Arquitecturas y Programación de Computadores Cuánticos
Práctica 4: Los Algoritmos de Deutsch y Deutsch-Josza

Prof. Alberto A. del Barrio
Prof. Guillermo Botella

Índice

1. Objetivos

En esta práctica nos adentraremos en algunos de los algoritmos denominados *canónicos*, ya que se desarrollaron en los años tempranos de la Computación Cuántica y fueron los primeros en probar la potencial aceleración de los computadores cuánticos. En concreto,:

- Estudiaremos los algoritmos de Deutsch y Deutch-Jozsa, como una primera aproximación a los algoritmos cuánticos basados en *oráculo*.
- Nos familiarizaremos con el entorno de programación Forest.

2. Algoritmos basados en oráculo

Antes de profundizar en algoritmos canónico concretos, comentaremos ciertos aspectos comunes. Muchos de estos algoritmos cuánticos también son llamados *de caja negra* (black box) o *de consulta* (query model). En estos casos, hay una función f desconocida, pero seremos capaces de construir otra función U_f, denominada *oráculo*, a la cual preguntar para determinar la relación de ciertas entradas con ciertas salidas de la función f. Una de las formas de construir dicho oráculo es siguiendo la condición mostrada por la Ecuación 1.

$$U_f|x\rangle = |x \oplus f(x)\rangle \ , \tag{1}$$

donde $\oplus$ es la suma módulo-2. Así a simple vista, el uso de un oráculo puede parecer cuestión de magia, o que estamos haciendo trampa, pero debemos de pensar en U_f como si fuera una puerta más, aunque su implementación sea más compleja. Por ejemplo, aplicar una Hadamard consiste en multiplicar por la izquierda con la matriz H, al igual que hacemos con el oráculo en la Ecuación 1.

La parte interesante sobre desarrollar algoritmos cuánticos basados en el modelo de consulta es que obtenemos una cota inferior del número de pasos (puertas) de dicho algoritmo. Cada consulta será al menos un paso del algoritmo, por lo que si no se puede resolver un problema con un número moderado de consultas, ciertamente no podrá resolverse con puertas.

3. Algoritmo de Deutsch

El algoritmo de Deutsch [1] fue el primero en demostrar una ventaja clara de la Computación Cuántica frente a la clásica. El problema planteado por David Deutsch consiste en una caja negra que calcula una función booleana f de un bit. Podemos representar f como define la Ecuación 2.

$$f : \{0, 1\} \to \{0, 1\} \ . \tag{2}$$

Tabla 1: Funciones booleanas de 1-bit.

x	f_0	f_1	f_x	$f_{\overline{x}}$
0	0	1	0	1
1	0	1	1	0

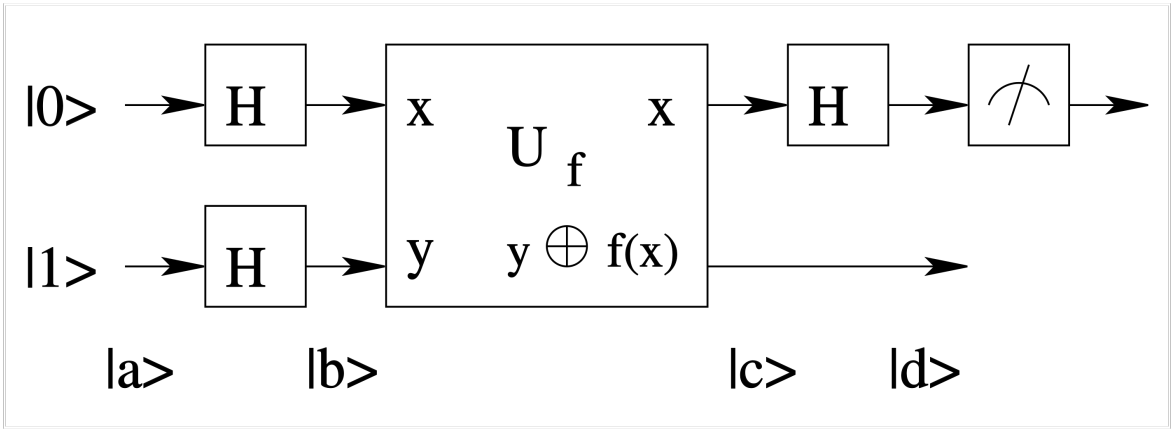

Figura 1: Esquema del circuito para resolver el problema de Deutsch. Tomado de [4].

