Universidad Nacional de Educación a Distancia
Facultad de Ciencias
Departamento de Física

Memoria del Trabajo de Fin de Grado

ESTUDIO DE ALGORITMOS CUÁNTICOS Y RELACIÓN DE LOS MISMOS CON LA FÍSICA

Raúl Osuna Sánchez-Infante
Tutor: Víctor Alberto Fairén Le Lay
Curso 2020/21

Índice

1. Resumen o abstract

Esta parte queda mejor si se rellena la última.

2. Introducción

2.1. Fundamentos matemáticos y lógicos

En principio, simplemente enunciarlas y quizás incluir una simple información al respecto de las herramientas que se emplearán, ya que ya se han estudiado en los cursos anteriores del grado y no es el objetivo de este TFG. Conviene considerar si se debería mover todo este punto a la sección de anexos.

2.1.1. Números complejos

Simple introducción.

2.1.2. Vectores

Análogo a los números complejos.

2.1.3. Álgebra de Boole y puertas lógicas clásicas

Seguir la idea de los dos puntos anteriores. Breve.

2.2. Bases físicas y particularidades de la física cuántica

Para no caer en el error de hacer un TFG con poca relación con la física, extenderse algo más aquí (en comparación con la implementación de los algoritmo).

2.2.1. Mecánica cuántica y aplicación a algoritmos

- Dualidad onda-partícula
- Superposición
- Entrelazamiento
- Interferencia
- Qubits y esfera de Bloch. Notación bra-ket. Probabilidades de un estado
- Postulados cuánticos
- Puertas cuánticas
- Representación matemática de los circuitos cuánticos
- Supremacía cuántica (resumen)
- Codificación superdensa (resumen)
- Criptografía cuántica (BB84) (resumen)
- Teleportación cuántica (resument)

2.3. Fundamentos computacionales

2.3.1. Algoritmos

Algoritmos cuánticos. Oráculos. Descripción de algoritmos (Deutsch-Josza, Grover, etc...). Puede que haya que mover parte de la explicación dada del punto 5 aquí (o en su defecto, añadir una introducción aquí, pero evitando repetir información).

3. Objetivos

En este trabajo de fin de grado se buscan diversos objetivos, tales como:

- Estudio de diversos ejemplos de algoritmos cuánticos.
- Comparación de los mismos con los algoritmos clásicos equivalentes, en términos como velocidad y eficiencia.
- Implementación práctica personal de un algoritmo ya existente (Deutsch-Josza).
- Implementación personal y ampliación de un algoritmo ya existente (Grover para 3 qubits, implementando la búsqueda de cualquier posible solución dentre de las $2^3 = 8$ posibles, incluyendo soluciones dobles).
- Ejecución de los algoritmos tanto en simuladores cuánticos, como en ordenadores cuánticos reales.

4. Métodos y material

Para el desarrollo de este trabajo se han empleado las siguiente herramientas:

4.1. Lenguaje Python para la programación de los algoritmos.

El lenguaje de programación empleado es python. No sólo por ser muy habitual para cualquier tipo de algoritmo hoy en día, ni por su sencillez, sino principalemente por la librería ya implementada existente, que se trata en el siguiente punto.

4.2. Librería Qiskit [1, 2]

Permite, entre otras cosas, la implementación de puertas cuánticas de una manera simple mediante código. Las posibilidades con esta librería son infinitas, y para cualquier consulta concreta lo mejor es hacer referenciar a la bibliografía añadida aquí. Se entiende que los detalles de Qiskit quedan un poco al margen de la Física y por ello se intenta centrar el trabajo en ella, tomando esta librería como lo que es, una herramienta de la que se hace uso para un fin, como podrían ser las Matemáticas en tantos campos de la Física.

4.3. IBM Quantum Experience [3]

IBM ofrece la posibilidad de acceder a hardware cuántico real mediante una cuenta gratuita, la cual permite ejecutar código en dicho hardware (lo que sería una analogía de implementar las puertas cuánticas deseadas en dicho hardware).

Una vez creada la cuenta, el acceso a ella en el código del programa no tiene mayor complicación y está detallado en la documentación de IBM referenciada aquí. Se ha de crear un token de acceso (que sustituye a la contraseña y se guarda en el propio equipo, en caso de ejecutarse el programa en otro equipo, haría falta el mismo token. O bien se podría modificar el código para usar la cuenta del usuario que desee ejecutarlo, tras haber guardado su propio token). En el caso de que no se opte por visualizar los resultados en hardware cuántico real, no es necesario configurar ninguna cuenta de IBM, ya que esta parte del código no se llegará a ejecutar y no daría por tanto ningún error.

