El diablo de los números

Sección a cargo de

Javier Cilleruelo Mateo

Factorización polinomial de números enteros

por

Jesús García López de Lacalle[1]

Introducción

En este artículo presentamos el algoritmo polinomial de factorización de números enteros obtenido por Peter W. Shor [19] en 1994. Se trata del resultado más prometedor del modelo cuántico de computación desarrollado en los últimos años. Este resultado rompe teóricamente uno de los protocolos criptográficos de clave pública más importantes, el protocolo RSA. Afortunadamente, el propio modelo de computación cuántica permite la implementación de protocolos criptográficos de clave privada seguros.

La computación cuántica empezó a desarrollarse en la década de los ochenta a raíz de las propuestas de Paul Benioff, David Deutsch y Richard Feynman. En 1982 Benioff [2] y Feynman [10] sugirieron independientemente que, dado el elevado coste computacional del cálculo de la evolución de sistemas cuánticos, la evolución de estos sistemas se podría considerar como una herramienta de cálculo más que como un objeto a calcular. Poco después, en 1985, y también de forma independiente Deutsch [8] propone la búsqueda de un ordenador que sea capaz de simular eficientemente un sistema físico arbitrario. La conjunción de todas estas ideas ha conducido a la concepción actual de ordenador cuántico.

Cuestionar el sistema de computación clásico, que cuenta con una sólida base teórica y con el aval de infinidad de aplicaciones en todos los ámbitos de la vida cotidiana, sólo tiene sentido si el modelo que se propone como alternativo es potencialmente mejor que el actual. Efectivamente así lo hacen Benioff,

[1]Este trabajo ha sido subvencionado por el MCYT, proyecto TIC2002-01541.

Deutsch y Feynman, fundamentando sus propuestas sobre la posibilidad de que los sistemas cuánticos tengan mayor potencia de cálculo que los clásicos. El argumento que todos utilizan para apuntar esta posibilidad es el hecho de que la simulación de un ordenador cuántico (sistema cuántico) en un ordenador clásico requiere una gran cantidad de operaciones.

El principal método para aumentar la capacidad de cálculo de un ordenador clásico es el procesamiento en paralelo. Los ordenadores que soportan este esquema de programación disponen de varios cientos o miles de procesadores. Sabemos que la capacidad de almacenamiento de información y la capacidad de cálculo de un ordenador son proporcionales al número de celdas de memoria y al número de procesadores respectivamente, es decir, al tamaño del ordenador. Entonces la capacidad de un ordenador clásico (de almacenamiento y de cálculo) crece linealmente con respecto a su tamaño.

En un ordenador cuántico la situación cambia por completo, hasta el punto que su capacidad crece exponencialmente con respecto a su tamaño. Este hecho, estrechamente relacionado con el principio de superposición de la mecánica cuántica, se denomina paralelismo cuántico. Llamamos *qubits* o *bits cuánticos* a los sistemas cuánticos elementales, es decir, a los sistemas cuánticos de dos estados. Los sistemas cuánticos de n *qubits* se describen mediante vectores de un espacio de Hilbert complejo de dimensión 2^n. Esto permite codificar una cantidad exponencial de información en el estado de un sistema cuántico de n *qubits*. Además, cualquier tranformación del estado del sistema se traduce en la modificación simultánea de toda la información almacenada. Por tanto, la capacidad de un ordenador cuántico (tanto de almacenamiento como de cálculo) crece exponencialmente con respecto a su tamaño.

Sin embargo, la medición de estados cuánticos es un inconveniente importante para la computación cuántica. Hay que recordar que las medidas cuánticas no son deterministas. Esto quiere decir, por ejemplo, que si medimos dos estados iguales los resultados no tienen por qué ser iguales. El proceso de medida es, por tanto, un experimento aleatorio en el que la probabilidad de cada resultado está determinada por el estado del sistema.

Las dificultades para sacar provecho del paralelismo cuántico son tan notables que hubo que esperar más de una década para encontrar el primer gran resultado. En 1994 Peter W. Shor sorprendió a todos presentando sendos algoritmos polinomiales para factorizar números enteros y para calcular logaritmos discretos [19]. Fueron los primeros problemas relevantes en los que se alcanzaba una aceleración exponencial con respecto a los mejores algoritmos clásicos conocidos. A raíz de este descubrimiento se generó una gran actividad, tanto en el desarrollo de la tecnología necesaria para la construcción de ordenadores cuánticos como en el estudio de algoritmos cuánticos.