Intuitivamente, aunque pueda parecer un problema poco realista, la caja negra podría ser una complicada función como un algoritmo de rutado, cuya salida (0 o 1) indica la ruta a tomar.

Formalmente, el problema de Deutsch se formula de la siguiente manera: dada una función f de un bit, determinar, consultando la función el menor número de veces posible, si la función es *balanceada* o *constante*. Una función constante será aquélla cuya salida siempre sea 0 o 1, mientras que una función balanceada tendrá tantas salidas igual a 0 como salidas igual a 1.

Existen exactamente 4 funciones con un bit de entrada y uno de salida, tal y como muestra la Tabla 1. Como puede observarse, f_0 y f_1 son funciones constantes, mientras que f_x y $f_{\overline{x}}$ son funciones balanceadas.

Desde el punto de vista clásico, está claro que debemos realizar dos consultas ($f(0)$ y $f(1)$). Sin embargo, y he aquí donde radica el destacable descubrimiento de Deutsch, es que desde el punto de vista cuántico solamente hace falta una consulta. La Figura 1 muestra el esquema para resolver el problema de Deutsch.

Dado que las función elegida puede ser arbitrariamente constante o balanceada, es posible que una salida no tenga inversa (es decir, f no es inyectiva). Para ser capaces de mantener la reversibilidad del circuito, hay que emplear una idea típica en Computación Cuántica: añadir un qubit de más. Este qubit será el qubit de la línea inferior en la Figura 1, el cual

debe inicializarse a $|1\rangle$, de ahí que se utilice una puerta X al comienzo. Es sobre este qubit que aplicaremos el oráculo. El *truco* de Deutsch consiste en aplicar el oráculo a un qubit en superposición, de lo contrario estaríamos en el mismo caso que un algoritmo clásico. Dicho oráculo se define como especifica la Ecuación 3.

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle \ , \tag{3}$$

donde $|x\rangle$ representa el qubit de datos e $|y\rangle$ el qubit auxiliar. Al aplicar las puertas Hadamard al comienzo del circuito de la Figura 1 ponemos ambos qubits en superposición. Al poner en superposición el qubit $|y\rangle$, dado que se inicializa a $|1\rangle$, tomará el valor $|-\rangle$. Por tanto, al aplicar el oráculo, y teniendo en cuenta la linealidad de la operación $\oplus$, obtendremos la Ecuación 4.

$$\begin{aligned} U_f(|x\rangle|-\rangle) &= |x\rangle|\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \oplus f(x)\rangle \\ &= |x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= |x\rangle\frac{1}{\sqrt{2}}(|f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= (-1)^{f(x)}|x\rangle|-\rangle \ . \end{aligned} \tag{4}$$

Para comprender el último paso en la Ecuación 4, hay que considerar los dos posibles valores de $f(x)$, sea cual fuera f.

- Supongamos que $f(x) = 0$, entonces la salida del oráculo será $|x\rangle|-\rangle$.
- Supongamos que $f(x) = 1$, entonces dentro del paréntesis de la derecha tendremos la expresión $|1\rangle - |0\rangle$, por lo que la salida del oráculo será $(-1)|x\rangle|-\rangle$.

Una demostración más completa puede encontrarse en [3].

El *truco de Deutsch* es lo que se conoce técnicamente como *phase kickback*. Al aplicar el oráculo sobre un qubit en superposición, el valor de la función quedará codificado en la fase, (*kicked back in the phase*). Debe notarse que para funciones constantes, la fase aplicada será la misma para todas las entradas $|x\rangle$, mientras que para funciones balanceadas, la fase variará según el valor de $|x\rangle$.