4.4. Control de versiones Github [4]

A la hora de codificar un programa de cualquier tipo, resulta muy útil el empleo de un software para control de versiones. Es más, este mismo software se puede usar para cualquier tipo de archivo (siendo especialmente útil cuando dicho archivo está en texto plano, ya que permite comparar versiones de una manera muy fácil e intuitiva, el ejemplo serían las fuentes de LaTeX (para el PDF correspondiente ya no tiene tanta utilidad al no estar éste en texto plano).

4.5. Procesador de textos LyX [5]

Como se introdujo en el punto anterior, esta memoria se ha escrito en LaTeX. Sin embargo, si no se tiene el conocimiento de dicho lenguaje, la curva de aprendizaje puede ser algo lenta al principio. Resulta mucho más fácil el empleo de un procesador de textos que haga uso LaTeX, sin tener que codificar el mismo. Esto es lo que logra el software libre LyX. Este documento es el segundo empleado para realizar un texto de características similares y el resultado es muy satisfactorio.

5. Resultados

5.1. Algoritmo de Deutsch-Josza

El algoritmo de Deutsch-Josza resuelve un problema que no tiene ningún objetivo práctico, salvo demostrar la existencia de enunciados cuya resolución a partir de un algoritmo cuántico es más eficiente que el método análogo correspondiente mediante computación tradicional.

El algoritmo se basa en hallar si una función cualquiera f(x) es constante o balanceada. Esto quiere decir, para una entrada de un único bit, se quiere hallar si su salida depende de la entrada o no.

La forma más fácil y gráfica de visualizar cada uno de los casos es mediante una tabla de la verdad, habitual en la electrónica digital. Dicha tabla podría tomar dos formas en cada uno de los dos casos, como se muestra en los siguientes apartados

5.1.1. Caso constante

Como el propio nombre indica, la salida será siempre 0 ó 1, independientemente de la entrada

x	f(x)
0	0
1	0

x	f(x)
0	1
1	1

Cuadro 1: Tablas de la verdad para el caso constante

5.1.2. Caso balanceado

La salida se alterna según sea la entrada 0 ó 1:

x	f(x)
0	0
1	1

x	f(x)
0	0
1	1

Cuadro 2: Tablas de la verdad para el caso balanceado

Se puede observar para ambos casos, que se necesitan 2 evaluaciones de la función para saber si la misma es constante o balanceada. Generalizando, para n bits, serían necesarias 2^n evaluaciones (o visto de otra manera, analizar todas las combinaciones posibles de bits de entrada). La complejidad del algoritmo clásico sería por tanto de $O(2^n)$.

Para la evaluación de la función mediante un algoritmo cuántico, que se espera que sea más eficiente, se conoce que todas las puertas cuánticas empleadas han de ser reversibles. Analizando f(x), se concluye rápidamente que ésta no es reversible. Se debe evitar este problema mediante la introducción de un oráculo (vocablo inglés, en español se podría denominar también como «caja negra»). Dicho oráculo tendrá como entradas dos qubits y como salidas otros dos qubits. Las entradas serán denominadas como q0 y q1, y las salidas serán simplemente q0 y la operación XOR de q1 con f(x). En el mundo cuántico, la puerta XOR no existe, sin embargo sí que hay un equivalente, que no es otro que la puerta CNOT.

El algoritmo emplea dos de las propiedades del mundo cuántico antes introducidas y que tanto se emplean para esta clase de algoritmos: la superposición y la interferencia (en este caso no se hace uso de la tercera propiedad, el entrelazamiento cuántico.

Esquemáticamente, el algoritmo para construcción del circuito cuántico y que se ha codificado en python como se puede ver en el código anexo, es el siguiente:

1. Definir dos qubits, «x» e «y», o «q0» y «q1».
2. Invertir «y» o «q1» para tener un valor de 1 en dicho qubit (los qubit se inicializan por defecto a 0).
3. Aplicar superposición mediante la puerta H de Hadamard a ambos qubits.
4. Aplicar el oráculo.
5. Aplicar de nuevo puertas H a ambos qubits, mediante lo cual se estará aplicando interferencia para poder medir el resultado.
6. Medir la salida. El valor de q0 codifica el tipo de función, significando 0 que dicha función es constante y 1 que es balanceada.

Se puede observar como el oráculo sólo se aplica una vez. Las puertas Hadamard tienen una complejidad computacional despreciable respecto al oráculo, una vez que se escala el algoritmo. Se ha pasado de una complejidad exponencial, a una constante $O(1)$, siendo por tanto la mejora exponencial.

