoritmo de Grover que permite amplificar la fase aplicada sobre los cúbits diferenciando de esta manera los estados que han dado una respuesta positiva.

Este proceso se repite cerca de $\sqrt{N}$ veces para tener un margen de error suficientemente bajo, y, repitiendo el experimento una cantidad de evaluaciones suficientemente alto se pueden diferenciar sin problema los estados "ganadores", que vienen definidos por un problema de satisfacibilidad booleana como el descrito a continuación y que ha sido usado en el experimento.

$$(\neg x \vee \neg y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y \vee \neg z) \wedge \\ (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z). \tag{8.1}$$

Vamos primero a ver como funciona el proceso empleando el *Sampler* por defecto que devuelve una métrica que trata de estimar las probabilidades de cada estado en función de cuantas veces han resultado ejecutando el circuito. Podemos ver los resultados en la figura 8.4.

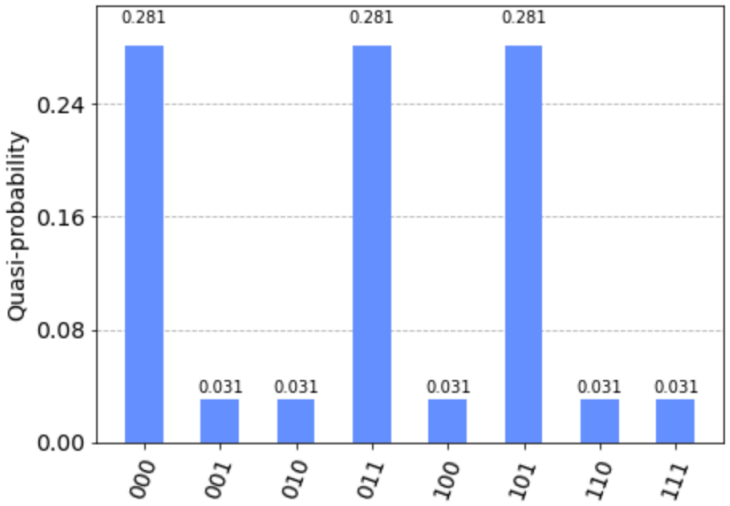


Figura 8.4: Resultados de 3SAT empleando Qiskit

Haciendo algunas modificaciones al código se puede emplear un *Backend-Sampler* en lugar del *Sampler* por defecto. Permitiéndonos así elegir en qué backend queremos que se ejecute. En el ejemplo contenido en el cuaderno Jupyter se emplea el backend "ibmq-lima" empleando el siguiente código:

```
1    provider = IBMProvider()
2    backend = provider.get_backend("ibmq_lima")
3    grover = Grover(sampler = BackendSampler(backend))
```

Lo cual proporciona los resultados que se muestran en la figura 8.5.

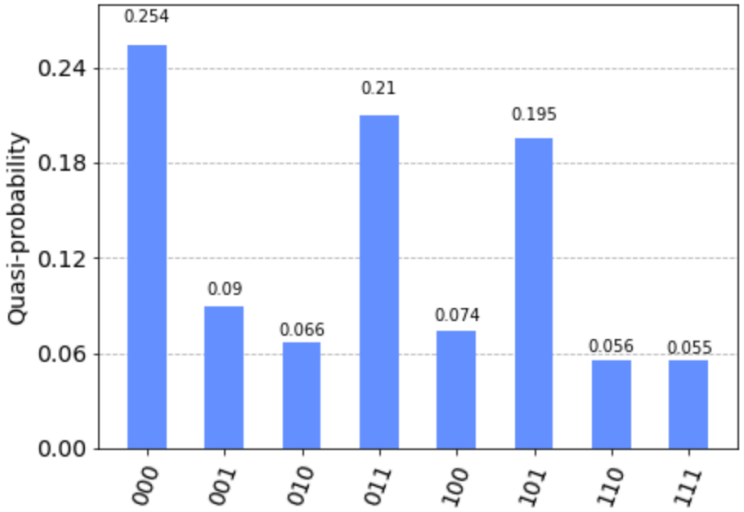


Figura 8.5: Resultados de 3SAT empleando Qiskit y el backend "ibmq-lima"

Se puede observar como a grandes rasgos el resultado se mantiene. Los tres estados que efectivamente verifican la satisfacibilidad del problema expresado anteriormente tienen un valor de probablidad superior a los otros, aunque en la ejecución en el backend "imbq-lima" los valores son considerablemente menos estables. Se puede asumir que el Sampler básico emplea el simulador en lugar de un computador cuántico real para hacer los cálculos y, debido a esto, posee una robustez frente a errores que los sistemas cuánticos actualmente no poseen.

