Índice general

Capítulo 1

Introducción

1.1 Introducción a la computación cuántica

1.1.1 Mecánica cuántica

La naturaleza se comporta de una forma sorprendentemente inesperada a medida que la escala a la que se observan los detalles se hace cada vez más pequeña. Resulta muy extraño describir dicho comportamiento en comparación con las cosas que estamos acostumbrados a ver en el día a día. Sin embargo, existe una serie de reglas que se han ido descubriendo, y que siempre se cumplen (hasta ahora lo han hecho). Además estas reglas son excepcionalmente simples, y pueden ser descritas mediante las fórmulas matemáticas que hasta ahora conocemos. Este conjunto de reglas se conocen como los postulados de la mecánica cuántica.

Una de las ventajas de poder describir el comportamiento de algún suceso, es la posibilidad de emplearlo de forma provechosa. Por ejemplo, saber que una substancia impide el crecimiento de una bacteria infecciosa permite erradicar una enfermedad—es el caso de la penicilina.

La mecánica cuántica puede emplearse para describir sistemas cuidadosamente diseñados para que se comporten como un mecanismo de cálculo. Actualmente los ordenadores emplean nuestro conocimiento de las leyes físicas como el electromagnetismo para realizar cómputos. Del mismo modo, se pueden emplear sistemas cuánticos dando lugar al término ordenador cuántico. La idea de construir tal dispositivo ha sido propuesta por primera vez por Richard Feynman en 1982 [5], con el objetivo de simular sistemas cuánticos.

1.1.2 Computación cuántica

La computación cuántica es un nuevo paradigma de computación, que busca la resolución de problemas empleando las propiedades de la mecánica cuántica. Fundamenta su utilidad en el hecho de que algunos problemas que tienen una alta complejidad en los ordenadores clásicos, y son intratables, pasan a ser tratables en un ordenador cuántico.

Peter Shor descubrió en 1994 [10] como resolver el problema de factorización de un número compuesto en el producto de dos primos, como $15 = 3 \cdot 5$. La complejidad de este cómputo es tan alta para números elevados, que se considera como base para algunos métodos de criptografía de como RSA. Se supone que nadie sería capaz de encontrar esos dos números primos en mucho tiempo. Pero empleando un ordenador cuántico, el tiempo se reduce lo suficiente como para encontrarlos. De modo que la criptografía tendría que buscar otros métodos.

La factorización de números mediante algoritmos cuánticos se ha llevado a cabo de forma experimental, siendo el número 56153 el más grande hasta la fecha [3].

Otro algoritmo cuántico fue descubierto en 1996 por Lov Grover [6] para encontrar un elemento en una lista desordenada de tamaño n con una probabilidad de 1/2. Mejorando el tiempo empleado por el mejor algoritmo probabilístico conocido de $O(n)$ a $O(\sqrt{n})$.

Una de las dificultades a la hora de diseñar un algoritmo cuántico es la gran diferencia de comportamiento, comparado con un algoritmo convencional o clásico. Nuestra intuición y formas actuales para diseñar algoritmos, fallan al tratar de comprender el proceso. Este es quizás uno de los motivos por los cuales no se han descubierto muchos algoritmos cuánticos [11].

1.2 El bit y el qubit

La forma de almacenar la información en un ordenador cuántico difiere en algunos aspectos a la forma en la que se almacena actualmente en los ordenadores convencionales.

1.2.1 Bit clásico

En los ordenadores actuales, la unidad básica de representación de información es el bit. La palabra bit significa dígito binario, que puede ser 0 o 1. Es el nexo de unión entre los sistemas físicos que implementan un bit, y una lógica binaria, que se abstrae de su implementación.

La forma en la que se implementa un bit depende de la arquitectura: la posición de una leva mecánica en la máquina, la existencia de una presilla en una cinta de vídeo (o también de cinta adhesiva), el estado de un interruptor, la presencia o ausencia de un agujero en una tarjeta perforada, anillos de ferrita que se magnetizan en un sentido o en otro, dos niveles de voltaje diferentes...

Todas estas representaciones, diferentes en su naturaleza física, comparten dos propiedades en común; es posible representar dos estados diferentes, y además es posible modificar y leer el estado en el que se encuentra el sistema.

De esta forma es importante distinguir entre la *representación física* y el *significado lógico*. En el caso de la tarjeta perforada, esta relación puede darse de la siguiente forma: Si hay un agujero, entonces representa un 0; en caso contrario, un 1. Para representar un bit, se empleará un estado $x \in \{0, 1\}$.

1.2.2 Bit probabilístico

Una moneda perfecta que se lanza al aire, y que cae sobre una superficie lisa, saca cara con igual probabilidad que cruz. Mientras se encuentra en el aire dando vueltas, su estado no es ni cara ni cruz. Es un estado diferente. Este estado puede representarse mediante un bit probabilístico, de forma que la cara es el 0, la cruz es el 1, y ambos tienen una probabilidad de 0.5 de ocurrir. Escribiendo las probabilidades en un vector, se define

$$y = \begin{pmatrix} 0{,}5 \\ 0{,}5 \end{pmatrix}$$

Comenzando la numeración en 0, el primer elemento $y[0]$ representa la probabilidad de que el sistema saque un 0. Y $y[1]$ de que saque un 1. El hecho de que el índice coincida con el estado que representa, será útil en el futuro.

Además, debido a que se trata de un sistema probabilístico, ambas probabilidades han de sumar la unidad.

$$y[0] + y[1] = 1$$

Una vez que la moneda cae, su estado deja de ser un bit probabilístico y, y se convierte en un bit clásico x. Esta operación es la *medición* de un sistema probabilístico.

1.2.3 Bit cuántico o qubit

Hasta ahora, la clase de sistemas que se han descrito, son conocidos por experiencias en la vida cotidiana o profesional, y sirven de analogías. Sin

embargo, el funcionamiento del sistema que se describirá a continuación no tiene un ejemplo conocido, tan sólo la imaginación será capaz de construir dicho sistema.

Un sistema cuántico puede encontrarse en un estado z, definido como

$$z = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Además, tanto α como β son números complejos. Este estado se denomina bit cuántico o qubit. Tras medir un qubit, el sistema se convertirá en un bit clásico x, con una probabilidad de que salga 0 de $|\alpha|^2$ y de que salga 1 de $|\beta|^2$. Es decir, que se comportará como un bit probabilístico con las probabilidades

$$y = \begin{pmatrix} |\alpha|^2 \\ |\beta|^2 \end{pmatrix}$$

Y de igual modo, cumple la restricción de que ambas probabilidades suman la unidad:

$$|\alpha|^2 + |\beta|^2 = 1$$

Los números α y β se denominan *amplitudes*, y tienen asociada una probabilidad que es $|\alpha|^2$ y $|\beta|^2$ respectivamente.

1.3 Múltiples qubits

Un sistema de dos monedas perfectas A y B tiene exactamente cuatro posibles resultados tras lanzarlas al aire $\{00, 01, 10, 11\}$. Siendo cara el 0, cruz el 1, y la cadena 01 que A sale cara y B sale cruz.

El número de resultados posibles aumenta con el número de monedas n de la forma 2^n. Si ambas monedas se describen como un bit probabilístico, se obtiene

$$y_A = \begin{pmatrix} 0{,}5 \\ 0{,}5 \end{pmatrix} \quad y_B = \begin{pmatrix} 0{,}5 \\ 0{,}5 \end{pmatrix}$$

Siendo y_m la descripción del estado de la moneda m. Entonces, la probabilidad de obtener cada uno de los cuatro diferentes estados al lanzar ambas, se puede describir en forma de vector. Sea $y_m[k]$ la probabilidad de que la moneda m salga k, entonces la probabilidad de que salga k_A y luego k_B será:

$$y_A[k_A] \cdot y_B[k_B]$$

De modo que se puede construir un vector que describa todas las posibilidades:

$$y = \begin{pmatrix} y_A[0] \cdot y_B[0] \\ y_A[0] \cdot y_B[1] \\ y_A[1] \cdot y_B[0] \\ y_A[1] \cdot y_B[1] \end{pmatrix} = \begin{pmatrix} y_A[0] \cdot y_B \\ y_A[1] \cdot y_B \end{pmatrix}$$

Esta operación se conoce como el producto tensorial y conocer sus propiedades será fundamental.

1.3.1 Producto tensorial

La operación $\otimes$ se define como el producto tensorial o producto de Kronecker. Si A y B son dos matrices de $n \times m$ y $k \times l$ respectivamente:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & \cdots & b_{1l} \\ \vdots & & \vdots \\ b_{k1} & \cdots & b_{kl} \end{pmatrix}$$

Entonces $C = A \otimes B$ es la matriz C de $nk \times ml$, definida como

$$C = A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{pmatrix}$$

El producto tensorial cumple varias propiedades interesantes.

- Es asociativo $(A \otimes B) \otimes C = A \otimes (B \otimes C)$
- Es distributivo respecto a la suma $A \otimes (B + C) = (A \otimes B) + (A \otimes C)$ y $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$
- Es distributivo respecto a la multiplicación de matrices $A \otimes (B \cdot C) = (A \otimes B) \cdot (A \otimes C)$ y $(A \cdot B) \otimes C = (A \otimes C) \cdot (B \otimes C)$
- Para un escalar k, cumple $(kA) \otimes B = A \otimes (kB) = k(A \otimes B)$

Pero en general, no es conmutativo: $A \otimes B \neq B \otimes A$.

1.3.2 Monedas y qubits

Empleando la notación de producto tensorial, ahora calcular el estado global de un sistema de dos monedas, se simplifica a

$$y = y_A \otimes y_B$$

Este mismo procedimiento se emplea en la descripción de qubits. Un sistema z de dos qubits z_A y z_B se puede describir como:

$$z_A = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix}, \quad z_B = \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix}, \quad z = z_A \otimes z_B$$

De forma que se obtiene:

$$z = \begin{pmatrix} \alpha_1\alpha_2 & \alpha_1\beta_2 & \beta_1\alpha_2 & \beta_1\beta_2 \end{pmatrix}^T$$

Por ahora no hay ninguna diferencia apreciable en la forma en la que comporta un sistema cuántico frente a uno probabilístico. Pero no será durante mucho tiempo. El sistema de dos monedas y está sujeto a la restricción de que al lanzar una moneda la suma de las probabilidades de los posibles resultados debe ser la unidad, esto es

$$y_A[0] + y_A[1] = 1, \quad y_B[0] + y_B[1] = 1$$

Cuando dos qubits se tratan por separado, esta restricción es análoga:

$$|\alpha_1|^2 + |\beta_1|^2 = 1, \quad |\alpha_2|^2 + |\beta_2|^2 = 1$$

Pero cuando se combinan en un sólo sistema cuántico, pasa a denominarse *registro cuántico*, y la restricción se convierte en:

$$\sum_{\alpha \in z} |\alpha|^2 = 1 \tag{1.1}$$

Permitiendo el caso en el que puede existir un registro de dos qubits con el estado:

$$z = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}^T$$

Que cumple la restricción (1.1) puesto que:

$$\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} + \frac{1}{2} = 1$$

Y sin embargo, z no puede ser expresado como el producto tensorial de dos qubits z_A y z_B:

$$z = z_A \otimes z_B = \begin{pmatrix} \alpha_1\alpha_2 & \alpha_1\beta_2 & \beta_1\alpha_2 & \beta_1\beta_2 \end{pmatrix}^T = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}^T$$

Se obtiene que $\alpha_1\alpha_2 = \frac{1}{\sqrt{2}}$, por lo que $\alpha_1 \neq 0$, y $\beta_1\beta_2 = \frac{1}{\sqrt{2}}$ entonces $\beta_2 \neq 0$ pero sin embargo $\alpha_1\beta_2 = 0$, lo cual es imposible.

Este extraño suceso no existe en el mundo clásico, y no se puede realizar con las monedas. Es único del comportamiento cuántico. Se denomina *entrelazamiento cuántico* cuando un sistema cuántico compuesto no puede ser descrito mediante el producto de sus constituyentes. Se dice que se encuentra entrelazado.

1.4 Notación de Dirac o bra-ket

A medida que aumenta el número de qubits de un registro cuántico, se hace cada vez más difícil describir los 2^n componentes que requiere dicho vector.

El matemático Hermann Grassmann empleó la notación $[u|v]$ en 1862 para indicar el producto interno de dos vectores [2]. Posteriormente, en 1939 el físico Paul Dirac empleó $\langle u|v\rangle$ para describir estados cuánticos, y dicha notación se ha extendido debido a su utilidad. Se denomina braket a $\langle u|v\rangle$, formado por el bra $\langle u|$ y el ket $|v\rangle$. De esta forma, se definen los kets

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

Y entonces un qubit z puede describirse como una combinación lineal de kets $|0\rangle$ y $|1\rangle$

$$z = \begin{pmatrix}\alpha\\\beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$$

Dado que cualquier qubit puede escribirse partiendo de los kets $|0\rangle$ y $|1\rangle$, forman una base vectorial y se les denomina vectores base, *kets base* o también *estados puros*. De forma general, se define un estado cuántico o registro de n qubits como el ket $|\psi\rangle$ con $X = \{0,1\}^n$:

$$|\psi\rangle = \sum_{x\in X} \alpha_x\,|x\rangle$$

Por ejemplo para $n = 2$, se tiene $X = \{00, 01, 10, 11\}$, y el ket $|\psi\rangle$:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle = \begin{pmatrix}\alpha_{00} & \alpha_{01} & \alpha_{10} & \alpha_{11}\end{pmatrix}^T$$

El conjunto de kets $|x\rangle$ para todo $x \in X$ forma de nuevo una base de vectores, para el registro $|\psi\rangle$. Al escribir un ket como $|01\rangle$ se hace referencia al producto tensorial:

$$|01\rangle = |0\rangle \otimes |1\rangle$$

Y también se puede escribir sin el símbolo $\otimes$, como $|0\rangle\,|1\rangle$. De igual forma, para estados más grandes como $|0110\rangle = |0\rangle\,|1\rangle\,|1\rangle\,|0\rangle$.