5.1.3. Definición del oráculo

La definición detallada del oráculo daría para otro estudio completo por separado. Tanto en este caso del algoritmo de Deutsch-Josza como en el siguiente de Grover, no se detallarán en demasía estas «cajas negras» y simplemente se hará uso de ellas, suponiendo conocida su forma. A la hora de codificar, la forma del oráculo se implementa conocido el tipo de función, balanceada o constante (esto es en realidad la solución del problema). Sin embargo, no se vuelve a consultar esta solución para obtener la respuesta. Esto permite centrarse en el algoritmo, tema principal de estudio de este trabajo. Para este caso de Deutsch-Josza, se conoce que para la entrada de dos qubits, se deja el primero de ellos igual y para el segundo se efectúa una XOR del mismo con la función a evaluar f(x). Se analizará si hay que hacer algún procesado adicional para cada uno de los cuatro casos enunciados en el punto anterior, los dos de la función constante y los dos de la función balanceada.

Con esta explicación, se procede a analizar las tablas de la verdad para los cuatro casos. Recuérdese que una de las salidas del oráculo es inmediata y coincide con el valor del primer qubit, por lo que no se muestra en las tablas para simplificar. Se estudia primero la construcción del oráculo para el caso de f(x) balanceada, como se muestra en el cuadro 3:

x(q0)	y(q1)	f(x)	y XOR f(x)
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	1

x(q0)	y(q1)	f(x)	y XOR f(x)
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	0

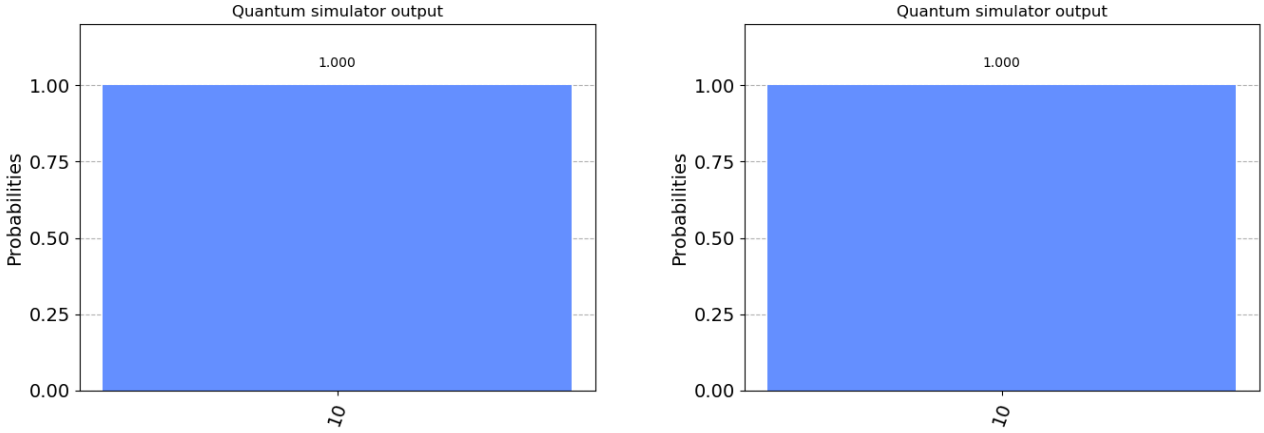
Cuadro 3: Tablas de la verdad del oráculo para el caso constante

Como se puede ver, en el primero de los casos la salida corresponde con y(q0), y en el segundo de ellos, su valor opuesto. La implementación mediante puertas cuánticas sería la que se puede observar en la figura 1. Por simplicidad, se muestra todo el circuito y no sólo el oráculo, que correspondería a lo que hay entre los separadores (barreras) segundo y tercero:

Figura 1: Circuitos cuánticos para el caso de una función constante

Las probabilidades obtenidas para cada uno de los casos se pueden ver en las gráficas de la figura 2 (las gráficas están en el mismo orden que los circuitos anteriores).

Figura 2: Probabilidades medidas para los circuitos cuánticos en el caso de una función constante



Desarrollando el caso análogo para una función balanceada, se llega a los valores mostrados en el cuadro 4:

x(q0)	y(q1)	f(x)	y XOR f(x)
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