8.3. Estimación de fase

En este apartado se procede a mostrar los resultados del experimento realizado para ejemplificar el funcionamiento del algoritmo de estimación de fase cuántica. Si se retoma la premisa dada en el apartado 7.2.2, el experimento se realiza con el fin de obtener el autovalor del estado $|-\rangle$ siendo autovector de la operación Pauli-X. Un análisis teórico revela que el autovalor de dicho ejemplo es -1 como se demuestra a continuación:

$$X|-\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = - \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} = -|-\rangle. \quad (8.2)$$

Tras la ejecución del circuito mostrado en la figura 7.10 empleando el simulador de IBM ("qasm-simulator"), obtenemos los resultados que aparecen en la figura 8.6.

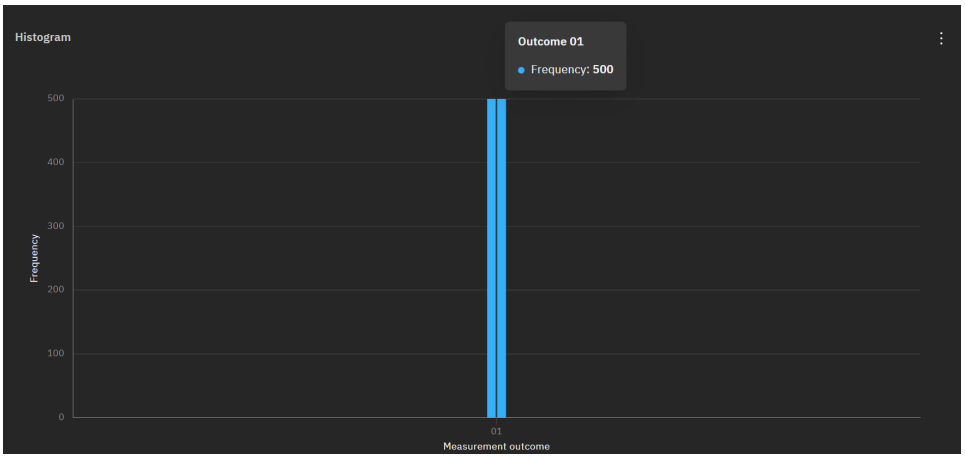


Figura 8.6: Resultados del algoritmo de estimación de fase cuántico (500 ejecuciones, ibmq_qasm_simulator)

La totalidad de las ejecuciones devolvieron el valor 01 de resultado, correspondiendose este con el autovalor -1 como se explicó anteriormente (ver apartado 7.2.2), lo cual supone un éxito en la ejecución del algoritmo. Se procede a observar los resultados de ejecución cuando se emplea un backend real en lugar del simulador (figura 8.7).

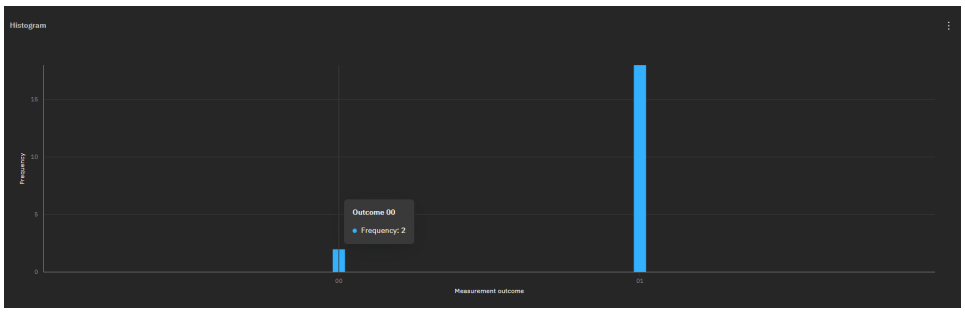


Figura 8.7: Resultados del algoritmo de estimación de fase cuántico (20 ejecuciones, ibmq_lima)

Los resultados, con cierto margen de error, son satisfactorios, y bastante consistentes, con lo que se puede dar por válido el algoritmo y proceder a ajustarlo para resolver el problema NP-completo del viajante de comercio (TSP).