3.1. Construyendo el oráculo

El esquema de la Figura 1 es sencillo, pero en la práctica cómo se implementa el oráculo? La clave está en el último párrafo de la sección anterior. **En funciones constantes, la fase aplicada será siempre la misma, en funciones balanceadas, la fase variará dependiendo de la entrada.** Siguiendo con las 4 funciones de la Tabla 1, para entender

Tabla 2: Definición de oráculos para las funciones booleanas de 1-bit.

$\|xy\rangle$	$f_0 \equiv I$	$f_1 \equiv NOT(y)$	$f_x \equiv CNOT(xy)$	$f_{\overline{x}} \equiv NOT(f_x)$
$\|00\rangle$	$\|00\rangle$	$\|01\rangle$	$\|00\rangle$	$\|01\rangle$
$\|01\rangle$	$\|01\rangle$	$\|00\rangle$	$\|01\rangle$	$\|00\rangle$
$\|10\rangle$	$\|10\rangle$	$\|11\rangle$	$\|11\rangle$	$\|10\rangle$
$\|11\rangle$	$\|11\rangle$	$\|10\rangle$	$\|10\rangle$	$\|11\rangle$

la estructura del oráculo, basta con aplicar la Ecuación 3. Podemos ver las transformaciones asociadas en la Tabla 2.
Por ejemplo, si aplicamos la Ecuación 3 sobre el estado $|11\rangle$ con la función constante f_0, obtendremos:

$$U_{f_0}|1\rangle|1\rangle = |1\rangle|1 \oplus f_0(1)\rangle = |1\rangle|1 \oplus 0\rangle = |1\rangle|1\rangle \ .$$

O si aplicamos la Ecuación 3 sobre el estado $|10\rangle$ con la función balanceada f_x, obtendremos:

$$U_{f_x}|1\rangle|0\rangle = |1\rangle|0 \oplus f_x(1)\rangle = |1\rangle|0 \oplus 1\rangle = |1\rangle|1\rangle \ .$$

Estas transformaciones se corresponden con las equivalencias mostradas en la cabecera de la Tabla 2. Una vez tenemos los oráculos, siguiendo el esquema mostrado en la Figura 1 será posible diseñar el circuito que resuelva el problema de Deutsch.

Nota: para probar todas las funciones, es recomendable crear un *diccionario de oráculos* y posteriormente llamar a un oráculo concreto desde el programa principal, en lugar de escribir un código entero para cada función a probar.

4. Algoritmo de Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa (DJ) [2] es una generalización del problema de Deutsch para funciones de n-bits de entrada. Dicho problema consiste en decidir si una función f no es constante (A), o si no es balanceada (B). Desde el punto de vista clásico, será necesario consultar como máximo la mitad de casos más uno, es decir, $2^{(n-1)} + 1$. Nuevamente, desde el punto de vista cuántico esto se consigue con una única consulta al oráculo.

Para entender la aproximación clásica, consideremos un ejemplo con $n = 3$.

- Consultamos $f(000)$ al oráculo. Sin pérdida de la generalidad, supongamos que la salida es 0. No podemos decidir nada todavía, hemos de probar más casos.