x(q0)	y(q1)	f(x)	y XOR f(x)
0	0	1	1
0	1	1	0
1	0	0	0
1	1	0	1

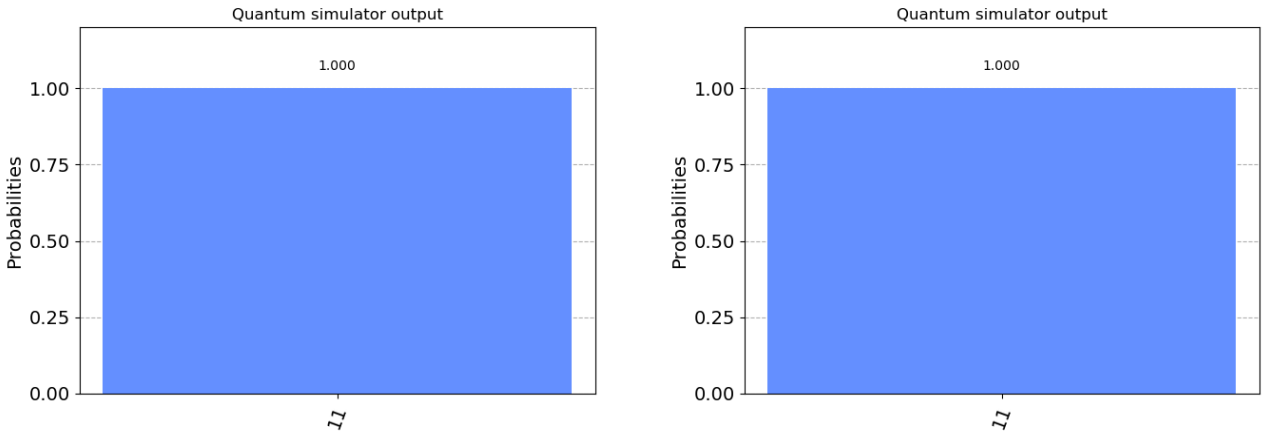
Cuadro 4: Tablas de la verdad del oráculo para el caso balanceado

La implementación utilizará las mencionadas puertas cuánticas CNOT, como se puede mostrar en la figura 3:

Figura 3: Circuitos cuánticos para el caso de una función balanceada

Y se pueden añadir las probabilidades calculadas, tal y como se hizo en el caso anterior, como se muestra en la figura 4, de nuevo siguiendo el mismo orden que el elegido para los circuitos:

Figura 4: Probabilidades medidas para los circuitos cuánticos en el caso de una función balanceada



5.2. Algoritmo de Grover

El algoritmo fue propuesto por Lov Grover en 1996. Se trata de un algoritmo para hallar un elemento en una lista, problema muy habitual en computación y cuya resolución clásica se estudia en cualquier curso básico de programación. Esto mismo se puede aplicar no sólo para búsquedas de una base de datos, sino para problemas de optimización (que también se basan en búsquedas de posibles soluciones), o estádisticas tales como la media, mediana, etc... Resulta intuitivo pensar que la complejidad de un algoritmo clásico de búsqueda en una lista, buscando elemento por elemento de la misma, es de $O(n)$. Se asume que la lista no tiene estructura y se ha tomado el peor caso posible (worst case scenario).

El algoritmo cuántico a estudiar reduce la complejidad al $O(\sqrt{N})$, produciendo una mejora cuadrática. Análogamente al caso estudiado anteriormente de Deutsch-Josza, el algoritmo se vuelve a basar en los principios de superposición e interferencia para acelerar la búsqueda. Se necesitarán $log_2 N$ qubits. Esquemáticamente, los pasos del algoritmo serían los siguientes:

1. Inicialización de los qubits a 0.
2. Aplicar una puerta H de Hadamard a cada uno de los qubits (superposición).
3. Aplicación de un oráculo a los qubits, y aplicación del algoritmo de difusión de Grover (interferencia).
4. Medición.

Los puntos 2 y 3 se pueden repetir, para los casos en los que el algoritmo precise de más de una iteración. Esto se verá con más claridad en alguno de los casos de la implementación.

5.2.1. Aplicación a dos qubits [6]

En primer lugar, se asigna igual probabilidad a cualquiera de los elementos de la lista, ya que no se presupone ninguna solución más probable que otra, lo cual se consigue mediante la superposición (segundo punto).

La mayor complejidad del algoritmo estará por tanto en el punto tercero. Respecto al oráculo («caja negra»), para este caso lo que hará será simplemente multiplicar la amplitud de la probabilidad (que no es más que la raíz cuadrada de la probabilidad) de la solución por un valor de (-1). Recuérdese que el oráculo conoce la solución (como se introdujo anteriormente, la construcción real del oráculo es un campo complicado en el que se están realizando muchos estudios al respecto). Analizando matemáticamente, y para un ejemplo de cuatro elementos en la lista (necesitándose por tanto dos qubits), con el elemento a buscar en la cuarta posición, se necesitaría pasar del vector de amplitudes de probabilidad: $\frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$, a este otro vector: $\frac{1}{2}\begin{pmatrix}1\\1\\1\\-1\end{pmatrix}$

Este se conseguiría aplicando una puerta cuya matriz fuera: $\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{pmatrix}$, matriz que coincide con la puerta Z controlada. Para el caso genérico de una posición distinta a la cuarta, se deberán manipular debidamente los qubits para obtener una matriz análoga con el -1 en otro punto de la diagonal principal, sin más que aplicar puertas X de Pauli (equivalente a la puerta NOT de los ordenadores clásicos) antes y después de la puerta Z controlada. Esta implementación se puede ver en el código.