8.4. Viajante de comercio - Estimación de fase

Se procede a presentar los resultados del experimento (ideado por [42] y explicado en el apartado 7.2.2) que consiste en aplicar el algoritmo de estimación de fase cuántico. En el paper original se presenta la tabla 8.1 en la que se calculan los resultados teóricos para los autovalores del problema dado junto a los resultados experimentales.

Autovector	Teórico	Experimental
11000110	100100	100000
01101100	100100	101000
10001101	100000	010000
01110010	100000	010000
11100001	011000	101000
10110100	011000	010000

Cuadro 8.1: Resultados de la simulación del algoritmo de estimación de fase cuántico para resolver el TSP. Se trata de un caso con 4 ciudades cuya matriz de costes se describe en la ecuación 7.31.

En primer lugar, el algoritmo emplea demasiados cúbits como para poder ser ejecutado en los ordenadores cuánticos que IBM pone a disposición del público, siendo el límite de 7 cúbits. Adicionalmente, si reducimos el número de ciudades a tres, la cantidad de hamiltonianos diferentes suponiendo costes simétricos es uno, dos en el caso de costes asimétricos.

En segundo lugar, los resultados teóricos no coinciden con los experimentales, tal y como se afirma en [42]. Este hecho ha sido comprobado en este experimento y los resultados coinciden en su totalidad con los obtenidos en el experimento original. Se afirma que la causa de la discrepancia es la falta de cúbits en el registro de autovalor para poder estimar con mayor precisión el mismo. En el experimento se emplean 6 cúbits para el registro de autovalor, y, por desgracia, no se puede aumentar este tamaño por el momento.

Se realizó una prueba para tratar de escalar el registro de autovalor ya que una vez comprendido el algoritmo es una tarea sencilla. Se requiere una puerta U cuadrática más por cada cúbit en el registro y posteriormente un paso más en la transformada inversa cuántica de Fourier. Esto sin embargo no resultó, ya que en el momento de comenzar la ejecución, la plataforma IBMQ alerta de que no puede ejecutar circuitos de más de 15 cúbits, a pesar de que el procesador del simulador afirma tener capacidad para 32 cúbits. Tras una sesión de investigación no ha sido posible encontrar el motivo tras esta limitación, la cual impide comprobar si la precisión de la estimación hace converger los resultados teóricos y prácticos o se trata de un error de cálculo por los autores.

Pasamos a ver los resultados de la ejecución propia del circuito presentado en la figura 7.13. Se observa en la figura 8.8 cómo el único resultado obtenible es efectivamente el valor 101000.