1.4.1 Superposición

Cuando un qubit o un registro se encuentra en un estado diferente a un estado puro, por ejemplo el ket

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$$

se dice que se encuentra en *superposición*. El qubit se comporta como si se encontrase en ambos estados $|0\rangle$ y $|1\rangle$ a la vez, y cuando se observa, colapsa en un estado puro. A diferencia de una moneda que se lanza, existe la forma de determinar el estado de la moneda a medida que se mueve por el aire. Con una cámara que grabe a una alta velocidad, se puede observar como la moneda da vueltas oscilando entre cara y cruz. Sin embargo no existe ninguna forma actual para determinar el estado interno de un qubit o registro.

Adicionalmente, uno de los puntos clave de la computación cuántica, se basa en que es posible operar con ambos estados a la vez, obteniendo un paralelismo que crece de forma exponencial con el número de qubits. Un sistema de n qubits puede realizar cálculos operando directamente con las 2^n amplitudes asociadas a cada estado puro. En un sistema clásico, se requieren 2^n registros de bits para alcanzar un cálculo similar.

1.5 Operaciones

Las operaciones que se pueden realizar sobre un qubit se representan como una matriz A de 2×2 elementos complejos y se denominan *operadores*.

Además, todos los operadores cuánticos deben ser *reversibles*, es decir que la matriz A debe ser *unitaria*, cumpliendo que A por su transpuesta conjugada $A^\dagger$ sea la identidad:

$$AA^\dagger = A^\dagger A = I$$

Cuando se aplica un operador A sobre un qbit $|\psi\rangle$, se obtiene un estado resultante $|\psi'\rangle$ obtenido por el producto matricial

$$|\psi'\rangle = A\,|\psi\rangle$$

A los operadores cuánticos también se les denomina *puertas cuánticas*, por su parecido con las puertas lógicas. Por ejemplo la puerta lógica NOT, invierte el estado de un bit, y su correspondiente cuántica es

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Que aplicada sobre los estados puros los invierte.

$$NOT\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$NOT\,|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

Y sobre un estado en superposición $|\psi\rangle$, invierte sus amplitudes

$$NOT\,|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

1.5.1 Operadores de varios qubits

Además de poder aplicar un operador sobre un qubit, es posible hacerlo sobre un registro de varios qubits. En concreto, para n qubits, la matriz A que describe al operador contiene $2^n \times 2^n$ filas y columnas.

Una forma de construir operadores grandes es aplicando el producto tensorial de varios operadores simples. Por ejemplo, el operador que invierte de igual forma que el NOT un registro de dos qubits, se calcula como

$$NOT \otimes NOT = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Que al aplicarse sobre el estado $|01\rangle$ se convierte en:

$$\begin{aligned} (NOT \otimes NOT)\,|01\rangle &= (NOT \otimes NOT)(|0\rangle \otimes |1\rangle) = \\ &= NOT\,|0\rangle \otimes NOT\,|1\rangle = |1\rangle \otimes |0\rangle = |10\rangle \end{aligned}$$

Obteniéndose el estado invertido $|10\rangle$. La operación $NOT \otimes NOT$ que consiste en multiplicar tensorialmente un operador por si mismo, es equivalente a la potencia tensorial de operadores, y se indica como

$$NOT \otimes NOT = NOT^{\otimes 2}$$

De forma que se puede extender para cualquier exponente.

$$NOT^{\otimes n} = \underbrace{NOT \otimes NOT \otimes \ldots \otimes NOT}_{n \text{ veces}}$$

Operadores comunes

Existen una serie de operadores que se emplean con mucha frecuencia en la computación cuántica. Uno de los más importantes es el operador de *Hadamard*, representado por la letra H.

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

Tiene la peliculiaridad de descomponer el estado puro $|0\rangle$ en un estado superpuesto:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

El operador de Hadamard para n qubits $H^{\otimes n}$, aplicado sobre un estado $|x\rangle$, se puede generalizar de la forma

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_z (-1)^{\langle x|z\rangle}|z\rangle$$

Donde $\langle x|z\rangle$ representa el producto interno de x y z, bit a bit módulo dos:

$$\langle x|z\rangle = x_1z_1 + x_2z_2 + \ldots + x_nz_n \quad \text{mód } 2$$

El operador identidad I deja un estado tal y como estaba, no realiza ninguna acción, y se define

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

Otros son el desplazamiento de fase R_θ, la puerta SWAP que intercambia dos qubits, la puerta not controlada CNOT que realiza la negación del segundo qubit cuando el primero es $|1\rangle$.

$$R_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}, \quad SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

1.5.2 Varias operaciones

Cuando se realizan varias operaciones seguidas sobre un estado $|\psi\rangle$, el resultado es el producto de las matrices de los operadores por el estado. Siguiendo el orden en el que fueron aplicados. Por ejemplo, aplicar el operador de Hadamard sobre el estado $|0\rangle$ y posteriormente el NOT, se representa

$$NOT(H|0\rangle) = NOT\ H|0\rangle$$

El orden viene dado por el producto de matrices, de derecha a izquierda. Y se calcula:

$$NOT\ H|0\rangle = NOT\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

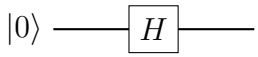
Circuitos cuánticos

Los *circuitos cuánticos* son agrupaciones de operadores aplicados en un orden determinado, que tienen un propósito. Se pueden describir como una matriz resultante de multiplicar todas las matrices que operan de forma sucesiva.

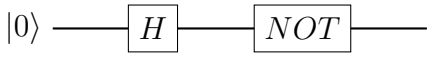
A medida que se combinan los operadores, es posible crear complejas estructuras que son difíciles de comprender empleando la notación de producto de matrices. Los circuitos cuánticos se describen habitualmente con diagramas que incluyen varios elementos.

Las líneas horizontales representan qubits o registros de qubits. Las puertas cuánticas se colocan intersecando las líneas que representan los qubits en los que actúan.

Por ejemplo el operador de Hadamard, aplicado sobre el qubit $|0\rangle$, se representa con el siguiente circuito.

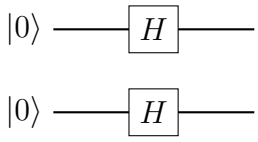


Si se aplica primero el operador de Hadamard y luego la puerta NOT, se obtiene

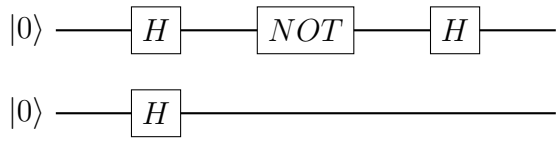


También se pueden realizar operaciones en *paralelo*. En un sistema de dos qubits, en el que el que se aplica un operador de forma simultánea en dos líneas, corresponde con el producto tensorial.

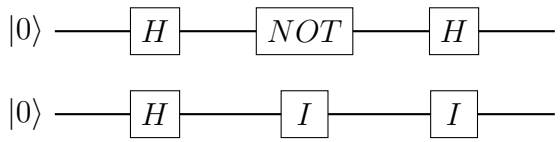
En el caso de aplicar el operador de Hadamard sobre los dos qubits del registro $|00\rangle$ se representa



Y corresponde con el resultado de calcular $(H \otimes H)\,|00\rangle$. De la misma forma, combinando operaciones secuenciales y en paralelo, se pueden construir circuitos complejos, que se representan de forma sencilla.

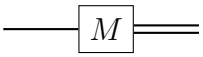


Que es equivalente a $(H \otimes H)(NOT \otimes I)(H \otimes I)\,|00\rangle$. Se observa como las líneas son equivalentes a aplicar el operador identidad I.



Operadores de medición

Existe una operación que es diferente de todas las demás puertas cuánticas. Se trata de los operadores de medición, representados por la letra M, o otras veces por el dibujo de un aparato de medida analógico.



Este operador realiza la medición de una línea, y produce un resultado binario. Destruyendo el estado cuántico en el proceso de medida. El resultado de medir un qubit será un bit. En el caso de medir un registro, será un resultado binario correspondiente a sus estados puros. Las dos líneas paralelas que salen del circuito tras el operador M representan una línea de información clásica.

No puede ser representado por una matriz unitaria, ya que el resultado no es reversible. Una vez realizada una medición, el estado cuántico se colapsa. Una de las interesantes cuestiones de los estados entrelazados como el estado de Bell denominado por $|\beta_{00}\rangle$:

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Es que es posible realizar una medida sobre uno de sus qubits. El resultado será 0 o 1 con una probabilidad del 50 %. Inmediatamente al conocer el estado de uno de los qubits, el otro también se conoce. Este hecho, que intrigó a Einstein, Podolsky y Rosen [1], produjo la formulación de la paradoja EPR, en la que se discutía como era posible que al realizar la medición de un qubit, el otro inmediatamente se alterase aunque ambos qubits estuvieran separados.

Este hecho, ha sido comprobado de forma experimental con resultados positivos, contradiciendo totalmente la intuición. Además es posible aprovechar sus consecuencias para "teletransportar" información cuántica a través de un canal clásico de bits.

Capítulo 2

Circuitos cuánticos

El objetivo de los circuitos cuánticos es el de aprovechar las propiedades que otorga la computación cuántica, para resolver problemas de forma más rápida y eficiente que los algoritmos convencionales. El estudio de sus propiedades permite determinar como aprovecharlas.

2.1 Técnicas fundamentales

La mayoría de los circuitos emplean alguna de estas técnicas básicas, en las que se apoya la aceleración de la computación cuántica.

2.1.1 Paralelismo

Una de las técnicas en las que se basan los algoritmos cuánticos para acelerar su ejecución, consiste en la posibilidad de ejecutar muchas acciones a la vez. Esta característica se debe a la propiedad que posee un estado cuántico de encontrarse en superposición. Por ejemplo el estado

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Para observar como se puede hacer uso de esta propiedad, se diseñará un problema de ejemplo [8]. Sea $f(x) : \{0,1\} \rightarrow \{0,1\}$ una función binaria como la puerta clásica NOT. Entonces, una forma de implementarla en un operador cuántico U_f es

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

Al aplicar el operador a un registro de dos qubits, al segundo registro se le suma (módulo 2) el resultado de f sobre el primero. Por ejemplo, partiendo

de un estado $|1\rangle\,|0\rangle$, y aplicando el operador U_f, se obtiene

$$U_f\,|1\rangle\,|0\rangle = |1\rangle\,|f(1)\rangle$$

Esta extraña forma de computar f, está causada por la restricción de que los operadores cuánticos (como U_f) deben ser unitarios, para permitir una computación reversible.

Una vez obtenido el operador U_f, que dependerá de la función en cuentión, (pero será una matriz de 4×4 igualmente) es posible emplear un estado superpuesto como $|\psi\rangle$ como entrada de la función, colocándolo en $|x\rangle$. El otro qubit $|y\rangle$ queda como ket cero $|0\rangle$. El operador U_f actuará sobre el estado $|\psi\rangle\,|0\rangle$, produciendo:

$$|\psi_1\rangle = U_f\,|\psi\rangle\,|0\rangle = U_f\frac{1}{\sqrt{2}}(|0\rangle\,|0\rangle + |1\rangle\,|0\rangle) = \frac{1}{\sqrt{2}}(U_f\,|0\rangle\,|0\rangle + U_f\,|1\rangle\,|0\rangle)$$

Lo que está ocurriendo es que el operador U_f está siendo aplicado simultáneamente sobre dos estados, $|0\rangle\,|0\rangle$ y $|1\rangle\,|0\rangle$, obteniendo finalmente:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle\,|f(0)\rangle + |1\rangle\,|f(1)\rangle)$$

En este estado, se observa como la función f se evalúa simultáneamente en 0 y en 1. De forma que en un sólo paso, se realizan dos evaluaciones de f. Esta propiedad se conoce con el nombre de *paralelismo cuántico*. Sin embargo, no es posible observar los estados entrelazados, ya que el proceso de medición hace que se colapsen a uno de ellos. Es necesario realizar más operaciones para poder extraer la información deseada.