Operando en el ejemplo anterior:

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

A partir de aquí, sólo faltaría el operador de difusión de Grover, que es el que se encarga de reflejar las amplitudes respecto de la media. La media de las amplitudes ahora es de 1. Si se refleja la amplitud respecto a esa nueva media, las tres primeras componentes del vector van a resultar nulas y la última la unidad, es decir, se pasa del vector $\frac{1}{2}\begin{pmatrix}1\\1\\1\\-1\end{pmatrix}$, al vector: $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$.

En este caso se puede ver como la medición nos daría la solución con una probabilidad del 100 % tras una única ejecución del algoritmo (recuérdese que $\sqrt{N}$ es un valor esperado para N suficientemente grande). En la figura (5) se puede ver el circuito cuántico correspondiente. Se han separado mediante barreras o separadores cada uno de los pasos del algoritmo: paso 2 al principio (los qubits ya están inicializados a cero), primera parte del paso 3 (oráculo) tras la siguiente barrera, algoritmo de difusión después de la siguiente barrera y por último la medición tras la última barrera (que se ha dibujado como doble en este caso):

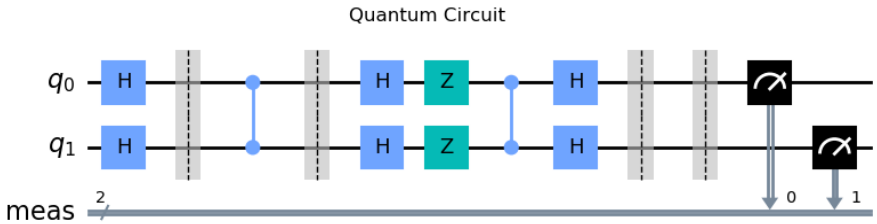


Figura 5: Circuito para el algoritmo cuántico de Grover con 2 qubits y solución «11»

Se adjunta también la distribución de probabilidades de la medición, que en este caso es del 100 %, como se puede ver en la figura 6:

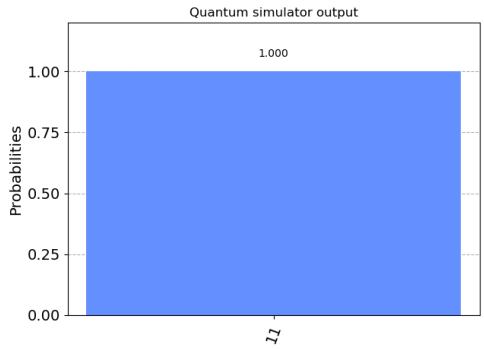


Figura 6: Probabilidad medida para el algoritmo cuántico de Grover con 2 qubits y solución «11»

Esta particularización para únicamente dos qubits se puede encontrar en muchos estudios. Por ello, y queriendo ampliar el algoritmo a un caso más complicado, se procede a implementar el mismo para tres qubits, lo que equivaldría a la búsqueda en una lista de ocho elementos.

5.2.2. Aplicación a tres qubits [7]

La base es la misma que en el caso anterior. Se estudirán dos casos distintos, el primero de ellos con una única solución (un único elemento a buscar), y el segundo con dos elementos distintos de la lista a buscar.

Para el primero de los casos, en la bibliografía se puede ver como la probabilidad de hallar el valor adecuado es de:

$$p = \left(\left[\frac{N-2t}{N} + \frac{2(N-t)}{N} \right] \frac{1}{\sqrt{N}} \right)^2 = \left(\frac{5}{4\sqrt{2}} \right)^2 = 0,78125 \tag{1}$$

En donde el parámetro «t» corresponde al número de soluciones a buscar. Comparado con el caso clásico, el mismo valor de probabilidad sería:

$$p' = \frac{t}{N} + \frac{N-t}{N} \cdot \frac{t}{N-1} = \frac{1}{8} + \frac{7}{8} \cdot \frac{1}{7} = 0,25$$

Se adjunta el circuito resultante para la medición del valor «000» en la figura 7. En el mismo se puede apreciar, gracias a la separación mediante barreras, cada una de las partes del algoritmo: en primer lugar, la superposición (antes de la primera barrera). A continuación, la parte del oráculo, hasta la siguiente barrera. Seguidamente, el algoritmo de difusión de Grover, hasta la doble barrera final, tras la cual se produce la medición.