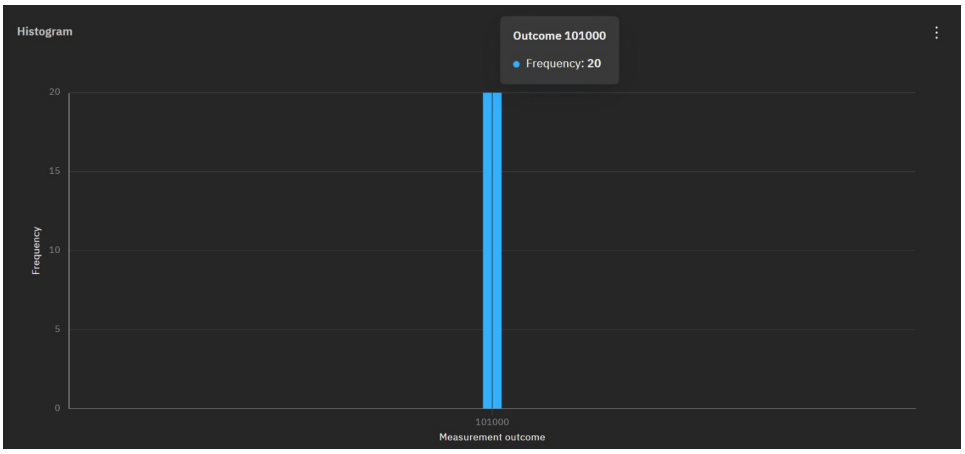


Figura 8.8: Resultado de la simulación de TSP para el autovalor $|01101100\rangle$ (20 ejecuciones, ibmq_qasm_simulator)

Quiero hacer mención personal a la dificultad tras la comprensión de este algoritmo, a pesar de que los resultados no sean especialmente vistosos. Dicho esto, se procede a explicar cómo se recupera la información a partir de los resultados.

En un inicio, se declara la matriz de costes D mediante unas determinadas fases que aplicaríamos a los cúbits del registro de autovalor. En simetría con el problema clásico, a mayor coste, mayor será la fase que apliquemos. Esta fase debe ser normalizada de forma que el coste total máximo no supere 2π, ya que si aplicamos una fase de esa cantidad perderemos toda la información almacenada. Una vez aplicadas las fases, el proceso de estimación de fase cuántico devolverá el autovalor asociado a nuestro autovector (la ruta que se ha tomado) en forma de bits. Ya que las fases se suman entre sí, a mayores fases, mayor será el valor númerico de los bits que formen el registro de autovalor. Por tanto al finalizar el algoritmo para todas las posibles rutas, lo que se tendrá será un vector de números de m bits, donde m es el tamaño del registro de autovalor, y sólo debemos tomar el menor de estos para obtener la ruta de menor coste.

Capítulo 9

Conclusiones y Trabajos futuros

Como apunte previo a las conclusiones me gustaría comentar que la mecánica cuántica es una rama que queda considerablemente lejos del actual ámbito de estudio del grado en Ingeniería Informática. No obstante, la computación cuántica comprende una serie de herramientas a medio camino entre la mecánica cuántica y la programación clásica que puede ser comprendida por los alumnos tanto de Ingeniería Informática como de Física, si bien los primeros considero tendrán mayores facilidades para entender ciertas peculiaridades asociadas a los circuitos, algoritmos y otros.

Con esto quiero expresar que, si bien se trata de un proceso lento y complicado, aprender computación cuántica es una tarea asumible y que bien podría incorporarse a los grados como una asignatura troncal más.

Sin más que añadir se procede con las conclusiones propias del proyecto.

9.1. Conclusiones

Este trabajo ha comenzado tratando de dar una comparación empírica a los algoritmos clásicos frente a la computación cuántica. Esto se debe a que numerosos de los resultados teóricos están ampliamente extendidos por la comunidad científica y, gracias a la tarea divulgativa, también por el resto de la población.

El primer algoritmo por tanto consistió en una pequeña prueba (ver apartado 8) con la finalidad de comprobar si era posible realizar dicha comparación de forma consistente. La respuesta resultó negativa. Los avances tecnológicos, y la democratización de los recursos cuánticos, liderados actualmente por IBM, no están de momento a la altura de ningún ordenador

convencional con el que se ejecute un algoritmo clásico.

Es en este punto que el proyecto toma un ligero desvío y se procede a explicar en profundidad todo lo necesario para que una persona sin conocimientos previos de computación cuántica sea capaz de comprender, replicar y modificar los algoritmos cuánticos actuales. En el proceso se han obtenido resultados de algunos de los algoritmos cuánticos más extendidos (ver apartados 8.1 y 8.3) y, tal y como se propuso inicialmente, se aplicaron dichos algoritmos a problemas reales que en el ámbito de la computación siguen tratándose (ver apartados 7.1.4 y 7.2.2).