2.1.2 Interferencia

El efecto de interferencia proviene de las ondas, pues dos ondas con la misma frecuencia con la misma fase se suman, pero con fases opuestas se destruyen. Esta propiedad, presente en el comportamiento cuántico, permite aprovechar los estados superpuestos computados en paralelo, y transformarlos para poder obtener información útil de todos ellos. Un ejemplo, consiste en una función binaria $f : \{0,1\} \to \{0,1\}$, a la que se desea determinar si $f(0) = f(1)$ o si por el contrario son diferentes. Una solución clásica es determinar $f(0)$, luego $f(1)$, y luego comparar los valores. Pero requiere evaluar *secuencialmente* la función dos veces. Una alternativa es emplear un circuito cuántico. Partiendo de un estado inicial de dos qubits $|\psi_0\rangle = |01\rangle$, y del mismo operador U_f anteriormente descrito, el objetivo ahora será determinar que ocurre cuando se ejecuta la operación:

$$(H \otimes I)(U_f)(H \otimes H)\,|\psi_0\rangle$$

Este circuito se llama algoritmo de Deutsch, y fue diseñado para mostrar las capacidades de la computación cuántica [4]. Primero, el operador de Hadamard actúa sobre ambos qubits

$$|\psi_1\rangle = (H \otimes H)(|0\rangle \otimes |1\rangle) = (H\,|0\rangle \otimes H\,|1\rangle)$$

Transformándolos a un estado superpuesto

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Expresado conjuntamente como

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)$$

Que se puede simplificar a

$$|\psi_1\rangle = \frac{1}{2}\left(\,|0\rangle \otimes |0\rangle - |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle\,\right)$$

Finalmente el operador U_f, definido como $U_f\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle$, se aplica sobre $|\psi_1\rangle$, transformándolo en

$$\begin{aligned}|\psi_2\rangle = \frac{1}{2}\Big(\,&|0\rangle \otimes |0 \oplus f(0)\rangle - |0\rangle \otimes |1 \oplus f(0)\rangle + \\ &|1\rangle \otimes |1 \oplus f(1)\rangle - |1\rangle \otimes |1 \oplus f(1)\rangle\,\Big)\end{aligned}$$

Que se puede expresar como

$$\begin{aligned}|\psi_2\rangle = \frac{1}{2}\Big(\,&|0\rangle \otimes (|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle + \\ &|1\rangle \otimes (|1 \oplus f(1)\rangle - |1 \oplus f(1)\rangle\,\Big)\end{aligned}$$

Y observando el hecho de que

$$|0 \oplus k\rangle - |1 \oplus k\rangle = (-1)^k(|0\rangle - |1\rangle)$$

Para $k = f(0)$ y $k = f(1)$, se simplifica en

$$\begin{aligned}|\psi_2\rangle = \frac{1}{2}\Big((-1)^{f(0)}\,&|0\rangle \otimes (|0\rangle - |1\rangle) + \\ &(-1)^{f(1)}\,|1\rangle \otimes (|0\rangle - |1\rangle)\Big)\end{aligned}$$

Y finalmente, en

$$|\psi_2\rangle = \frac{1}{2}\Big((-1)^{f(0)}\,|0\rangle + (-1)^{f(1)}\,|1\rangle\Big) \otimes \Big(|0\rangle - |1\rangle)\Big)$$

Este es un estado muy interesante. Para identificar lo que ocurre, es necesario realizar una observación. Sea $a = (-1)^{f(0)}$ y $b = (-1)^{f(1)}$, se reescribe

$$|\psi_2\rangle = \frac{1}{2}\Big(a\,|0\rangle + b\,|1\rangle\Big) \otimes \Big(|0\rangle - |1\rangle)\Big)$$

A partir de aquí, solo se empleará el primer qubit, de modo que se divide el estado $|\psi_2\rangle$ en sus dos qubits $|\psi_2^1\rangle \otimes |\psi_2^2\rangle$, que son

$$|\psi_2^1\rangle = \frac{a\,|0\rangle + b\,|1\rangle}{\sqrt{2}} \qquad |\psi_2^2\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Ahora el operador de Hadamard se aplica sobre el primer qubit, $|\psi_3^1\rangle = H\,|\psi_2^1\rangle$, obteniendo

$$|\psi_3^1\rangle = \frac{a\,|0\rangle + a\,|1\rangle + b\,|0\rangle - b\,|1\rangle}{2} = \frac{a+b}{2}\,|0\rangle + \frac{a-b}{2}\,|1\rangle$$

Si $a = b = \pm 1$, el coeficiente del ket $|0\rangle$ se sumará, quedando ± 1. Es el denominado efecto de inferferencia constructiva. El ket $|1\rangle$ por el contrario sufrirá el efecto opuesto, llamado interferencia destructiva, pues $(a - b) = 0$. De forma que el estado se transformará en $\pm\,|0\rangle$.

Por el otro lado, si $a = -b = \pm 1$, sucederá el proceso inverso. El coeficiente del ket $|0\rangle$ se destruirá, $(a+b) = (a-a) = 0$, y el del ket $|1\rangle$ se sumará constructivamente $(a - b) = (a + a) = \pm 1$, obteniéndose $\pm\,|1\rangle$. Resultando los dos casos:

$$|\psi_3^1\rangle = \begin{cases} \pm\,|0\rangle\,, & \text{si } a = b \\ \pm\,|1\rangle\,, & \text{si } a = -b \end{cases}$$

Ahora si se mide este qubit, y se obtiene un 0, se deduce que $a = b$. Si se obtiene un 1, que $a = -b$. Esta información permite determinar si $f(0) = f(1)$, cuando $a = b$, o si $f(0) \neq f(1)$ cuando $a = -b$. El paralelismo cuántico permite evaluar la función f de forma simultánea, y el efecto de la interferencia, transformar ese estado superpuesto, de forma que se obtenga un 0 o un 1 correspondiente a la información que se desea obtener de la función.

2.2 Otras técnicas

La resolución de problemas complejos, requiere de técnicas más sofisticadas, que permitan transformar el problema en otro más simple. Un ejemplo

es el caso de la transformada de Fourier para la cual se tiene una versión cuántica, empleada en el algoritmo de factorización numérica de Shor. La complejidad de la transformada de Fourier rápida clásica, se sitúa en $O(n2^n)$ para números de n bits. Sin embargo, para la versión cuántica, la complejidad se reduce a $O(n^2)$.

Por otra parte, otros algoritmos como el de búsqueda de Grover, emplean una técnica de aproximación al resultado. De forma que, iterativamente se acercan a la solución. La probabilidad de encontrar la solución correcta al final del algoritmo se reduce a medida que se realizan más iteraciones.

El algoritmo de Simon, que se analizará de forma minuciosa posteriormente, emplea la técnica de iteración, para obtener en cada ejecución nueva información que permite acercarse a la solución.

Capítulo 3

Algoritmo de Simon

3.1 Algoritmo de Simon

3.1.1 Descripción del problema

Daniel R. Simon encontró un problema que presentaba una complejidad exponencial al tratar de resolverse mediante métodos de computación tradicionales [12]. Sin embargo, al abordarlo mediante una solución cuántica, dicha complejidad disminuía de orden, a una lineal.

El problema consiste en encontrar una cadena binaria desconocida, denominada s, que contiene n bits, relacionada con una función f. Sea V el conjunto de cadenas de n bits, $V = \{0,1\}^n$, entonces la función f asigna a cada cadena de V otra cadena del mismo conjunto, $f : V \to V$.

Además la función cumple la restricción

$$f(x_1) = f(x_2) \iff x_1 = x_2 \oplus s \tag{3.1}$$

Por una parte, si el resultado de dos salidas es idéntico, es decir $f(x_1) = f(x_2)$ entonces entradas deben cumplir que $x_1 = x_2 \oplus s$:

$$f(x_1) = f(x_2) \implies x_1 = x_2 \oplus s$$

Y por otra parte, si dos pares de entradas cumplen $x_1 = x_2 \oplus s$ entonces sus salidas serán idénticas:

$$x_1 = x_2 \oplus s \implies f(x_1) = f(x_2)$$

Con un ejemplo de 2 bits, y una cadena $s = 01$, se define f:

x	$f(x)$	$x \oplus s$	$f(x \oplus s)$
00	00	01	00
01	00	00	00
10	01	11	01
11	01	10	01

Se observa que $f(x)$ es idéntico a $f(x \oplus s)$, de modo que esta función cumple los requisitos para el problema.

Solución clásica

Una solución clásica consistiría en evaluar f con entradas diferentes hasta encontrar una entrada x_1 que produzca una salida que ya se hubiera observado con otra entrada x_2. Entonces, debido a la restricción 3.1 se tiene que $x_2 = x_1 \oplus s$, y por lo tanto se puede calcular s como $s = x_1 \oplus x_2$.

Debido a que dos entradas producen la misma salida, y que todas las posibles entradas forman pares de la forma $x_2 = x_1 \oplus s$, la función f tendrá la mitad de sus salidas iguales. Es decir, que sólo existen $2^{n/2}$ salidas diferentes, por lo que al evaluar una más, $2^{n/2}+1$ se asegura encontrar dos entradas que produzcan la misma salida. De esta forma la complejidad se sitúa en $\mathcal{O}(2^{n/2})$.

Esta solución tiene el inconveniente de ser muy costosa computacionalmente, dado que para encontrar una coincidencia en la salida es necesario evaluar la función un número exponencial de veces con respecto al número de bits n.

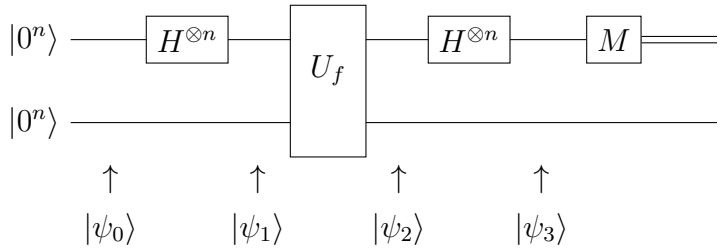
Empleando la función del ejemplo, bastaría con evaluarla k veces con

$$k = 2^{2/2} + 1 = 3$$

Entonces si $y_i = f(x_i)$, se obtiene $\boldsymbol{x} = \begin{pmatrix} 00 & 01 & 10 \end{pmatrix}$ y $\boldsymbol{y} = \begin{pmatrix} 00 & 00 & 01 \end{pmatrix}$. Como se puede observar $y_1 = y_2$, de modo que $x_1 = x_2 \oplus s$, y por lo tanto $s = x_1 \oplus x_2 = 00 \oplus 01 = 01$

3.1.2 Solución cuántica

El algoritmo de Simon consiste en un circuito cuántico con dos líneas, cada una de n qubits.



Primero el sistema se inicia con ambas líneas a $|0^n\rangle$. Luego se aplica un operador de Hadamard sobre la primera, que creará una superposición de estados. Posteriormente, el operador U_f computa los resultados de la función f de forma simultánea. Para terminar, se revierte el operador de Hadamard sobre la primera línea, que finalmente se mide.

Los estados asociados a cada paso de la computación, han sido etiquetados como $|\psi_i\rangle$, de forma que en todo momento se pueda determinar el lugar al que corresponden en el circuito.

3.1.3 Funcionamiento

Para comprender el funcionamiento del algoritmo, se empleará un ejemplo sencillo con sólo dos bits, $n = 2$ y de período $s = 01$:

x	$f(x)$
00	00
01	00
10	01
11	01

Sea $V = \{0,1\}^n$ entonces $f : V \to V$ y $f(x) = f(x \oplus s)$. De esta forma, $f(00) = f(00 \oplus s) = f(01)$ y también $f(10) = f(10 \oplus s) = f(11)$.

El sistema ha de iniciarse con ambas líneas a $|0^n\rangle$, de modo que:

$$|\psi_0\rangle = |0^n\rangle \otimes |0^n\rangle$$

A continuación, el operador de Hadamard es aplicado sobre la línea superior.

$$\begin{aligned} |\psi_1\rangle &= (H^{\otimes n} \otimes I^{\otimes n}) |\phi_0\rangle \\ &= (H^{\otimes n} \otimes I^{\otimes n})(|0^n\rangle \otimes |0^n\rangle) \\ &= (H^{\otimes n} |0^n\rangle) \otimes (I^{\otimes n} |0^n\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x \in V} |x, 0^n\rangle \end{aligned} \tag{3.2}$$

Produciendo el estado entrelazado

$$|\psi_1\rangle = \frac{1}{\sqrt{2^2}}(|00,00\rangle + |01,00\rangle + |10,00\rangle + |11,00\rangle) \tag{3.3}$$

Posteriormente, la puerta U_f actúa sobre $|\psi_1\rangle$ transformándolo en

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in V} |x, f(x) \oplus 00\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in V} |x, f(x)\rangle \tag{3.4}$$

En este estado, la función f está siendo evaluada simultáneamente en todo V, y este efecto será clave para la reducción de complejidad. Con la función f previamente descrita, dicho estado es

$$|\psi_2\rangle = \frac{1}{\sqrt{2^2}}(|00,00\rangle + |01,00\rangle + |10,01\rangle + |11,01\rangle) \tag{3.5}$$

Finalmente, el operador de Hadamard es nuevamente aplicado sobre la línea superior, de este modo se obtiene

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{x\in V}\sum_{z\in V}(-1)^{\langle z|x\rangle}\,|z, f(x)\rangle \tag{3.6}$$

Por consiguiente, el coeficiente de cada ket será

$$c_k = \frac{1}{2^n}(-1)^{\langle z|x\rangle} \tag{3.7}$$

Pero dado que $f(x) = f(x \oplus s)$, los kets $|z, f(x)\rangle$ y $|z, f(x \oplus s)\rangle$ serán el mismo, por lo que el coeficiente para este ket será

$$c_k = \frac{1}{2^n}\left((-1)^{\langle z|x\rangle} + (-1)^{\langle z|x\oplus s\rangle}\right) \tag{3.8}$$

El producto interno $\langle z|x \oplus s\rangle$ puede descomponerse en $\langle z|x\rangle \oplus \langle z|s\rangle$, y substituyendo

$$\begin{aligned} c_k &= \frac{1}{2^n}\left((-1)^{\langle z|x\rangle} + (-1)^{\langle z|x\rangle\oplus\langle z|s\rangle}\right) \\ &= \frac{1}{2^n}\left((-1)^{\langle z|x\rangle} + (-1)^{\langle z|x\rangle}(-1)^{\langle z|s\rangle}\right) \end{aligned} \tag{3.9}$$

Cuando $\langle z|s\rangle = 1$, el coeficiente c_k será

$$c_k = \frac{1}{2^n}\left((-1)^{\langle z|x\rangle} - (-1)^{\langle z|x\rangle}\right) = \frac{1}{2^n}(0) = 0 \tag{3.10}$$

Y cuando $\langle z|s\rangle = 0$, será

$$c_k = \frac{1}{2^n}\left((-1)^{\langle z|x\rangle} + (-1)^{\langle z|x\rangle}\right) = \frac{1}{2^n}(\pm 2) = \pm 2^{1-n} \tag{3.11}$$

De modo que $c_k \neq 0$ si en el ket $|z, f(x)\rangle$ se cumple que $\langle z|s\rangle = 0$. Así que sólo los estados de esta forma tendrán coeficientes no nulos. En el caso del ejemplo será

$$\begin{aligned} |\psi_3\rangle = 2^{-n}\big(&+ |00,00\rangle + |01,00\rangle + |10,00\rangle + |11,00\rangle \\ &+ |00,00\rangle - |01,00\rangle + |10,00\rangle - |11,00\rangle \\ &+ |00,01\rangle + |01,01\rangle - |10,01\rangle - |11,01\rangle \\ &+ |00,01\rangle - |01,01\rangle - |10,01\rangle + |11,01\rangle \big) \end{aligned} \tag{3.12}$$

Finalmente, tras anular los términos en los que $\langle z|01\rangle = 1$, es decir $z = 01$ y $z = 11$, queda

$$|\psi_3\rangle = 2^{1-n}\left(|00,00\rangle + |10,00\rangle + |00,01\rangle - |10,01\rangle\right) \tag{3.13}$$

En este estado, si se realiza una medición de z, el resultado siempre cumplirá la restricción $\langle z|s\rangle = 0$. Una vez que se obtengan $n-1$ vectores linealmente independientes, será posible construir un sistema de ecuaciones y calcular s.