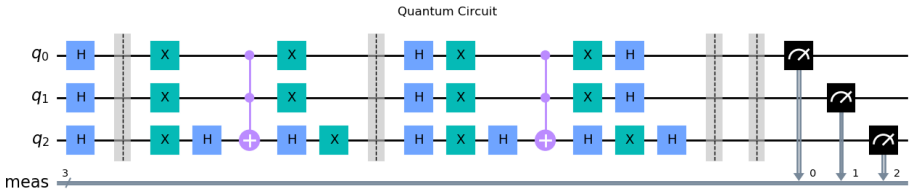


Figura 7: Circuito para el algoritmo cuántico de Grover con 3 qubits y solución «000»

Para este caso también se adjunta la probabilidad medida, como se puede ver en la figura 8:

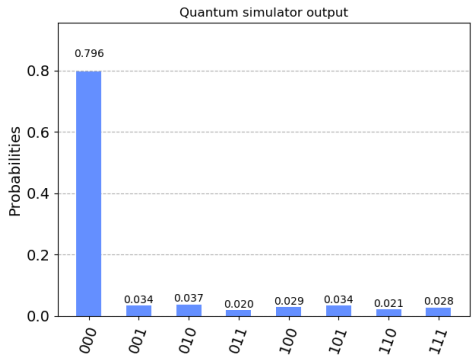


Figura 8: Probabilidad medida para el algoritmo cuántico de Grover con 3 qubits y solución «000»

Obviamente la probabilidad para el caso cuántico repitiendo el algoritmo una segunda vez mejorará, al tener $2 < \sqrt{N} < 3$. En el circuito cuántico correspondiente de la figura 9 se puede apreciar como hay una parte duplicada que corresponde a la segunda iteración. En este caso se está tratando de hallar la solución «111». Gracias a los separadores resulta fácil observar que la parte del oráculo y la del algoritmo de difusión están repetidas una vez, produciéndose la medición después de esta repetición.

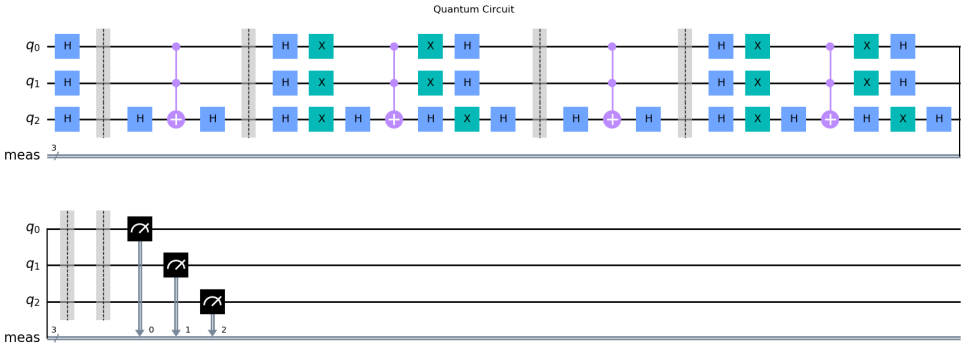


Figura 9: Circuito para el algoritmo cuántico de Grover con 3 qubits, doble iteración y solución «111»

Y efectivamente, la probabilidad aumenta en este caso:

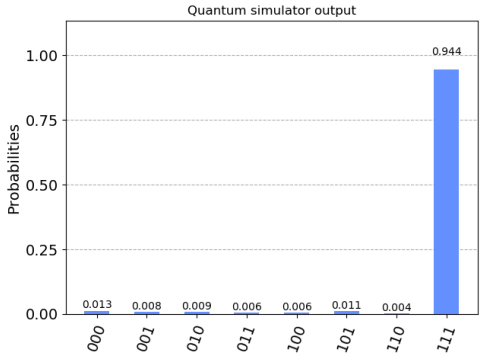


Figura 10: Probabilidad medida para el algoritmo cuántico de Grover con 3 qubits, doble iteración y solución «111»

Si se repitiera una tercera vez, el valor pasaría a empeorar (lo cual no parece intuitivo a primera vista, pero se basa en la parte del algoritmo de difusión de Grover, que refleja las amplitudes de probabilidad respecto de la media tras invertir previamente la del valor a buscar, tal y como se comentó en el punto anterior).

Para el caso de dos soluciones a buscar, $t = 2$, sustituyendo en (1), se obtiene $p = 1$ (probabilidad de encontrar al menos una de las dos soluciones), mientras que el mismo caso para el algoritmo clásico obtendría $p' = 0,464$. Por ello, en este caso no se ha permitido al algoritmo ejecutarse más de una vez.