Los resultados obtenidos han sido satisfactorios dentro de las limitaciones que se han encontrado en el proceso, lo cual constituye el pilar más fuerte de este apartado, las limitaciones a nivel de recursos cuánticos; desde la capacidad de los procesadores, la disponibilidad de los mismos, la falta de diferentes recursos entre los que elegir demuestran cuánto le falta a esta rama por desarrollarse. Es fácil ver que las aplicaciones de esta tecnología serían numerosas en caso de llegar a desarrollarse lo suficiente, pero es algo que aún requiere avances importantes. A día de hoy no es realista plantear un algoritmo de uso práctico mediante computación cuántica, todos los algoritmos propuestos y demostrados poseen una fuerte base teórica en la que se apoyan pero los resultados empíricos no alcanzan más allá.

Por todo esto concluyo que este trabajo carece actualmente del soporte necesario para ser llevado al siguiente paso que supondría diseñar algoritmos cuánticos que superen empíricamente los actuales.

Considero que puede tomarse este proyecto como un punto de inicio para el aprendizaje de la computación cuántica debido a la cantidad de información condensada y detallada en el mismo. Cada apartado ha sido desglosado de forma que sea entendible por alumnos con conocimientos básicos de matemáticas e insto a cualquier posible lector a emplear este trabajo como guía en su iniciación por esta rama de la computación.

Haciendo referencia a asuntos externos al contenido del trabajo, el mismo ha sido redactado haciendo uso de LaTeXmediante el editor en línea (https://www.overleaf.com/). Ha resultado ser una herramienta realmente versátil, permitiendo, mediante el uso de paquetes, introducir información en el documento con numerosos formatos, algunos sencillos como imágenes y tablas (que tienen su propia nomenclatura y codificación), y otros mucho más sofisticados y complejos como las esferas de Bloch, las cuales están codificadas en el documento y, aunque puedan parecerlo, no son imágenes importadas, si no renderizadas en tiempo de compilación. La bibliografía ha sido realizada mediante BibTeX, una librería especializada para dicha tarea, herramienta que ha sido empleada por primera vez en este trabajo, como otros muchos componentes.

En resumen este trabajo ha sido un gran proyecto, que ha supuesto aprender a utilizar múltiples herramientas así como mejorar en la utilización de herramientas ya conocidas y entre las diferentes iteraciones del proyecto ha podido observarse ese aprendizaje progresivo que ha sido plenamente satisfactorio de experimentar.

9.2. Trabajos futuros

Debido a las limitaciones ya mencionadas considero que es necesario dar tiempo a esta tecnología emergente a desarrollarse, especialmente en el ámbito Hardware, antes de tratar de hacer avances mayores. La computación es una ciencia en la que la teoría y la práctica se encuentran ligadas más que en muchas otras disciplinas, lo que implica que es difícil realizar avances teóricos si estos no tienen un soporte práctico sobre el que comprobar los resultados.

Una vez los procesadores cuánticos alcancen una capacidad y velocidades mínimamente comparables a los actuales desearía personalmente repetir estos experimentos y sumar algunos más complejos para comprobar su eficiencia práctica, y, en el mejor caso, ser capaz de optimizarlos.

Desde el ámbito científico puede continuarse este trabajo para formar un glosario de algoritmos cuánticos, cada uno con su explicación, implementación y aplicaciones posibles, de forma que la información de mayor relevancia para los programadores en computación cuántica se encuentre condensada en un mismo lugar, y no requiera buscar información dispersa entre artículos y libros.

Finalmente, en el ámbito académico como ha sido mencionado previamente puede utilizarse este trabajo o una extensión del mismo como guía para un posible curso de iniciación a la computación cuántica, aunque para dicha tarea sería más recomendable emplear como bibliografía el libro de Thomas G. Wong ([50]).

Capítulo 10

Bibliografía

[1] A. Kerr A. Butterfield G. E. Ngondi. *A Dictionary of Computer Science*. 7.ª ed. Oxford Quick Reference. Oxford University Press, 2016. ISBN: 0199688974, 9780199688975. URL: http://gen.lib.rus.ec/book/index.php?md5=2fd856c4e6bc92cfa8deb03f1b621a73.

[2] W. Bangyal et al. "Comparative Analysis of Low Discrepancy Sequence-Based Initialization Approaches Using Population-Based Algorithms for Solving the Global Optimization Problems". En: *Applied Sciences* 11 (ago. de 2021), pág. 7591. DOI: 10.3390/app11167591.