En este ejemplo, después de medir la primera línea, el conjunto de vectores posibles resulta $z \in \{00, 10\}$. En todo caso, se tiene que $\langle z|s\rangle = 0$ de modo que $\langle 00|s\rangle = 0$ y $\langle 10|s\rangle = 0$, produciendo las ecuaciones:

$$\begin{aligned} 0s_0 \oplus 0s_1 &= 0 \\ 1s_0 \oplus 0s_1 &= 0 \end{aligned} \tag{3.14}$$

Cuyas soluciones son $s \in \{00, 01\}$, pero dado que $s \neq 00$, se obtiene $s = 01$.

3.1.4 Análisis de complejidad

Para una función de n bits, la salida del algoritmo tras medir, corresponderá a un vector z de n bits, con $W = \{z/\langle z|s\rangle = 0\}$ y $z \in W$. Los diferentes vectores de W, son exactamente 2^{n-1}

En cuanto se obtienen $n-1$ vectores independientes, el proceso concluye, ya que es posible averiguar s. Sin embargo, todos los vectores de W pueden salir con igual probabilidad.

Si se disponen de i vectores independientes de los 2^{n-1} posibles, hay 2^i vectores que se pueden formar realizando combinaciones lineales de los i vectores independientes, y que no sirven. De modo que quedan $2^{n-1} - 2^i$ vectores independientes. La probabilidad de obtener un nuevo vector en la siguiente medición será:

$$p(\text{Nuevo vector independiente}) = \frac{2^{n-1} - 2^i}{2^{n-1}} = 1 - 2^{i-n+1}$$

Y por lo tanto, la de no obtenerlo:

$$p(\text{Nuevo vector dependiente}) = 2^{i-n+1}$$

Sea R una variable aleatoria discreta que mide el número de ejecuciones necesarias para completar los $n-1$ vectores independientes. El valor medio de ejecuciones $E[R]$, puede medir cual es la media de ejecuciones que son necesarias a la larga [9]. Para calcularlo:

$$E[R] = \sum_{x=1}^{\infty} x \cdot p(R = x)$$

Es necesario determinar cual es la probabilidad de terminar en x ejecuciones consecutivas del algoritmo, o pasos.

Probabilidad de terminar

La probabilidad de terminar en p pasos depende del número de bits n y del número de vectores linealmente independientes i ya obtenidos previamente.

Sea T_p^i la probabilidad de terminar en p pasos, cuando ya se han obtenido i vectores independientes de los $n-1$ necesarios. La probabilidad de que el algoritmo termine desde el comienzo en p pasos será T_p^0, puesto que al comienzo no se tiene ningún vector.

Tras un paso de la ejecución, será posible encontrar un nuevo vector independiente con una probabilidad I^i, partiendo de que ya se tenían i previamente. La probabilidad de no obtenerlo será entonces representada por $\overline{I^i}$, siendo $\overline{I^i} = 1 - I^i$.

De este modo, la probabilidad T_p^i se puede expresar como una recurrencia, dividida en dos casos:

Cuando al obtener un nuevo vector, éste no es linealmente independiente, con probabilidad $\overline{I^i}$, de forma que será necesario resolver el problema con un paso menos, y los mismos vectores i, representado dicha probabilidad como T_{p-1}^i.

Y cuando se obtiene un vector linealmente independiente, con probabilidad I^i, que entonces existirá un nuevo vector, pero con un paso menos, con la probabilidad T_{p-1}^{i+1} de terminar.

En forma general, se tiene:

$$T_p^i = \overline{I^i}\, T_{p-1}^i + I^i\, T_{p-1}^{i+1} \tag{3.15}$$

Adicionalmente, cuando el número de pasos p restantes es 0, y se han conseguido exactamente $n-1$ vectores, la probabilidad de acabar es total:

$$T_0^{n-1} = 1$$

En cualquier otro caso, cuando $p = 0$ y $i \neq n-1$, será nula, debido a que no se han conseguido los vectores necesarios.

$$T_0^i = 0$$

Además si en algún momento se obtienen los $n-1$ vectores linealmente independientes, y el número de pasos $p \neq 0$, se ha resuelto el problema antes de lo previsto, y por lo tanto no es válido:

$$T_p^{n-1} = 0$$

Finalmente, las probabilidades de obtener y no obtener un vector linealmente independiente, partiendo de i previamente, en un problema de n bits son, respectivamente:

$$\begin{aligned} I^i &= 1 - 2^{i-n+1} \\ \overline{I^i} &= 2^{i-n+1} \end{aligned} \tag{3.16}$$

Al poder determinar la probabilidad con la que el algoritmo resuelve el problema en p pasos, es posible determinar el valor estimado de pasos a la larga.

Valor esperado de ejecuciones

El valor esperado de ejecuciones mide el número promedio de veces que es necesario repetir el algoritmo cuántico, a la larga, para resolver el problema. El número de ejecuciones R es una variable aleatoria discreta, y el valor esperado $E[R]$ puede calcularse partiendo de las probabilidades de cada valor de R. Si $p(R = r) = T_r^0$, entonces se define el valor esperado de R como

$$E[R] = \sum_{p=1}^{\infty} p \cdot T_p^0 \tag{3.17}$$

Resolución de la recurrencia T_p^0

Para resolver la recurrencia (3.15) se empleará un método de prueba y simplificación hasta dar con un patrón [7]. Se comprobarán los primeros casos para $3 \leq n \leq 5$, ya que crecen muy rápidamente. Caso para $n = 3$

$$\begin{aligned} T_0^0 &= T_1^0 = 0 \\ T_2^0 &= I^0 I^1 \\ T_3^0 &= I^0 I^1 (\overline{I^0} + \overline{I^1}) \\ T_4^0 &= I^0 I^1 (\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}) \\ T_5^0 &= I^0 I^1 (\overline{I^0}\,\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^1}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}\,\overline{I^1}) \end{aligned} \tag{3.18}$$

Para $n = 4$:

$$\begin{aligned} T_0^0 &= T_1^0 = T_2^0 = 0 \\ T_3^0 &= I^0 I^1 I^2 \\ T_4^0 &= I^0 I^1 I^2 (\overline{I^0} + \overline{I^1} + \overline{I^2}) \\ T_5^0 &= I^0 I^1 I^2 (\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^2} + \overline{I^1}\,\overline{I^1} + \overline{I^1}\,\overline{I^2} + \overline{I^2}\,\overline{I^2}) \\ T_6^0 &= I^0 I^1 I^2 (\overline{I^0}\,\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^0}\,\overline{I^2} + \overline{I^0}\,\overline{I^1}\,\overline{I^1} + \overline{I^0}\,\overline{I^1}\,\overline{I^2} + \\ &\quad \overline{I^0}\,\overline{I^2}\,\overline{I^2} + \overline{I^1}\,\overline{I^1}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}\,\overline{I^2} + \overline{I^1}\,\overline{I^2}\,\overline{I^2} + \overline{I^2}\,\overline{I^2}\,\overline{I^2}) \end{aligned} \tag{3.19}$$

Para $n = 5$:

$$
\begin{aligned}
&T_0^0 = T_1^0 = T_2^0 = T_3^0 = 0 \\
&T_4^0 = I^0 I^1 I^2 I^3 \\
&T_5^0 = I^0 I^1 I^2 I^3 (\overline{I^0} + \overline{I^1} + \overline{I^2} + \overline{I^3}) \\
&T_6^0 = I^0 I^1 I^2 I^3 (\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^2} + \overline{I^0}\,\overline{I^3} + \overline{I^1}\,\overline{I^1} + \overline{I^1}\,\overline{I^2} + \\
&\quad \overline{I^1}\,\overline{I^3} + \overline{I^2}\,\overline{I^2} + \overline{I^2}\,\overline{I^3} + \overline{I^3}\,\overline{I^3}) \\
&T_7^0 = I^0 I^1 I^2 I^3 (\overline{I^0}\,\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^0}\,\overline{I^2} + \overline{I^0}\,\overline{I^0}\,\overline{I^3} + \overline{I^0}\,\overline{I^1}\,\overline{I^1} + \\
&\quad \overline{I^0}\,\overline{I^1}\,\overline{I^2} + \overline{I^0}\,\overline{I^1}\,\overline{I^3} + \overline{I^0}\,\overline{I^2}\,\overline{I^2} + \overline{I^0}\,\overline{I^2}\,\overline{I^3} + \overline{I^0}\,\overline{I^3}\,\overline{I^3} + \overline{I^1}\,\overline{I^1}\,\overline{I^1} + \\
&\quad \overline{I^1}\,\overline{I^1}\,\overline{I^2} + \overline{I^1}\,\overline{I^1}\,\overline{I^3} + \overline{I^1}\,\overline{I^2}\,\overline{I^2} + \overline{I^1}\,\overline{I^2}\,\overline{I^3} + \overline{I^1}\,\overline{I^3}\,\overline{I^3} + \overline{I^2}\,\overline{I^2}\,\overline{I^2} + \\
&\quad \overline{I^2}\,\overline{I^2}\,\overline{I^3} + \overline{I^2}\,\overline{I^3}\,\overline{I^3} + \overline{I^3}\,\overline{I^3}\,\overline{I^3})
\end{aligned}
\tag{3.20}
$$

Se observa que en cada expresión T_p^0 se encuentra un producto de I^j. Dado que es necesario encontrar exactamente $n-1$ vectores linealmente independientes, este producto se corresponde con las probabilidades de encontrar los $n-1$ vectores uno detrás de otro. Primero partiendo de 0 vectores, luego de 1 y así sucesivamente hasta $n-2$. Dado que se conoce $n-1$ este producto se puede expresar como

$$J = \prod_{j=0}^{n-2} I^j$$

El resto de la ecuación se denotará como S_p, de forma que se obtiene

$$T_p^0 = J \cdot S_p$$

La expresión S_p es una suma de probabilidades, y representan las diferentes combinaciones de obtener los $n-1$ vectores. Por ejemplo, para $n = 3$, es necesario obtener 2 vectores linealmente independientes. Sea el número de pasos $p = 4$. Se empleará una cadena binaria de p bits para representar si en una ejecución se obtiene un nuevo vector independiente indicado con un 1, o con un 0 si no se obtiene.

Dado que son necesarios exactamente $n-1$ vectores, todas las ejecuciones contendrán exactamente dos unos. Además no puede haber ejecuciones que consigan haber conseguido 2 unos antes de terminar, dado que entonces habrían resuelto el problema en menos pasos de los necesarios. Las 3 posibles ejecuciones son:

- Conseguirlos al final de todo: 0011
- Conseguir uno en la segunda ejecución y otro al final: 0101

- Conseguir uno al principio y otro al final: 1001

La probabilidad de que no se obtenga un vector linealmente independiente, expresada como $\overline{I^i}$, tan sólo depende del número de vectores independientes de los que ya se dispone. De esta forma, las probabilidades de obtener el primer resultado 0011 serán:

$$\overline{I^0}\,\overline{I^0}\,I^0 I^1$$

Para el segundo, 0101:

$$\overline{I^0}\,I^0\overline{I^1}\,I^1$$

Y para el último, 1001:

$$I^0\overline{I^1}\,\overline{I^1}\,I^1$$

Estas probabilidades son las tres posibles opciones que resuelven el problema en exactamente 4 pasos. Al sumarlas se obtiene la probabilidad de que el problema sea resuelto empleando alguna posible ejecución correcta. Dado que cada ejecución es independiente se tiene:

$$\overline{I^0}\,\overline{I^0}\,I^0 I^1 + \overline{I^0}\,I^0\overline{I^1}\,I^1 + I^0\overline{I^1}\,\overline{I^1}\,I^1 = I^0 I^1(\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}) \tag{3.21}$$

Que es exactamente la misma expresión obtenida por la recurrencia (3.18) con T_4^0 para $n = 3$:

$$T_4^0 = \underbrace{I^0 I^1}_{J}(\underbrace{\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}}_{S_p})$$

El siguiente paso, será determinar como se forman las secuencias S_p. Dado que son necesarios $n-1$ vectores linealmente independientes, el resto de ejecuciones $p-(n-1)$ proporcionan vectores dependientes o repetidos. Es decir que se obtendrán secuencias de $\overline{I^i}$ con $p-(n-1)$ elementos.