Se puede ver un ejemplo de la ejecución del algoritmo en la figura 11:

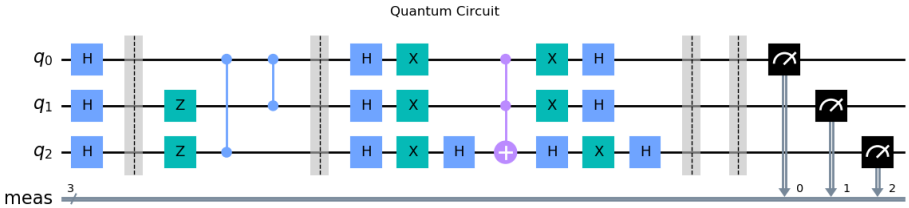


Figura 11: Circuito para el algoritmo cuántico de Grover con 3 qubits y soluciones «010» y «100»

Y las probabilidades asociadas:

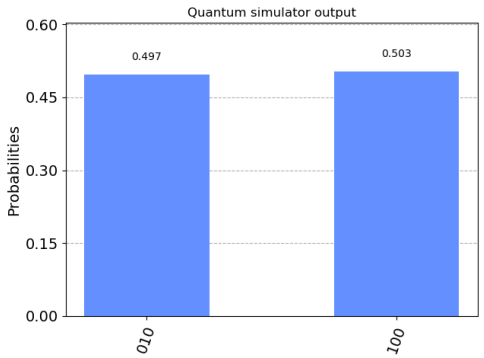


Figura 12: Probabilidad medida para el algoritmo cuántico de Grover con 3 qubits y soluciones «010» y «100»

Todos estos resultados han sido implementados en un simulador cuántico, esto es, una situación ideal y teórica.

Adicionalmente, se ha añadido la opción al programa implementado de ejecutar los distintos algoritmos en hardware cuántico real, gracias al empleo de una cuenta gratuita en el programa «IBM Quantum Experience». Al habilitar esta opción, lo único que se hace es añadir una gráfica adicional con el cálculo de probabilidades para este caso. Se ha de destacar que esta parte de la simulación tardará bastante más en ejecutarse, ya que el acceso al hardware cuántico es limitado y dicha simulación se añadirá a una cola de ejecución. El programa irá informando de la situación en la cola, y tras unos minutos estarán disponibles los resultados, que se discutirán en la siguiente sección.

6. Discusión

6.1. Discusión de los errores en el hardware real

Se estudia el caso del algoritmo de Grover para 2 qubits (como se podría haber estudiado cualquier otro). Se añade la opción para ejecución en hardware cuántico real. En primer lugar, se muestran los resultados teóricos, ya conocidos de la sección anterior (figura 6). Tras unos pocos minutos, se obtiene también la gráfica con la distribución de probabilidades del experimento realizado en dicho hardware cuántico, que se muestra en la figura:

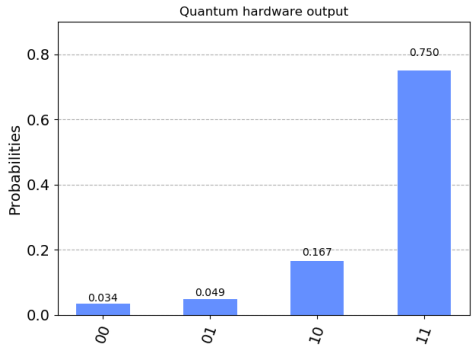


Figura 13: Probabilidad medida para el algoritmo cuántico de Grover con 2 qubits y solución «11», ejecutado en hardware cuántico real.

Parece lógico pensar que, en caso de existir errores en los resultados, sea más fácil que cambie un único qubit (resultados «01» ó «10»), que el caso de que ambos cambien a la vez (resultado «00»).

Cabe plantearse si algoritmos como los estudiados en este trabajo se pueden aplicar de una manera realista en la actualidad, y de no ser así, si se podrá en un futuro. Se puede adelantar, que hoy en día no se puede aplicar el algoritmo de Grover para una cantidad de datos significativos. Existen diversos problemas para ello, tales como:

- Insuficiencia de qubits (no se podrá aplicar una búsqueda en una lista demasiado grande)
- Ruido en los qubits (los resultados no serán fieles a la realidad)
- Imperfecciones en las puertas cuánticas (las salidas tras cada puerta no son las esperadas).

Al comenzar a estudiar los algoritmos, la mayoría de los resultados se obtuvieron en un simulador cuántico «ideal» (perfecto). Este caso no existe en la realidad, por lo que antes o después se planteará la pregunta de cómo elegir un equipo cuántico y cómo saber cómo de bueno es el mismo. Gracias a los criterios de DiVincenzo [8], esto resulta más fácil. Dicho criterio establece cinco condiciones para establecer la calidad de un ordenador cuántico:

1. Un sistema físico escalable y con qubits bien caracterizados. Esto es, qubits de un sistema con dos valores ($|0>$ y $|1>$), y cuyos propiedades (como podrían ser los valores de energía) están bien definidas. Un qubit con valor de $|1>$ decae a $|0>$ con el tiempo, según una gráfica de una exponencial caracterizada por un valor de tiempo de relajación, T_1: $p(|1>) = K \cdot e^{-t/T_1}$. La construcción de buenos ordenadores cuánticos se basa en tratar

de aislar los qubits todo lo posible, mediante el empleo de mejores materiales y reducción los mecanismos de pérdidas.