[3] E. Bauer. "Mecánica cuántica para alumnos de Astronomía". En: *Libros de Cátedra* (2023).

[4] J. S. Bell. "On the Einstein Podolsky Rosen paradox". En: *Physics Physique Fizika* 1 (3 nov. de 1964), págs. 195-200. DOI: 10.1103/PhysicsPhysiqueFizika.1.195. URL: https://link.aps.org/doi/10.1103/PhysicsPhysiqueFizika.1.195.

[5] J. S. Bell. *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*. 2.ª ed. Cambridge University Press, 2004. ISBN: 0521523389, 9780521523387. URL: http://gen.lib.rus.ec/book/index.php?md5=BF857A6CF1D6E2A0BA81C74DB429DA49.

[6] C. Bennett et al. "Strengths and Weakness of Quantum Computing". En: *SIAM Journal on Computing* 26 (oct. de 2016), págs. 1510-1523. DOI: 10.1137/S0097539796300933.

[7] E. Bernstein y U. Vazirani. "Quantum Complexity Theory". En: *SIAM Journal on Computing* 26.5 (1997), págs. 1411-1473. DOI: 10.1137/S0097539796300921. URL: https://doi.org/10.1137/S0097539796300921.

[8] S. Bouamama, Christian Blum y Abdellah Boukerram. "A population-based iterated greedy algorithm for the minimum weight vertex cover

problem". En: *Applied Soft Computing* 12 (jun. de 2012), págs. 1632-1639. DOI: 10.1016/j.asoc.2012.02.013.

[9] B. J. Broxson. "The kronecker product". En: (2006).

[10] J. Chen y R. Xu. "Minimum vertex cover problem based on Ant Colony Algorithm". En: vol. 2011. Ene. de 2011, págs. 125-129. DOI: 10.1049/cp.2011.1389.

[11] V. Choi. "Different adiabatic quantum optimization algorithms for the NP-complete exact cover problem". En: *Proceedings of the National Academy of Sciences* 108.7 (ene. de 2011). DOI: 10.1073/pnas.1018310108. URL: https://doi.org/10.1073%2Fpnas.1018310108.

[12] J. B. Copeland. "The Church-Turing Thesis". En: *The Stanford Encyclopedia of Philosophy*. Ed. por Edward N. Zalta. Summer 2020. Metaphysics Research Lab, Stanford University, 2020.

[13] G. E. Crooks. *Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem*. 2018. arXiv: 1811.08419 [quant-ph].

[14] A. W. Cross et al. *Open Quantum Assembly Language*. 2017. arXiv: 1707.03429 [quant-ph].

[15] Z. Dagdeviren. "Weighted Connected Vertex Cover Based Energy-Efficient Link Monitoring for Wireless Sensor Networks Towards Secure Internet of Things". En: *IEEE Access* PP (ene. de 2021), págs. 1-1. DOI: 10.1109/ACCESS.2021.3050930.

[16] Walter Dean. "Computational Complexity Theory". En: *The Stanford Encyclopedia of Philosophy*. Ed. por Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.

[17] L. Ding et al. "Self-Aligned U-Gate Carbon Nanotube Field-Effect Transistor with Extremely Small Parasitic Capacitance and Drain-Induced Barrier Lowering". En: *ACS nano* 5 (mar. de 2011), págs. 2512-9. DOI: 10.1021/nn102091h.

[18] A. Einstein. "Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt". En: *Annalen der Physik* 322.6 (1905), págs. 132-148. DOI: https://doi.org/10.1002/andp.19053220607. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19053220607. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19053220607.

[19] A. Einstein. "Zur Theorie der Brownschen Bewegung". En: *Annalen der Physik* 324.2 (1906), págs. 371-381. DOI: https://doi.org/10.1002/andp.19063240208. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19063240208. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19063240208.

[20] E. Farhi, J. Goldstone y S. Gutmann. "A Quantum Approximate Optimization Algorithm". En: (nov. de 2014).