Sea K una sucesión de m productos $K = \overline{I^{i_1}}\,\overline{I^{i_2}}\,\ldots\overline{I^{i_m}}$, partiendo de la definición de $\overline{I^i}$, se tiene:

$$\begin{aligned} K &= 2^{i_1-n+1}\,2^{i_2-n+1}\ldots 2^{i_m-n+1} \\ &= 2^{(i_1-n+1)+(i_2-n+1)+\ldots+(i_m-n+1)} \\ &= 2^{m(-n+1)+i_1+i_2+\ldots+i_m} \\ &= 2^{m(-n+1)}\,2^{i_1+i_2+\ldots+i_m} \end{aligned} \tag{3.22}$$

Examinando las secuencias de productos para $n = 3$, con $p \geq n$:

$$\begin{aligned} S_3 &= \overline{I^0} + \overline{I^1} \\ S_4 &= \overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^1} + \overline{I^1}\,\overline{I^1} \\ S_5 &= \overline{I^0}\,\overline{I^0}\,\overline{I^0} + \overline{I^0}\,\overline{I^0}\,\overline{I^1} + \overline{I^0}\,\overline{I^1}\,\overline{I^1} + \overline{I^1}\,\overline{I^1}\,\overline{I^1} \end{aligned} \tag{3.23}$$

Reemplazando las secuencias con la ecuación (3.22):

$$\begin{aligned} S_3 &= 2^{1(-n+1)}(2^0 + 2^1) = 1/4^1 \cdot 3 \\ S_4 &= 2^{2(-n+1)}(2^{0+0} + 2^{0+1} + 2^{1+1}) = 1/4^2 \cdot 7 \\ S_5 &= 2^{3(-n+1)}(2^{0+0+0} + 2^{0+0+1} + 2^{0+1+1} + 2^{1+1+1}) = 1/4^3 \cdot 15 \end{aligned} \tag{3.24}$$

Para $n = 4$:

$$\begin{aligned} S_4 &= 1/8^1 \cdot 7 \\ S_5 &= 1/8^2 \cdot 35 \\ S_6 &= 1/8^3 \cdot 155 \end{aligned} \tag{3.25}$$

Para $n = 5$:

$$\begin{aligned} S_5 &= 1/16^1 \cdot 15 \\ S_6 &= 1/16^2 \cdot 155 \\ S_7 &= \underbrace{1/16^3}_{P_p} \cdot \underbrace{1395}_{Q_p} \end{aligned} \tag{3.26}$$

Por una parte, S_p contiene la potencia $P_p = 2^{(p-n+1)(-n+1)}$, y por otra, un número Q_p, de modo que

$$S_p = P_p \cdot Q_p$$

Al examinar las secuencias Q_p, se obtienen los valores:

n	Q_n	Q_{n+1}	Q_{n+2}	Q_{n+3}	Q_{n+4}
3	3	7	15	31	63
4	7	35	155	651	2667
5	31	651	11811	2007878	3309747

Estas secuencias se corresponden con los coeficientes binomiales de Gauss con el parámetro $q = 2$, definidos para $r \leq m$, como:

$$\binom{m}{r}_{q=2} = \frac{\prod_{i=m}^{m-r+1}(1-2^i)}{\prod_{i=1}^{r}(1-2^i)} = \prod_{i=0}^{r-1}\frac{(2^m-2^i)}{(2^r-2^i)}$$

De esta forma, el valor de Q_p se determina como:

$$Q_p = \binom{p-1}{n-2}_{q=2}$$

Reunificando todos los componentes que forman la recurrencia, se obtiene

$$\begin{aligned} T_p^0 &= J \cdot S_p = J \cdot P_p \cdot Q_p = \\ &= \prod_{j=0}^{n-2} I^j \cdot P_p \cdot Q_p = \\ &= \prod_{j=0}^{n-2} I^j \cdot 2^{(p-n+1)(-n+1)} \cdot Q_p = \\ &= \prod_{j=0}^{n-2} I^j \cdot 2^{(p-n+1)(-n+1)} \cdot \binom{p-1}{n-2}_{q=2} \end{aligned} \tag{3.27}$$

Cálculo del valor medio

Una vez obtenida la ecuación (3.27), se puede reemplazar la probabilidad T_p^0 en $E[R]$, quedando:

$$\begin{aligned} E[R] &= \sum_{p=1}^{\infty} p T_p^0 = \\ &= \sum_{p=1}^{\infty} p \cdot \prod_{j=0}^{n-2} I^j \cdot 2^{(p-n+1)(-n+1)} \cdot \binom{p-1}{n-2}_{q=2} = \\ &= \prod_{j=0}^{n-2} I^j \cdot \sum_{p=1}^{\infty} p \cdot 2^{(p-n+1)(-n+1)} \cdot \binom{p-1}{n-2}_{q=2} \end{aligned} \tag{3.28}$$

De este modo, se pueden computar los valores teóricos esperados de ejecuciones que cabría esperar a la larga.

n	2	3	4	5	6	7	8	9	10
$E[R]$	2.000	3.333	4.476	5.542	6.575	7.591	8.598	9.602	10.60
$\frac{E[R]}{n}$	1.000	1.111	1.119	1.109	1.096	1.084	1.075	1.067	1.060

Tabla 3.1: Valor esperado de ejecuciones del algoritmo de Simon

Se observa como $E[R]$ se va aproximando cada vez más a n. De esta forma, el algoritmo se comporta con una complejidad a la larga de $O(n)$.

Capítulo 4

Simulador cuántico

Un simulador de circuitos cuántico, es un programa ejecutado en una máquina clásica que permite efectuar una simulación de un circuito, de forma que los resultados obtenidos sean idénticos a los que se obtendrían en una máquina cuántica que ejecute el mismo circuito.

La principal desventaja en un simulador cuántico, es que la simulación es un proceso muy costoso, que crece en complejidad de forma exponencial con el tamaño de la entrada. Sin embargo, permite analizar el comportamiento de un algoritmo, y comprobar que el comportamiento simulado es el predicho de forma teórica.

El proceso de simulación se basa en dos partes. Primero se determinan todas las puertas del circuito, y se aplican al estado inicial de forma secuencial. De esta forma se obtiene el estado final. Esta es la parte de simulación cuántica.

Por otra parte, es necesario un proceso de medida y cómputo empleando bits clásicos. De modo que, partiendo del estado final, se determina el proceso de medición calculando las probabilidades asociadas a cada estado clásico. Con la distribución de probabilidad fijada, se efectúa la generación de una medida aleatoria, y a continuación los cálculos necesarios con las mediciones resultantes.

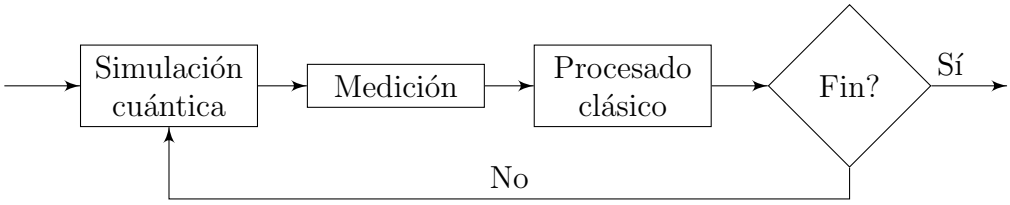
4.1 Diseño del simulador

El simulador consta de tres partes bien diferenciadas. Cada una se encarga de realizar una tarea en concreto.

- Simulación del circuito cuántico.
- Medición de estados.

- Procesado clásico.

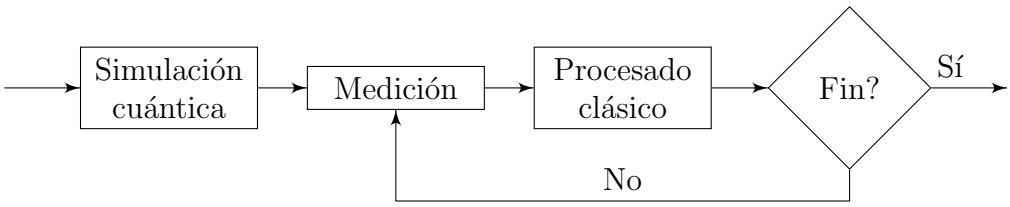
Primero, el circuito cuántico es simulado, produciendo un estado final cuántico. A continuación, un proceso de medición colapsa los registros cuánticos en bits clásicos. Finalmente, un última etapa de cálculo clásico, toma dicha información para ser procesada. La simulación se repite hasta que el problema se haya completado.



4.1.1 Diseño optimizado

Realizar la simulación de un circuito cuántico es una operación muy costosa. Además, partiendo de la misma entrada, el estado cuántico final es idéntico. Debido a que al realizar la medición en el simulador, no se destruye el estado cuántico, es posible almacenarlo.

De este modo, tan sólo será necesario realizar la ejecución del circuito cuántico para obtener el estado final; y posteriormente, emplear dicho estado en las mediciones sucesivas. El esquema optimizado de la simulación, se puede apreciar en el siguiente diagrama.



4.2 Estructuras de datos

A medida que crece el número de qubits n, crece el espacio necesario para almacenar los operadores y los estados. Un estado de n qubits, emplea un vector de 2^n amplitudes, y cada amplitud está representada por un número complejo. Un operador que actúe sobre dicho estado, requiere de $2^n \cdot 2^n = 2^{2n}$

elementos. Sea $|\psi\rangle$ un estado de n qubits, H un operador, y $m = 2^n - 1$:

$$|\psi\rangle = \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_m \end{pmatrix}, H = \begin{pmatrix} u_{00} & u_{01} & \dots & u_{0m} \\ u_{10} & u_{11} & \dots & u_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m0} & u_{m1} & \dots & u_{mm} \end{pmatrix}$$

De modo que aplicar el operador H sobre el estado $|\psi\rangle$, equivale a una multiplicación de una matriz por un vector:

$$H|\psi\rangle = \begin{pmatrix} u_{00} & u_{01} & \dots & u_{0m} \\ u_{10} & u_{11} & \dots & u_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m0} & u_{m1} & \dots & u_{mm} \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_m \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{pmatrix} = |\psi'\rangle$$

Sin embargo, generalmente $|\psi\rangle$ contiene muchos elementos que son cero, $e_i = 0$ y no necesitan ser almacenados. Las matrices huecas aprovechan esta propiedad para reducir el espacio de almacenamiento.

4.2.1 Matrices huecas

Si la matriz $\mathbf{A}$ contiene *muchos* elementos nulos, se considera una matriz *hueca*. Aprovechar esta propiedad permite: por una parte reducir el número de elementos que es necesario almacenar, y por otra, reducir el número de sumas y multiplicaciones al operar con la matriz.

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Sea N_z el número de elementos no nulos de $\mathbf{A}$. Entonces la *densidad* de una matriz se define como dicho número entre el total, N_z/mn.

Almacenamiento por coordenadas COO

La matriz $\mathbf{A} \in \mathbb{M}^{m \times n}$ puede almacenarse en tres vectores AA, JR y JC; de forma que AA contiene todos los elementos no nulos junto con JR y JC que almacenan los índices de la fila y columna respectivamente. Este esquema se conoce como almacenamiento por coordenadas (COO).

Sea S_e y S_i el tamaño de almacenamiento de un elemento de la matriz y de un índice respectivamente, entonces el espacio necesario es

$$S_{COO} = S_e N_z + 2 S_i N_z = O(N_z)$$

En comparación con la representación íntegra de la matriz, se conseguiría una reducción del espacio si

$$N_z/mn < \frac{S_e}{S_e + 2S_i}$$

Ejemplo 4.2.1. La matriz **A** de 5×5 elementos, contiene 10 no nulos, de un total de 25, que se almacenan en el vector AA. A cada uno le corresponde un índice en JR que indica la fila, y otro en JC con la columna. El elemento 8 se encuenta en AA[7]. Entonce JR[7] indica la tercera fila en la matriz, y JC[7] la tercera columna.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 9 & 0 & \mathbf{8} & 0 & 0 \\ 0 & 0 & 6 & 5 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix} \quad \begin{aligned} AA &= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 9 & \mathbf{8} & 6 & 5 & 4 \end{bmatrix} \\ JR &= \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 3 & \mathbf{3} & 4 & 4 & 5 \end{bmatrix} \\ JC &= \begin{bmatrix} 1 & 4 & 1 & 2 & 4 & 1 & \mathbf{3} & 3 & 4 & 5 \end{bmatrix} \end{aligned}$$

Almacenamiento por filas comprimidas (CSR)

El almacenamiento por coordenadas contiene información redundante. Los elementos de la misma fila repiten el mismo índice en JR. Para reducir el espacio, se puede emplear sólo el índice en AA del primer elemento de cada fila.

De este modo, AA almacena los valores no nulos, ordenados por filas. El vector JA mantendrá las posiciones de las columnas para cada elemento de AA, y finalmente, IA, el índice en AA del primer elemento de la fila. Adicionalmente IA termina con un elemento extra, que indica el final de la última fila, con valor $N_z + 1$.

El tamaño necesario para almacenar una matriz de m filas en CSR es

$$S_{CSR} = S_e N_z + S_i(N_z + m + 1)$$

El espacio necesario sería menor que en COO, si $m + 1 < N_z$.