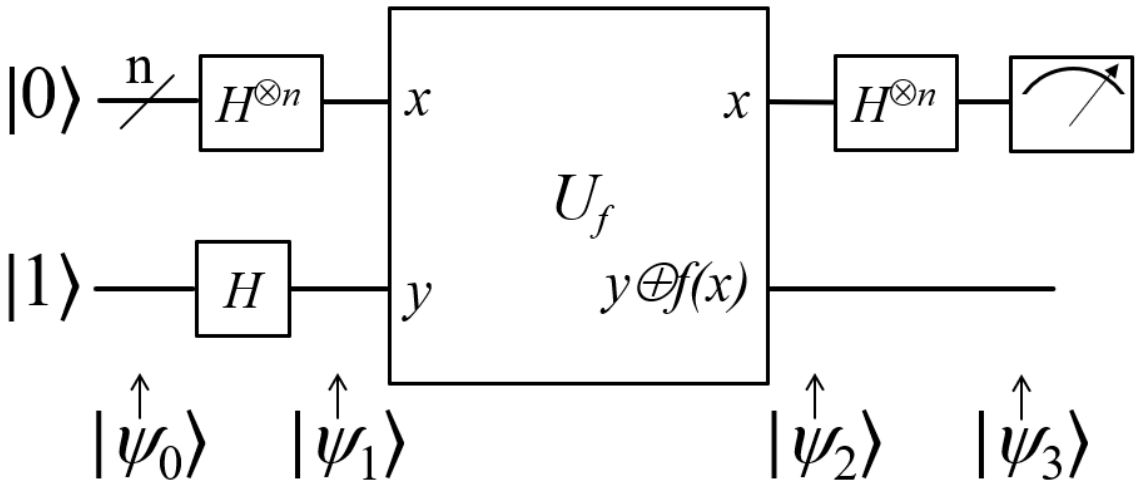

Figura 2: Esquema del circuito para resolver el problema de Deutsch-Jozsa. Tomado de [3].

- Consultamos $f(001)$ al oráculo. Si la salida es 1, (A) es cierto y hemos terminado. Si la salida es 0, probamos otro caso.
- Consultamos $f(010)$ al oráculo. Si la salida es 1, (A) es cierto y hemos terminado. Si la salida es 0, probamos otro caso.
- Consultamos $f(011)$ al oráculo. Si la salida es 1, (A) es cierto y hemos terminado. Si la salida es 0, probamos otro caso.
- Consultamos $f(100)$ al oráculo. Si la salida es 1, (B) es cierto y hemos terminado. Si la salida es 0, (B) también será cierto. En cualquiera de los casos la función nunca será balanceada.

Ha de notarse que cada entrada puede ser arbitraria, por lo que en lugar de probar las entradas en un orden, podemos probar cualquier permutación de 0s y 1s.

Sin profundizar en detalles matemáticos, que son análogos al caso del algoritmo de Deutsch, el circuito que resuelve el problema DJ se muestra en la Figura 2 y solo requiere de una consulta al oráculo, el cual se define de la misma manera que en el algoritmo de Deutsch. Por tanto, los pasos a seguir para implementar un circuito que resuelva el problema DJ son los siguientes:

- Preparar un bit auxiliar (*ancilla*) y forzarlo al valor $|1\rangle$.
- Aplicar $n + 1$ puertas Hadamard sobre todos los qubits en paralelo (una por qubit).
- Consultar al oráculo. Este paso es específico de cada función f.

- Aplicar n puertas Hadamard en paralelo, una por cada qubit de datos.
- Medir los qubits de datos. Si el resultado es todo 0s, entonces la función es constante. De lo contrario, es balanceada.

5. Desarrollo de la práctica

En esta práctica se proponen varios ejercicios.

Ejercicio 1. Implementar con Forest el algoritmo de Deutsch. Mostrar y comentar resultados.

Ejercicio 2. Implementar con Forest el algoritmo de Deutsch-Jozsa para 2 y 3 qubits. Probar las funciones constantes y al menos 2 funciones balanceadas por caso. Mostrar y comentar resultados.

Ejercicio 3. Repetir los ejercicios 1 y 2 codificándolos con Qiskit y realizando simulaciones y ejecuciones en backends reales.

Entregables: todos los códigos desarrollados en formato texto, así como una memoria con respuestas justificadas a los ejercicios propuestos.

Referencias

[1] Deutsch, D., Quantum theory, the Church–Turing principle and the universal quantum computer, *Proc. R. Soc. Lond. A*, 400:97–117, 1985.

[2] Deutsch, D. and Jozsa, R., Rapid solution of problems by quantum computation, *Proc. R. Soc. Lond. A*, 439:553–558, 1992.

[3] Deutsch-Jozsa algorithm. `https://en.wikipedia.org/wiki/Deutsch-Jozsa\_algorithm`

[4] Algorithmo de Deutsch-Jozsa. `https://es.wikipedia.org/wiki/Algoritmo\_de\_Deutsch-Jozsa`