[21] Lance Fortnow. "The Status of the P versus NP problem". En: *Commun. ACM* 52 (sep. de 2009), págs. 78-86. DOI: 10.1145/1562164.1562186.

[22] R. Hasudungan et al. "Solving Minimum Vertex Cover Problem Using DNA Computing". En: *Journal of Physics: Conference Series* 1361 (nov. de 2019), pág. 012038. DOI: 10.1088/1742-6596/1361/1/012038.

[23] Md. R. Islam, I. Hossain y R. Shuvo. "Generalized vertex cover using chemical reaction optimization". En: *Applied Intelligence* 49 (jul. de 2019). DOI: 10.1007/s10489-018-1391-z.

[24] M. Javad-Kalbasi et al. "Digitally Annealed Solution for the Vertex Cover Problem with Application in Cyber Security". En: mayo de 2019, págs. 2642-2646. DOI: 10.1109/ICASSP.2019.8683696.

[25] Jix. *Varisat 0.2.2 documentation*. URL: https://jix.github.io/varisat/manual/0.2.0/index.html. (accessed: 10.08.2023).

[26] H. J. Kimble, M. Dagenais y L. Mandel. "Photon Antibunching in Resonance Fluorescence". En: *Phys. Rev. Lett.* 39 (11 sep. de 1977), págs. 691-695. DOI: 10.1103/PhysRevLett.39.691. URL: https://link.aps.org/doi/10.1103/PhysRevLett.39.691.

[27] K. Kishkin, Dimitar Arnaudov y Venelin Todorov. "Multicriteria Optimization of an Algorithm for Charging Energy Storage Elements". En: sep. de 2022, págs. 257-265. ISBN: 978-3-031-06838-6. DOI: 10.1007/978-3-031-06839-3_13.

[28] T. Lattimore y C. Szepesvári. "Notation". En: *Bandit Algorithms*. Cambridge University Press, 2020, págs. xv-xviii.

[29] A. Leporati y S. Felloni. "Three "quantum" algorithms to solve 3-SAT". En: *Theoretical Computer Science* 372.2 (2007). Membrane Computing, págs. 218-241. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2006.11.026. URL: https://www.sciencedirect.com/science/article/pii/S0304397506008851.

[30] D. S. Johnson M. R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. First Edition. Series of Books in the Mathematical Sciences. W. H. Freeman, 1979. ISBN: 0716710455, 9780716710455. URL: http://gen.lib.rus.ec/book/index.php?md5=77f747b4a3cc87aaf94f6a9ea1cbc1cf.

[31] R.A Millikan. "A Direct Determination of "h."". En: *Phys. Rev.* 4 (1 jul. de 1914), págs. 73-75. DOI: 10.1103/PhysRev.4.73.2. URL: https://link.aps.org/doi/10.1103/PhysRev.4.73.2.

[32] W. Tittel N. Gisin G. Ribordy y H. Zbinden. "Quantum cryptography". En: *Rev. Mod. Phys.* 74 (1 mar. de 2002), págs. 145-195. DOI: 10.1103/RevModPhys.74.145. URL: https://link.aps.org/doi/10.1103/RevModPhys.74.145.

[33] M. A. Nielsen y L. I. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[34] M. Mosca P. Kaye R. Laflamme. "An introduction to Quantum Computing". En: Oxford University Press, 2006. Cap. 1.1.

[35] M. Pelikan, Mark Hauschild y Pier Luca Lanzi. "Transfer Learning, Soft Distance-Based Bias, and the Hierarchical BOA". En: mar. de 2012. ISBN: 978-3-642-32936-4. DOI: 10.1007/978-3-642-32937-1_18.

[36] S. Pervaiz et al. "A Systematic Literature Review on Particle Swarm Optimization Techniques for Medical Diseases Detection". En: *Computational and Mathematical Methods in Medicine* 2021 (sep. de 2021), págs. 1-10. DOI: 10.1155/2021/5990999.

[37] M. Planck. "Ueber das Gesetz der Energieverteilung im Normalspectrum". En: *Annalen der Physik* 309.3 (1901), págs. 553-563. DOI: https://doi.org/10.1002/andp.19013090310. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19013090310. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19013090310.