Ejemplo 4.2.2. En la misma matriz:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 9 & 0 & \mathbf{8} & 0 & 0 \\ 0 & 0 & 6 & 5 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix} \quad \begin{aligned} AA &= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 9 & \mathbf{8} & 6 & 5 & 4 \end{bmatrix} \\ JA &= \begin{bmatrix} 1 & 4 & 1 & 2 & 4 & 1 & \mathbf{3} & 3 & 4 & 5 \end{bmatrix} \\ IA &= \begin{bmatrix} 1 & 3 & \mathbf{6} & 8 & 10 & 11 \end{bmatrix} \end{aligned}$$

Para acceder al elemento (3, 3), primero se calcula la posición en AA del elemento que comienza la tercera fila, este índice es IA[3] = 6. La tercera fila se sitúa en AA desde IA[3] = 6 hasta IA[3 + 1] − 1 = 7. Para determinar la columna se recorren los índices de JA desde 6 hasta 7, buscando la columna 3, que se encuentra en JA[7] = 3. De modo que el elemento buscado se encuentra en AA[7] = 8.

4.2.2 Operadores secuenciales

Un circuito cuántico, que contiene varios operadores que se aplican de forma secuencial, es equivalente a un único operador que realiza todo el proceso. Este operador C sería el resultado de multiplicar las matrices que conforman las operaciones en cada etapa, en el mismo orden. En el caso del algoritmo de Simon:

$$C = (H^{\otimes n} \otimes I^{\otimes n}) \cdot U_f \cdot (H^{\otimes n} \otimes I^{\otimes n})$$

Calcular el producto de las matrices es computacionalmente muy costoso, incluso prohibitivo para un n grande. Adicionamente, los operadores que sólo actúan sobre una línea, como $(H^{\otimes n} \otimes I^{\otimes n})$, y en general cualquier operador del tipo $(A \otimes I^{\otimes n})$ permiten una optimización en tiempo y espacio.

Semi-operadores

Un *semi-operador* es un operador A que sólo actúa en la mitad superior de un circuito. De modo que su correspondiente operador se calcula como $D = (A \otimes I^{\otimes n})$. Sea $B = I^{\otimes n}$, y tanto A como B sean matrices de $2^n \times 2^n$, los elementos de $D = A \otimes B$ se determinan como

$$\begin{aligned} B[i,j] &= A[i_H, j_H] \cdot B[i_L, j_L] \\ i_H &= \lfloor i/2^n \rfloor \\ j_H &= \lfloor j/2^n \rfloor \\ i_L &= i \mod 2^n \\ j_L &= j \mod 2^n \end{aligned} \tag{4.1}$$

Debido a que $B[i,j]$ es la identidad, cuando $i \neq j, B[i,j] = 0$. Los cálculos se simplifican.

El estado $|\psi\rangle$ de $2n$ qubits, si $m = 2^{2n} - 1$, se puede descomponer en:

$$|\psi\rangle = \sum_{i=0}^{m} a_i \, |i\rangle$$

Sea J el conjunto de posiciones i para las cuales $a_i \neq 0$, $J = \{i \mid a_i \neq 0\}$, entonces

$$|\psi\rangle = \sum_{i=0}^{m} a_i |i\rangle = \sum_{i \in J} a_i |i\rangle$$

Además, como $|i\rangle$ es un estado básico, se puede descomponer en $|i\rangle = |i_H\rangle \otimes |i_L\rangle$, con $i_H = \lfloor i/2^n \rfloor$ y $i_L = (i \mod 2^n)$, obteniendo

$$|\psi\rangle = \sum_{i \in J} a_i |i_H\rangle \otimes |i_L\rangle$$

De modo que aplicar el operador D sobre $|\psi\rangle$ es

$$\begin{aligned} D|\psi\rangle &= (A \otimes B)|\psi\rangle = (A \otimes B) \sum_{i \in J} a_i |i_H\rangle \otimes |i_L\rangle \\ &= \sum_{i \in J} a_i (A \otimes B)(|i_H\rangle \otimes |i_L\rangle) \\ &= \sum_{i \in J} a_i (A|i_H\rangle) \otimes (B|i_L\rangle) \end{aligned} \tag{4.2}$$

Dado que B es el operador identidad, $B|i_L\rangle = |i_L\rangle$.

$$D|\psi\rangle = \sum_{i \in J} a_i (A|i_H\rangle) \otimes (|i_L\rangle)$$

Pero como $|i_H\rangle$, es un estado básico, tendrá un 1 en la posición i_H (comenzando en 0) y el resto de amplitudes serán 0. Quedando el producto $A|i_H\rangle$:

$$A \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} A[0, i_H] \\ \vdots \\ A[i_H, i_H] \\ \vdots \\ A[m, i_H] \end{pmatrix}$$

Es decir, el producto será un vector con la columna i_H de A. Representando dicho vector como $A[:, i_H]$,

$$D|\psi\rangle = \sum_{i \in J} a_i (A[:, i_H] \otimes |i_L\rangle)$$

Esta operación, no necesita calcular el producto tensorial $A \otimes B$, y almacenarlo en la memoria, empleando una matrix de $2^{2n} \times 2^{2n}$. Sólo requiere almacenar un estado cuántico de 2^{2n} elementos, y iterativamente calcular el

sumatorio. Lo cual reduce drásticamente el espacio requerido. El algoritmo se detalla a continuación.

Algoritmo 4.1: Aplicar semi-operador $A \otimes I^{\otimes n}$ al estado $|\psi\rangle$.

Entrada *Operador A y estado* $|\psi\rangle$;
Salida *Estado* $|\psi'\rangle = (A \otimes I^{\otimes n})\,|\psi\rangle$;
$|\psi'\rangle \longleftarrow \begin{pmatrix}0 & 0 & \dots & 0\end{pmatrix}^T$;
for $i \in J$ **do**
  $i_H \longleftarrow \lfloor i/2^n \rfloor$;
  $i_L \longleftarrow i \mod 2^n$;
  $|\psi'\rangle \longleftarrow |\psi'\rangle + a_i(A[:, i_H] \otimes |i_L\rangle)$;
end

4.2.3 Operadores y estados

El circuito de Simon requiere dos líneas de n qubits, por lo tanto, un estado requiere $N = 2n$ qubits, un total de 2^{2n} amplitudes. El entorno de simulación cuenta con una memoria limitada de 512 Mb $= 2^{29}$ bytes.

Sea $S_e = 16 = 2^4$ bytes, el tamaño de un número complejo. Un estado $|\psi\rangle$, empleará $2^{2n} \cdot S_e = 2^{2n+4}$ bytes. Limitando el espacio de la simulación: $2^{2n+4} \leq 2^{29}$ entonces $2n + 4 \leq 29 \implies n \leq 25/2 = 12{,}5$. Sólo sería posible almacenar estados de un tamaño de hasta $n = 12$. Al emplear las matrices huecas, el tamaño se reduce

n	$\log_2 S(\|\psi_1\rangle)$	$\log_2 S(\|\psi_3\rangle)$	$2n+4$
2	6.64	6.64	8
3	8.34	8.60	10
4	10.17	10.59	12
5	12.09	12.59	14
6	14.04	14.59	16
7	16.02	16.59	18
8	18.01	18.58	20
9	20.01	20.58	22
10	22.00	22.58	24

Tabla 4.1: Tamaño de los estados en escala logarítmica.

Sin embargo, un operador requiere mucha más memoria. En el caso de una matriz densa de $2n \times 2n$ serán necesarios $2^{2n} \cdot 2^{2n} = 2^{4n}$ números complejos.

De esta forma, se limita el tamaño a:

$$2^{4n+4} \leq 2^{29} \implies n \leq 25/4 = 6{,}25$$

Un máximo de $n = 6$. Sin embargo, empleando la propiedad de que tanto lo operadores como los estado contienen un gran número de elementos nulos (son matrices huecas), es posible reducir en gran medida su tamaño, y ampliar n hasta 11.

n	$\log_2 S(H)$	$\log_2 S(U_f)$	$4n+4$
2	7.21	7.61	12
3	9.10	9.59	16
4	11.05	11.59	20
5	13.02	13.59	24
6	15.01	15.59	28
7	17.01	17.58	32
8	19.00	19.58	36
9	21.00	21.58	40
10	23.00	23.58	44

Tabla 4.2: Tamaño de los operadores en escala logarítmica.

n	2	3	4	5	6	7	8	9	10
$2n$	4	6	8	10	12	14	16	18	20
$\log_2 S_T$	9.09	10.99	12.95	14.93	16.92	18.91	20.91	22.91	24.91

Tabla 4.3: Tamaño total de los operadores y los estados.

4.3 Implementación del simulador

La simulación se realiza con el lenguaje de programación `python`, empleando paquetes externos que aportarán las partes comunes del simulador. De esta forma se reutiliza el software ya existente.

Para los cálculos numéricos se emplea `numpy` y `scipy`, tales como multiplicación y suma de matrices, producto tensorial y funciones matemáticas vectorizadas. Para el almacenamiento de operadores y estados cuánticos, se emplea el paquete `qutip`. La clase `qutip.Qobj` permite emplear de forma implícita las matrices huecas de `scipy.sparse` con el formato de almacenamiento CSR o COO.

4.3.1 Circuito cuántico

Un circuito cuántico se describe como la aplicación sucesiva de operadores. Realizar la aplicación de un operador consiste en realizar la multiplicación de la matriz que lo representa por el estado cuántico.

El algoritmo de Simon, partiendo de los operadores $H^{\otimes n} \otimes I^{\otimes n}$, y U_f, se simplifica en el código

```
psi0 = basis(0, 2*n) # Estado inicial |ψ0⟩ = |0^n⟩ |0^n⟩
psi1 = H_I * psi0
psi2 = U_f * psi1
psi3 = H_I * psi2
```

4.3.2 Construcción de operadores

Para la simulación del circuito cuántico, serán necesarios varios operadores. Por una parte el semi-operador $H^{\otimes n} \otimes I^{\otimes n}$ que se empleará en dos ocasiones, y por otra el operador U_f.

Operador $H^{\otimes n} \otimes I^{\otimes n}$

El operador $H^{\otimes n} \otimes I^{\otimes n}$ es un semi-operador, debido a que no modifica la línea inferior. No es necesario construir íntegramente el operador completo, el algoritmo 4.1 permite calcular el resultado partiendo de $H^{\otimes n}$. Para obtener $H^{\otimes n}$ el paquete `qutip` incluye la operación `hadamard_transform` que computa las potencias tensoriales de H. La implementación del operador $H^{\otimes n} \otimes I^{\otimes n}$ empleando el algoritmo 4.1 se muestra a continuación.

```
def HI(self, state):
    bits = self.bits

    nz = state.data.nonzero()[0]
    ai = state.data[nz,0].T.toarray()[0]
    iH = nz >> bits
    iL = nz % 2**bits

    tmp = qutip.zero_ket(2**(2*bits))
    for i in range(len(nz)):
        a = ai[i]
        ketH = self.H * ket(iH[i], bits)
        ketL = ket(iL[i], bits)
        tmp += a * tensor(ketH, ketL)

    return tmp
```

En la línea 4, se obtienen los índices `nz` de los elementos no nulos del estado `state` al que se aplicará el operador. Para cada índice, en la línea 5, se almacena en `ai` la amplitud correspondiende. A continuación, se calcula i_H e i_L para cada índice, y se inicializa el estado resultante a un vector de ceros con `qutip.zero_ket`. Finalmente en las líneas de 9 a 14 se realiza el bucle que iterativamente calcula el estado resultante en `tmp`, tal y como se describe en el algoritmo 4.1.

Operador U_f

La operación U_f se define, a partir de una función f, como

$$U_f \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle$$

Esta operación, puede describirse como una matriz de $2^{2n} \times 2^{2n}$, aplicada sobre el estado $\left|x\right\rangle \left|y\right\rangle$, que es un vector de 2^{2n} componentes, siendo n el número de qubits de $\left|x\right\rangle$ y también de $\left|y\right\rangle$. Sea $m = 2^{2n} - 1$, se tiene

$$U_f \left|x\right\rangle \left|y\right\rangle = \begin{pmatrix} u_{00} & \dots & u_{0m} \\ \vdots & & \vdots \\ u_{m0} & \dots & u_{mm} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} b_0 \\ \vdots \\ b_m \end{pmatrix}$$

El estado $\left|x\right\rangle \left|y\right\rangle$ se puede escribir también como

$$\left|x\right\rangle \left|y\right\rangle = \sum_{k=0}^{m} a_k \left|k_H\right\rangle \left|k_L\right\rangle$$

Siendo $k_H = \lfloor k/2^n \rfloor$ y $k_L = k \mod 2^n$. De modo que al aplicar U_f,

$$U_f \left|x\right\rangle \left|y\right\rangle = \sum_{k=0}^{m} a_k \; U_f \left|k_H\right\rangle \left|k_L\right\rangle = \sum_{k=0}^{m} a_k \left|k_H\right\rangle \left|k_L \oplus f(k_H)\right\rangle$$

El estado $\left|k_H\right\rangle \left|k_L\right\rangle$ es un vector de 2^{2n} componentes, con un 1 en $k = k_H \cdot 2^n + k_L$ y con el resto 0; el estado $\left|k_H\right\rangle \left|k_L \oplus f(k_H)\right\rangle$ en $k' = k_H \cdot 2^n + k_L \oplus f(k_H)$. Se observa que la operación U_f toma la componente a_k en la posición k y la translada a la posición k'. Es decir, se trata de una matriz de permutación. Para realizar esta transformación, basta con colocar un 1 en la columna k y en la fila k'. De esta forma se define U_f como:

$$U_f[i,j] = \begin{cases} 1, & \text{si } i = k',\ j = k \\ 0, & \text{en otro caso} \end{cases}$$

Para $0 \leq i, j \leq m$ con $k = j$ y $k' = k_H \cdot 2^n + k_L \otimes f(k_H)$.
El algoritmo de construcción del operador U_f se detalla a continuación

```
1  def build_U(self, f):
2      n = self.bits
3      m = 2 ** (2 * n) - 1
4      k = np.arange(0, m + 1)
5      kH = k >> n
6      kL = k % 2 ** n
7      _kH = kH
8      _kL = f[kH] ^ kL
9      _k = _kH << n | _kL
10     data = [1]*n
11     U = coo.coo_matrix((data, (_k, k)), shape=[n, n])
12     return qutip.Qobj(U)
```

La operación de desplazamiento de bits $k_H = k \gg n$ de la línea 5, equivale a una división entera $k_H = \lfloor k/2^n \rfloor$.