El algoritmo de Shor rompió teóricamente el sistema criptográfico más difundido en la actualidad, el sistema RSA propuesto por Rivest, Shamir y Adleman en 1978 [18]. Este hecho contribuyó a su vez al desarrollo de los sistemas criptográficos cuánticos [3]. Las técnicas que se utilizan para garantizar la confidencialidad de los canales cuánticos se apoyan en una propiedad

característica de la mecánica cuántica: los estados cuánticos no se pueden copiar (clonar) [24]. En el área de las comunicaciones, además del estudio de la confidencialidad, se están investigando otros problemas como, por ejemplo, la codificación de información clásica en canales cuánticos y el teletransporte de estados cuánticos.

Sin embargo, el estudio de este modelo de computación apenas si ha comenzado. Hasta el momento, sólo se han desarrollado ordenadores cuánticos basados en resonancia magnética nuclear [6, 12], en trampas de iones [5] y en cavidades cuánticas [9]. Con la primera de estas técnicas se han conseguido prototipos de hasta *10 qubits*, sobre los que se ha probado el algoritmo de Shor. También se ha propuesto la construcción de ordenadores cuánticos aprovechando los conocimientos actuales sobre semiconductores [16, 23], aunque esta técnica está menos desarrollada.

Desde el punto de vista algorítmico, sólo se ha podido hacer efectiva una ganancia exponencial en el cálculo de transformadas de Fourier y, en estos momentos, ésta es la herramienta más importante de la computación cuántica. Otra técnica que permite mejorar la complejidad de algunos algoritmos clásicos, aunque con ganancia solamente cuadrática, es el método de Grover de búsqueda en conjuntos desordenados [14].

Los ordenadores cuánticos, a diferencia de los clásicos, son dispositivos analógicos. Este hecho plantea mayores dificultades para la construcción de ordenadores cuánticos que las que se tuvieron que afrontar para ordenadores clásicos. En primer lugar, los estados cuánticos se modifican por la influencia del *entorno*. Esto provoca el fenómeno denominado decoherencia que se convierte en una fuente de errores. Y, en segundo lugar, la imprecisión del propio ordenador cuántico al aplicar el algoritmo constituye una nueva fuente de errores. Estas son las razones fundamentales por las que la computación cuántica requiere un mecanismo para acotar la acumulación de errores durante el proceso de cálculo. Para ello hay que superar dificultades importantes: los errores cuánticos son continuos, no se pueden copiar estados y, hasta el final, no se puede leer (medir) la información codificada en un estado cuántico. Estos obstáculos fueron finalmente superados, una vez que Shor [20] y Steane [21] establecieran las ideas básicas para la construcción de códigos cuánticos y se formalizase de forma consistente la teoría cuántica de códigos [7, 13].

Contenido

El artículo está organizado en seis secciones, cinco apéndices y una lista de referencias. En la primera sección hemos dibujado a grandes rasgos la computación cuántica, destacando el enorme impacto que el algoritmo de Shor produjo en la comunidad científica y apuntando las principales dificultades a las que se enfrenta el nuevo modelo de computación.

En la sección siguiente presentamos el modelo cuántico de computación. Para entenderlo no se precisan conocimientos específicos de física cuántica.

El modelo, como tal, es una estructura matemática que únicamente requiere resultados básicos de algebra lineal y multilineal. Recordamos las propiedades esenciales del modelo clásico e introducimos el modelo cuántico describiendo los tres procesos básicos de cualquier modelo de computación: cómo codificar información, cómo transformar la información codificada (cálculo) y cómo obtener los resultados del cálculo (leer la información). Para finalizar la sección presentamos como ejemplo el problema de Simon. Fue el primer problema en el que se demostró que el modelo cuántico es exponencialmente más rápido que el clásico y jugó un papel importante en la obtención del algoritmo de factorización de Shor.

En la sección cuarta estudiamos el problema clásico de factorización, reduciéndolo al problema de calcular el periodo de una función discreta. Aquí es donde entra en juego la computación cuántica, permitiendo calcular la transformada discreta de Fourier de forma mucho más eficiente que la computación clásica.

Las dos últimas secciones están dedicadas respectivamente al algoritmo de transformada cuántica de Fourier y al algoritmo de Shor. Finalmente se facilita un enlace electrónico a los cinco apéndices que recogen las demostraciones de los resultados que se utilizan a lo largo del artículo.