2. Posibilidad de inicializar el estado de los qubits (a uno de los valores expuestos en el punto anterior, siendo $|0>$ el valor más habitual).

3. Largos periodos de decoherencia (tiempo en el que el qubit preserva sus propiedades), mucho mayores que los tiempos de operación de las puertas cuánticas. Esto se caracteriza mediante un tiempo de decoherencia, llamado T_2 y que se puede relacionar con el tiempo de relajación mediante la expresión: $\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}$, siendo T_ϕ el tiempo de desfase. El resumen de estos tres puntos es tratar de aislar los qubits del entorno, bien reduciendo el ruido del mismo, bien haciendo a los qubits menos susceptibles a este ruido.

4. Empleo de un conjunto universal de puertas cuánticas, tales como H, S, T, CNOT. El control de la respuesta de dichas puertas es complicado. En la teoría una puerta va a suponer una transaformación en la esfera de Bloch a un punto en concreto (ver la sección correspondien en la introducción). Sin embargo, al tener dicha esfera un número infinito de puntos, las posibilidades de error son también infinitas. Resumidamente, hay mucho margen de error. Para cuantificar la calidad de las puertas cuánticas se empleará la fidelidad de puerta, que no es otra cosa que el complementario de la tasa de error: $f = 1 - \varepsilon$, siendo ε la tasa de error. La medición de la fidelidad es campo en activa investigación. Se podría pensar en una tasa de fidelidad $f = 0,995$ como una buena cifra. Sin embargo, suponiendo un circuito con 100 puertas con esta tasa de fidelidad, la fidelidad global sería de $f = 0,995^{100} = 0,6058 = 60,58\,\%$. Peor aún, para 200 puertas: $f = 0,995^{200} = 0,367 = 36,7\,\%$. Con «tan sólo» mejorar esta tasa de fidelidad de puerta a $f = 0,999$, los cálculos serían ahora: $f = 0,999^{100} = 90,48\,\%$ o $f = 0,999^{200} = 81,86\,\%$. Y un circuito cuántico con 200 puertas no es nada poco habitual.

5. Una capacidad de medida específica para qubits (si la medición no es adecuada, los resultados no valen para nada).

- Limitaciones de los algoritmos. Hardware. Estado del arte para la generación de qubits. Caracterización de los circuitos y puertas cuánticas. Quizás se pueda mover a conclusiones.

7. Conclusiones

8. Anexos

8.1. Código fuente

Disponible en Github: https://github.com/raulillo82/TFG-Fisica-2021
Listado de archivos:

```
license.txt
d-j.py
grover.py
```

9. Agradecimientos [9]

La idea de este trabajo de fin de grado nació al cursar durante el curso 2020/21 el programa «Introduction to Quantum Computing with IBM Quantum», de «Qubit x Qubit». Se han tomado partes de dicho material a modo de resumen teórico, y la idea de los algoritmos a plantear estaba propuesta en él. Por ejemplo, se hacía una implementación básica del algoritmo de Grover para 2 qubits con una solución fija, y en este trabajo se ha generalizado para cualquier solución. En el caso de 3 qubits, sólo se estudiaba el caso de solución única con una única iteración y con una solución concreat. Tras consultar bibliografía adicional, se implementó para cualquier posible solución única en una o dos iteraciones, así como para una solución doble, tal y como se ha explicado en la sección de resultados.

10. Bibliografía

Referencias

[1] Qiskit homepage: https://qiskit.org/

[2] Wikipedia: Qiskit (https://en.wikipedia.org/wiki/Qiskit)

[3] IBM Quantum: https://quantum-computing.ibm.com/

[4] Github: https://github.com

[5] Lyx: https://www.lyx.org/

[6] Qiskit reference: Grover's Algorithm (https://qiskit.org/textbook/ch-algorithms/grover.html)

[7] Nature: Complete 3-Qubit Grover search on a programmable quantum computer (https://www.nature.com/articles/s41467-017-01904-7)

[8] Wikipedia: DiVincenzo's criteria (https://en.wikipedia.org/wiki/DiVincenzo%27s_criteria)

[9] Qubit x Qubit: Programs (https://www.qubitbyqubit.org/programs)