El funcionamiento del algoritmo, consiste en calcular las posiciones k y k', para $0 \leq k \leq m$, como se observa en la línea 4. Luego se calcula k' aplicando la función f y la suma $\oplus$, en las líneas 8 y 9. Finalmente en la línea 11 se construye la matriz U_f indicando sólo las posiciones en las que se encuentran los unos, las filas k' y columnas k. El resto de la matriz contiene elementos nulos, que no es necesario almacenar, ahorrando un espacio considerable.

Capítulo 5

Análisis de la simulación

El proceso de la simulación se analiza exhaustivamente para determinar por una parte la *eficiencia* del simulador, midiendo el tiempo empleado y la memoria. Además también se observa el número de ejecuciones del algoritmo cuántico, para calcular su *complejidad*.

5.1 Análisis del tiempo de CPU

"En casi todo cómputo son posibles una gran variedad de configuraciones para la sucesión de un proceso, y varias consideraciones pueden influir en la selección de estas según el propósito de un motor de cálculos. Una objetivo esencial es escoger la configuración que tienda a minimizar el tiempo necesario para completar el cálculo."—Augusta Ada Lovelace. [1]

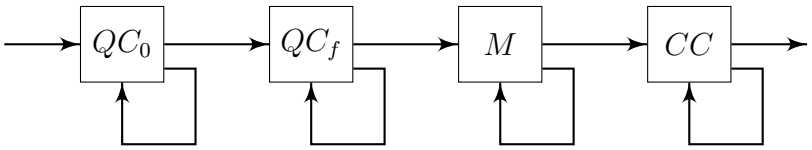
La simulación debe ser realizada teniendo en cuenta el tiempo de procesamiento requerido por la CPU. Es importante investigar como reducirlo para conseguir que la simulación sea eficiente.

Para calcular con precisión los datos medidos sobre cada simulación, se realizan varias ejecuciones, y posteriormente se analiza la media y varianza de las medidas obtenidas. El análisis divide la simulación en cuatro etapas. Las dos primeras, QC_0 y QC_f, realizan la simulación del circuito cuántico. A continuación se analiza el proceso de medición M que provee un nexo entre la parte cuántica y la etapa final de procesado clásico CC.

Cada etapa se mide repetidamente y de forma independiente, permitiendo una mayor precisión en la medida de la complejidad. El esquema se muestra

[1] Firmó las notas sobre la máquina analítica de Babbage en 1843 como A.A.L. para evitar la censura por ser una mujer. Fue la primera persona de la historia que creó un programa para ser ejecutado en una máquina (pese a que aún no había sido construida).

a continuación.



5.1.1 Simulación cuántica

El análisis de la simulación cuántica, se divide en dos procesos. La parte inicial QC_0, calcula el estado intermedio $|\psi_1\rangle$ que es independiente de la función f del problema. Permitiendo la reutilización de los cálculos en las etapas posteriores.

La parte final del análisis QC_f, toma el estado $|\psi_1\rangle$ previamente calculado, y continúa la simulación del circuito hasta el estado final $|\psi_3\rangle$. Para medir el tiempo que toma una etapa se denotará como $T_\mu(x)$ el tiempo medio en segundos de r ejecuciones, y $T_{\sigma^2}(x)$ la varianza del proceso x. Por lo general se realizarán $r = 100$ repeticiones, excepto si el proceso se demora demasiado, entonces se reducirán las iteraciones para mantener un tiempo de simulación razonable.

n	N	r	$T_\mu(QC_0)$	$T_{\sigma^2}(QC_0)$	$T_\mu(QC_f)$	$T_{\sigma^2}(QC_f)$
2	4	100	$8{,}01 \cdot 10^{-3}$	$1{,}97 \cdot 10^{-8}$	$1{,}66 \cdot 10^{-2}$	$1{,}39 \cdot 10^{-8}$
3	6	100	$8{,}08 \cdot 10^{-3}$	$3{,}06 \cdot 10^{-8}$	$3{,}09 \cdot 10^{-2}$	$4{,}00 \cdot 10^{-8}$
4	8	100	$8{,}44 \cdot 10^{-3}$	$2{,}61 \cdot 10^{-8}$	$6{,}11 \cdot 10^{-2}$	$4{,}84 \cdot 10^{-6}$
5	10	100	$9{,}34 \cdot 10^{-3}$	$6{,}39 \cdot 10^{-9}$	$1{,}20 \cdot 10^{-1}$	$2{,}16 \cdot 10^{-6}$
6	12	100	$1{,}34 \cdot 10^{-2}$	$7{,}03 \cdot 10^{-7}$	$2{,}52 \cdot 10^{-1}$	$5{,}78 \cdot 10^{-5}$
7	14	100	$3{,}00 \cdot 10^{-2}$	$1{,}50 \cdot 10^{-5}$	$5{,}93 \cdot 10^{-1}$	$3{,}10 \cdot 10^{-4}$
8	16	100	$9{,}64 \cdot 10^{-2}$	$2{,}20 \cdot 10^{-5}$	$1{,}84 \cdot 10^{0}$	$8{,}02 \cdot 10^{-3}$
9	18	33	$3{,}75 \cdot 10^{-1}$	$1{,}56 \cdot 10^{-5}$	$8{,}69 \cdot 10^{0}$	$4{,}09 \cdot 10^{-1}$
10	20	6	$1{,}57 \cdot 10^{0}$	$3{,}39 \cdot 10^{-4}$	$5{,}61 \cdot 10^{1}$	$2{,}87 \cdot 10^{1}$

Tabla 5.1: Tiempo empleado por QC_0 y QC_f en segundos.

Se observa en la tabla 5.1 como se incrementa el tiempo a medida que crece en número de qubits. La parte final QC_f requiere un orden de magnitud superior que QC_0. Ambas requieren un tiempo superior a $O(2^N)$.

5.1.2 Tiempo empleado en M y CC

Tras calcular el estado final del circuito, se analiza el proceso de medición M. Debe calcularse la distribución de probabilidad que asocia a cada

posible valor tras medir una línea, la probabilidad de que ocurra. Dado que cada línea tiene n qubits, serán necesarias $O(2^n)$ operaciones. El proceso de cálculo clásico, resuelve el sistema de ecuaciones, una vez se han obtenido $n-1$ vectores. El tiempo obtenido en estas dos últimas etapas, pese a ser exponencial, es mucho menor que en la simulación del circuito cuántico, como se observa en la tabla 5.2.

n	N	r	$T_\mu(M)$	$T_{\sigma^2}(M)$	$T_\mu(CC)$	$T_{\sigma^2}(CC)$
2	4	100	$1{,}25 \cdot 10^{-3}$	$8{,}56 \cdot 10^{-10}$	$1{,}38 \cdot 10^{-4}$	$8{,}17 \cdot 10^{-9}$
3	6	100	$1{,}30 \cdot 10^{-3}$	$2{,}03 \cdot 10^{-10}$	$2{,}37 \cdot 10^{-4}$	$1{,}01 \cdot 10^{-8}$
4	8	100	$1{,}43 \cdot 10^{-3}$	$1{,}22 \cdot 10^{-9}$	$3{,}21 \cdot 10^{-4}$	$1{,}00 \cdot 10^{-8}$
5	10	100	$1{,}65 \cdot 10^{-3}$	$4{,}15 \cdot 10^{-10}$	$3{,}70 \cdot 10^{-4}$	$9{,}09 \cdot 10^{-9}$
6	12	100	$2{,}31 \cdot 10^{-3}$	$1{,}55 \cdot 10^{-9}$	$4{,}61 \cdot 10^{-4}$	$2{,}72 \cdot 10^{-8}$
7	14	100	$4{,}46 \cdot 10^{-3}$	$2{,}53 \cdot 10^{-7}$	$5{,}50 \cdot 10^{-4}$	$1{,}29 \cdot 10^{-8}$
8	16	100	$1{,}50 \cdot 10^{-2}$	$2{,}30 \cdot 10^{-7}$	$6{,}60 \cdot 10^{-4}$	$8{,}28 \cdot 10^{-9}$
9	18	33	$7{,}33 \cdot 10^{-2}$	$1{,}24 \cdot 10^{-6}$	$7{,}95 \cdot 10^{-4}$	$8{,}80 \cdot 10^{-9}$
10	20	6	$5{,}01 \cdot 10^{-1}$	$5{,}95 \cdot 10^{-4}$	$1{,}02 \cdot 10^{-3}$	$6{,}94 \cdot 10^{-10}$

Tabla 5.2: Tiempo empleado por M y CC en segundos.

5.1.3 Tiempo total de la simulación

Debido al carácter exponencial que presenta la simulación, es conveniente emplear una representación con escala logarítmica. En la figura 5.1 se observa como crecen los tiempos de ejecución con el número de qubits. Para $N = 20$ es necesario un tiempo de simulación aproximado de un minuto. La simulación, pese a ser extremadamente costosa, se puede esbozar para $N \leq 20$.

5.2 Análisis de espacio

Para observar el comportamiento de un algoritmo es importante tener en cuenta además del tiempo que requiere su ejecución, el espacio que emplea. El simulador será analizado paso a paso mostrando como varía la memoria empleada a medida que crece el número de qubits del circuito.

5.2.1 Simulación cuántica

La simulación del circuito cuántico se divide en dos etapas, la parte inicial, hasta el estado $|\psi_1\rangle$ denominada QC_0, y la parte final, hasta el estado $|\psi_3\rangle$,

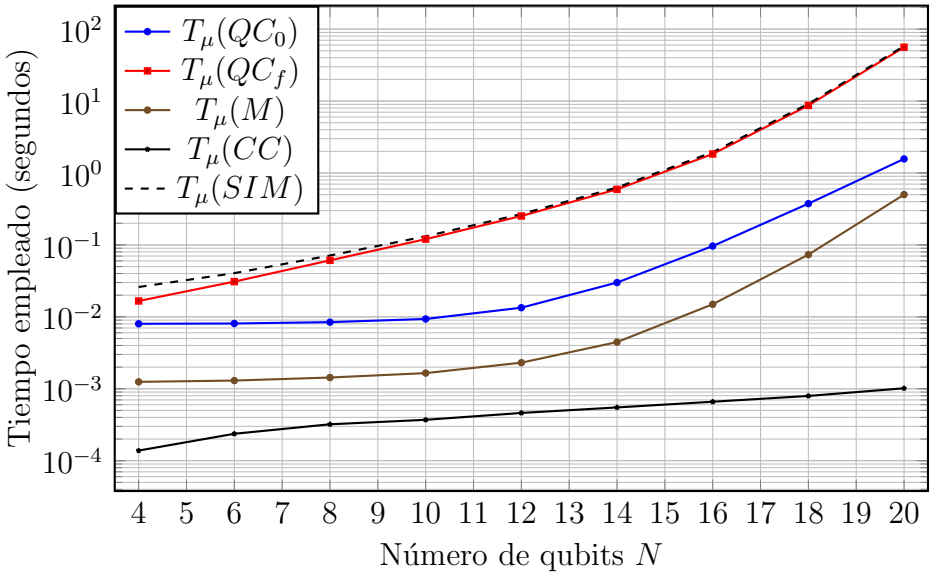
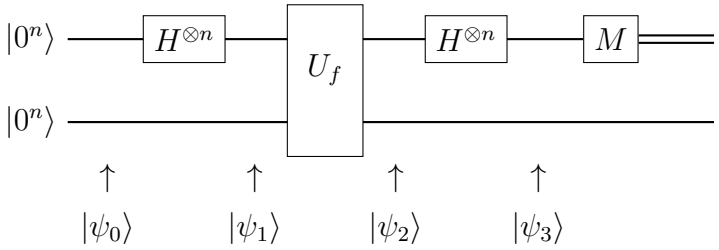
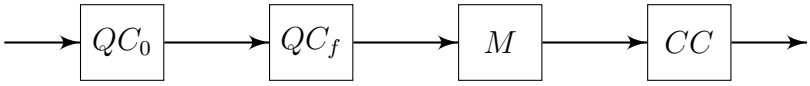


Figura 5.1: Tiempo de simulación en escala logarítmica.

denominada QC_f. El esquema del circuito cuántico muestra el orden de los operadores, y la posición de los estados.



Esquema de análisis de memoria de la simulación completa.



Análisis de la primera parte QC_0

En QC_0, será necesario almacenar en la memoria el operador de Hadamard $H^{\otimes n}$, además de un estado cuántico $|\psi\rangle$ de $N = 2n$ qubits. Al analizar el comportamiento del circuito, se obtiene la tabla 5.3. Sea $S(x)$ el tamaño del objeto x en bytes, y S_T el tamaño total requerido por la simulación. Entonces el espacio S'_T para QC_0 será una aproximación al espacio real S_T determinado como

$$S_T = S(H) + S(|\psi_1\rangle)$$

n	N	$\log_2 S(H)$	$\log_2 S(\|\phi_1\rangle)$	$\log_2 S_T$	$\log_2 S'_T$
2	4	7.21	6.64	7.95	7.58
3	6	9.10	8.34	9.77	9.58
4	8	11.05	10.17	11.68	11.58
5	10	13.02	12.09	13.63	13.58
6	12	15.01	14.04	15.61	15.58
7	14	17.01	16.02	17.60	17.58
8	16	19.00	18.01	19.59	19.58
9	18	21.00	20.01	21.59	21.58
10	20	23.00	22.00	23.59	23.58

Tabla 5.3: Espacio empleado por QC_0 en escala logarítmica (bytes).