[38] RE. Precup et al. "Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using Slime Mould Algorithm". En: *International Journal of Systems Science* (jun. de 2021), págs. 1-16. DOI: 10.1080/00207721.2021.1927236.

[39] J. Preskill. "Lecture Notes for Physics 219: Quantum Computation". En: (ene. de 1999).

[40] J. Law (editor) R. Rennie (editor). *A Dictionary of Physics (Oxford Quick Reference)*. 8.ª ed. OUP Oxford, 2019. ISBN: 0198821476, 9780198821472. URL: http://gen.lib.rus.ec/book/index.php?md5=2B9519B313187CBB64B2A16063C90276.

[41] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". En: *SIAM Review* 41.2 (1999), págs. 303-332. DOI: 10.1137/S0036144598347011. eprint: https://doi.org/10.1137/S0036144598347011. URL: https://doi.org/10.1137/S0036144598347011.

[42] Karthik Srinivasan et al. "Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience". En: *arXiv: Quantum Physics* (2018). URL: https://api.semanticscholar.org/CorpusID:51690041.

[43] A. Steane. "Quantum computing". En: *Reports on Progress in Physics* 61.2 (feb. de 1998), pág. 117. DOI: 10.1088/0034-4885/61/2/002. URL: https://dx.doi.org/10.1088/0034-4885/61/2/002.

[44] Qiskit Development Team. *Qiskit 0.44.0 documentation*. URL: https://qiskit.org/documentation/. (accessed: 01.08.2023).

[45] T. Toffoli. "Physics and Computation". En: *lnternattonal Journal of Theoretical Physics* 21.3/4 (1982), págs. 165-175. DOI: https://doi.org/10.1007/BF01857724.

[46] A.M. Turing et al. "On computable numbers, with an application to the Entscheidungsproblem". En: *J. of Math* 58.345-363 (1936), pág. 5.

[47] A. V. Aho; J. E. Hopcroft; J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley series in computer science and information processing. Addison-Wesley Pub. Co, 1974. ISBN: 0201000296, 9780201000290. URL: http://gen.lib.rus.ec/book/index.php?md5=4d4c7776a7df0294568d310bdbaa91ba.

[48] C. P. Williams. "Quantum Gates". En: *Explorations in Quantum Computing*. London: Springer London, 2011, págs. 51-122. ISBN: 978-1-84628-887-6. DOI: 10.1007/978-1-84628-887-6_2. URL: https://doi.org/10.1007/978-1-84628-887-6_2.

[49] H. Y. Wong. "Orthonormal Basis, Bra–Ket Notation, and Measurement". En: *Introduction to Quantum Computing: From a Layperson to a Programmer in 30 Steps*. Cham: Springer International Publishing, 2022, págs. 23-31. ISBN: 978-3-030-98339-0. DOI: 10.1007/978-3-030-98339-0_4. URL: https://doi.org/10.1007/978-3-030-98339-0_4.

[50] T.G. Wong. *Introduction to Classical and Quantum Computing*. Rooted Groove., 2022. ISBN: 9798985593105. URL: https://books.google.es/books?id=M3jqzgEACAAJ.

[51] W. Zhang, Jianhua Tu y Lidong Wu. "A multi-start iterated greedy algorithm for the minimum weight vertex cover P3 problem". En: *Applied Mathematics and Computation* 349 (mayo de 2019), págs. 359-366. DOI: 10.1016/j.amc.2018.12.067.

[52] Y.J. Zhang et al. "Applying the Quantum Approximate Optimization Algorithm to the Minimum Vertex Cover Problem". En: *Appl. Soft Comput.* 118.C (mar. de 2022). ISSN: 1568-4946. DOI: 10.1016/j.asoc.2022.108554. URL: https://doi.org/10.1016/j.asoc.2022.108554.

[53] Y.J. Zhang et al. “Applying the quantum approximate optimization algorithm to the minimum vertex cover problem”. En: *Applied Soft Computing* 118 (2022), pág. 108554. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.108554. URL: https://www.sciencedirect.com/science/article/pii/S1568494622000795.