Y aproximado mediante

$$\begin{aligned}
&\log_2 S(H) \approx N + 3 \\
&\log_2 S(|\psi_1\rangle) \approx N + 2 \\
&\log_2 S_T \approx N + \log_2(2^2 + 2^3) = N + 3{,}58 = \log_2 S'_T \\
&S'_T = 2^N \log_2 12
\end{aligned}$$

Se observa como el espacio requerido aumenta de forma exponencial a medida que aumenta el número de qubits del sistema y se encuentra en torno a $O(2^N)$.

Análisis de la parte final QC_f

En la etapa final de la simulación del circuito, se sobreescribe $|\psi_1\rangle$ con el estado final $|\psi_3\rangle$, de modo que sólo será necesario un espacio equivalente al del más grande. Además del estado, se necesita el operador U_f, que se calcula a partir de la función dada f. Los tamaños se muestran en la tabla 5.4.

El tamaño requerido en esta etapa de la simulación será S_T, aproximado a S'_T. Además el operador H se reutiliza de la etapa inicial, de modo que ese espacio debe tenerse en cuenta. El tamaño del estado $|\psi_3\rangle$ es siempre un poco más grande que $|\psi_1\rangle$. Dado que $|\psi_1\rangle$ se reemplazará por $|\psi_3\rangle$, será necesario el tamaño del más grande, siendo este $S(|\psi_3\rangle)$. El tamaño total S_T será

$$S_T = S(H) + S(U) + S(|\psi_3\rangle)$$

n	N	$\log_2 S(H)$	$\log_2 S(U)$	$\log_2 S(\|\phi_3\rangle)$	$\log_2 S_T$	$\log_2 S'_T$
2	4	7.21	7.61	6.64	8.79	8.70
3	6	9.10	9.59	8.60	10.74	10.70
4	8	11.05	11.59	10.59	12.72	12.70
5	10	13.02	13.59	12.59	14.71	14.70
6	12	15.01	15.59	14.59	16.70	16.70
7	14	17.01	17.58	16.59	18.70	18.70
8	16	19.00	19.58	18.58	20.70	20.70
9	18	21.00	21.58	20.58	22.70	22.70
10	20	23.00	23.58	22.58	24.70	24.70

Tabla 5.4: Espacio empleado por QC_f en escala logarítmica (bytes).

Que de forma aproximada S'_T, se determina como

$$\log_2 S(H) \approx N + \log_2 8 = N + 3$$
$$\log_2 S(U) \approx N + \log_2 12 \approx N + 3{,}58$$
$$\log_2 S(|\psi_3\rangle) \approx N + \log_2 6 \approx N + 2{,}58$$
$$\log_2 S_T \approx \log_2 S'_T = N + \log_2(8 + 12 + 6) \approx N + 4{,}70$$
$$S'_T = 2^N \log_2 26$$

Mostrando de nuevo el carácter exponencial del espacio requerido, en torno a $O(2^N)$. Dado que el espacio empleado para QC_f es superior a QC_0, se usará $S'_T = 2^N \log_2 26$ como aproximación a la memoria requerida por la simulación de todo el circuito.

Limitaciones de la simulación

El espacio de simulación tiene una memoria finita, fijada en 512MB, un total de 2^{29} bytes. En la figura 5.2 se observa como crece el espacio necesario a medida que aumentan los qubits. Dada la complejidad de carácter exponencial, la cantidad de qubits simulados por el circuito se ve severamente limitada por $S_T \leq 2^{29}$. Que de forma aproximada, se obtiene

$$S'_T = 2^N \log_2 26 \leq 2^{29}$$
$$N + \log_2 26 \leq 29$$
$$N \leq 29 - \log_2 26 \approx 24{,}30$$
$$N \leq 24$$

Por lo tanto, sólo es posible simular cicuitos de hasta un máximo de 24 qubits, en el entorno de simulación actual. Dado que el problema requiere dos líneas

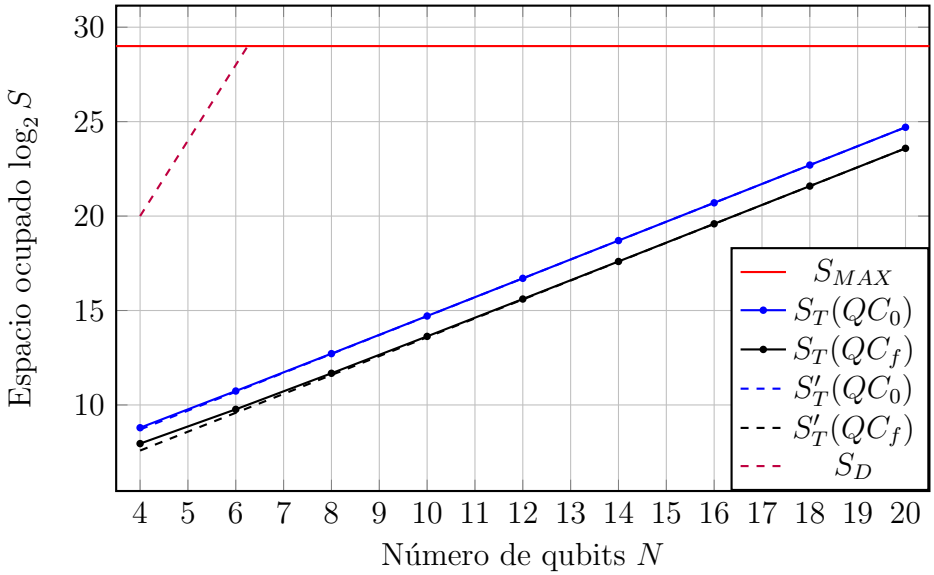


Figura 5.2: Espacio necesario para la simulación. Se muestra en línea continua el espacio real, y en discontinua el aproximado. El espacio requerido sin emplear matrices huecas, usando matrices densas se muestra como S_D. La memoria disponible para la simulación es $S_{MAX} = 2^{29}$.

de n bits, son necesarios $N = 2n$ qubits. Obteniendo la limitación $n \leq 12$.

5.2.2 Medición y computación clásica

Una vez obtenido el estado $|\psi_3\rangle$, tras la costosa simulación, el resto de componentes del simulador apenas requieren memoria. El proceso de medición, denominado M, toma el vector $|\psi_3\rangle$ y calcula la distribución de probabilidad resultante de medir la línea superior. Esta distribución tiene 2^n posibles resultados, que se almacenan en un vector de probabilidades v_p junto con otro vector del resultado correspondiente v_n. Con un tamaño

$$S(v_p) = S(v_n) = 2^n \cdot 4 = 2^{N/2} \cdot 4$$

Obteniéndose un espacio requerido para la medición

$$S_T(M) = S(v_p) + S(v_n) = 2^{N/2} \cdot 8$$

Debido a que tras el proceso de simulación, al finalizar la etapa QC_f, ya no es necesario mantener los operadores U_f y H almacenados, se libera un espacio

$$S(U_f) + S(H) = 2^N \log_2 20 \approx 2^N \cdot 4{,}32$$

Más que suficiente para los vectores v_p y v_n con $N \geq 2$.

El proceso de cómputo clásico posterior, denotado CC, requiere que M mantenga las distribuciones de probabilidad en memoria. Además, calcula para cada medición, el conjunto de combinaciones lineales de vectores acumulados. Como se requieren $n-1$ vectores, será necesario un espacio para los vectores acumulados v_a dado por

$$S_T(CC) = S(v_a) = (2^{n-1} - 1) \cdot 4 = 2^{N/2} \cdot 2 - 4$$

Conjuntamente con el requerido por M

$$S_T(M) + S_T(CC) = 2^{N/2} \cdot 10 - 4$$

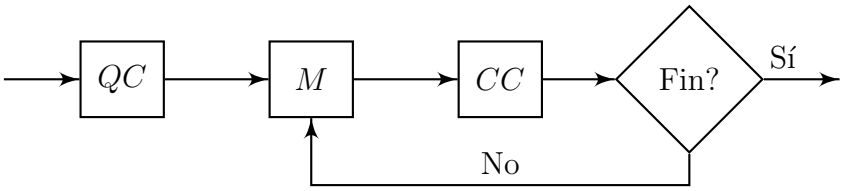
Que continúa siendo inferior al liberado por los operadores tras terminar QC_f para $N \geq 2$. Por tanto, la etapa final de la simulación cuántica QC_f es el cuello de botella que limita el espacio que empleará toda la simulación.

5.3 Complejidad del circuito cuántico

Tras analizar la eficiencia midiendo el tiempo y la memoria de la simulación, el último paso consiste en examinar la complejidad del circuito cuántico. Esta complejidad debe ser la observada en un ordenador cuántico real, y debe coincidir también con los cálculos teóricos.

El escenario de análisis consistirá en una serie de simulaciones completas, observando el número de ejecuciones necesarias del circuito cuántico para completar el problema.

Para evitar la costosa simulación sucesiva de la parte cuántica, se empleará el diseño optimizado que permite reutilizar los cálculos, almacenando el estado final. De esta forma se consigue que la simulación cuántica sólo sea necesaria realizarla una única vez. El esquema de la simulación se presenta a continuación.



Tras repetir la simulación una cantidad $r = 10000$ veces, se analiza la media y varianza, y se comparan con las calculadas previamente de forma teórica. En la tabla 5.5 se puede observar como ambos resultados se acercan mucho. En la figura 5.3 se observa como el crecimiento es lineal, de modo que la complejidad se aproxima a $O(n)$.

n	N	R_μ	R_{σ^2}	R'_μ	R'_{σ^2}
2	4	1,998	1,989	2,000	2,000
3	6	3,354	2,527	3,333	2,444
4	8	4,461	2,471	4,476	2,608
5	10	5,524	2,578	5,543	2,679
6	12	6,559	2,734	6,575	2,712
7	14	7,608	2,774	7,591	2,728
8	16	8,593	2,637	8,599	2,736
9	18	9,585	2,718	9,603	2,740
10	20	10,602	2,725	10,605	2,742

Tabla 5.5: Ejecuciones del algoritmo cuántico R experimentales, R' teóricas.

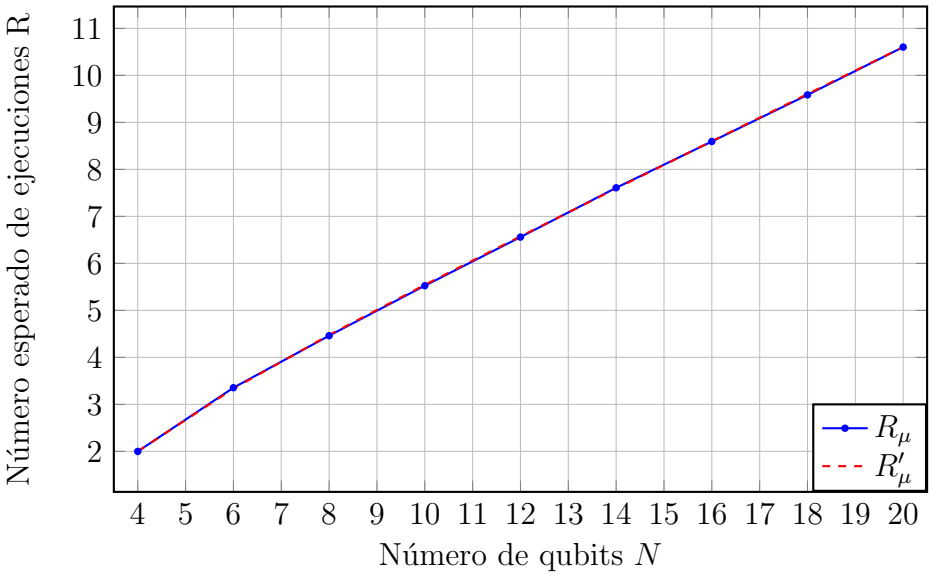


Figura 5.3: Número esperado de iteraciones a medida que aumentan los qubits. Se observa el valor experimental R_μ comparado con el teórico R'_μ.

Bibliografía

[1] A. Einstein, B. Podolsky y N. Rosen: *Can Quantum-Mechanical description of reality be considered complete?* Institute for Advanced Research, Marzo 1935.

[2] Cajori, Florian: *A History Of Mathematical Notations Vol II.* página 134, 1928.

[3] Dattani, N. S. y N. Bryans: *Quantum factorization of 56153 with only 4 qubits.* ArXiv e-prints, Noviembre 2014.

[4] Deutsch, D.: *Quantum theory, the Church-Turing principle and the universal quantum computer.* Proceedings of the Royal Society of London Series A, 400:97–117, Julio 1985.

[5] Feynman, Richard P.: *Simulating physics with computers.* International Journal of Theoretical Physics, 21(6):467–488.

[6] Grover, Lov K.: *A fast quantum mechanical algorithm for database search.* Proceedings, STOC, Mayo 1996.

[7] Lehman, Eric y Tom Leighton: *Mathematics for Computer Science.* 2004.

[8] Nielsen, Michael A. y Isaac L. Chuang: *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[9] Ross, Sheldon: *Simulación.* Prentice Hall, 1999.

[10] Shor, Peter W.: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.* SIAM Journal on Computing, 26(5):1484–1509, Octubre 1997.

[11] Shor, Peter W.: *Why Haven't More Quantum Algorithms Been Found?* Journal of the ACM, 50(1):87–90, 2003.

[12] Simon, Daniel R.: *On the Power of Quantum Computation.* SIAM Journal on Computing, 26:116–123, 1994.