Módelo cuántico de computación

El modelo cuántico de computación es una generalización del modelo clásico y, para entenderlo mejor, vamos a introducirlo a partir de éste. Describiremos ambos modelos a bajo nivel, es decir, en términos de las operaciones elementales que el procesador del ordenador clásico (cuántico) es capaz (se espera que sea capaz) de realizar. Empezamos recordando brevemente las características esenciales del modelo clásico.

Módelo clásico de computación

En el modelo clásico de computación la unidad básica de información es el *bit*. Un *bit* puede tener dos valores distintos que se denotan 0 y 1 respectivamente. Desde un punto de vista un poco más formal un *bit* es un elemento del conjunto $V = \{0, 1\}$. Los ordenadores clásicos se construyen a base de circuitos electrónicos en los que se identifica cada *bit* con el estado de carga de un condensador: si está descargado el *bit* vale 0 y si está cargado el *bit* vale 1.

Evidentemente un *bit* tiene poca información. Para representar cantidades mayores de información se utilizan conjuntos de n *bits* que se llaman cadenas de *bits*. Por ejemplo 0110 es una cadena de 4 *bits*. Desde un punto de vista más formal una cadena de n *bits* se puede considerar como un elemento del producto cartesiano $V^n = V \times \cdots \times V$. En una cadena de *bits* se puede representar cualquier tipo de información: basta establecer un criterio que permita codificar y descodificar. Por ejemplo, para codificar un carácter se emplean

8 *bits* (código ASCII) y para codificar un número natural n se utilizan los $\lceil \log_2(n) \rceil$ *bits* de la representación binaria de n.

En el modelo clásico de computación un algoritmo es un mecanismo para manipular cadenas de *bits*. Desde el punto de vista formal se puede considerar como un mecanismo para evaluar funciones booleanas. En efecto, dada una cadena α de n *bits*, el algoritmo la modifica generando otra cadena β de m *bits*. Si llamamos f a la función booleana de $V^n \to V^m$ tal que $f(\alpha) = \beta$ entonces el algoritmo es un mecanismo para evaluar f.

En un algoritmo clásico hay que detallar el mecanismo de evaluación de la función booleana f hasta reducirlo a una secuencia de puertas lógicas (véase la figura 1). Esto se debe a que los ordenadores clásicos sólo son capaces de evaluar puertas lógicas, no son capaces de evaluar funciones booleanas genéricas. Además, sabemos que son suficientes tres puertas lógicas: *not*, *or* y *and* para definir cualquier función booleana. Entonces un algoritmo clásico es una simple secuencia de puertas *not*, *or* y *and*.

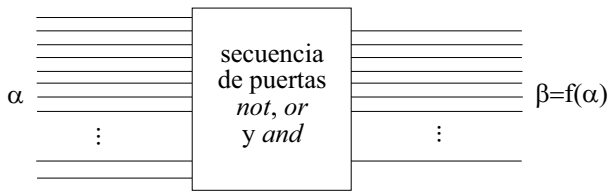

Figura 1: Algoritmo clásico

Si imponemos la condición de que las funciones booleanas que evalúan los algoritmos clásicos sean biyectivas (en particular deberá cumplirse que $m = n$), obtenemos un modelo de computación reversible equivalente al modelo clásico y mucho más parecido que éste al modelo cuántico.

Codificación cuántica de la información

En computación cuántica la unidad elemental de información se denomina *qubit* y, del mismo modo que el *bit* clásico, se define a partir de dos estados básicos que se denotan $|0\rangle$ y $|1\rangle$ respectivamente. Sin embargo, a diferencia de los *bits*, los *qubits* pueden estar en estados que son combinación lineal de los dos estados básicos. Por ejemplo, el estado de un *qubit* puede ser

$$\Psi = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \tag{1}$$

Esto significa que un *qubit* es un vector en el espacio vectorial generado por los dos estados básicos $|0\rangle$ y $|1\rangle$. Concretamente, un *qubit* es un vector unitario en el espacio de Hilbert complejo $\mathcal{H}$ generado por los vectores $|0\rangle$ y $|1\rangle$ y estos determinan una base ortonormal.

En estos momentos el prototipo de ordenador cuántico basado en *trampas de iones* es el más prometedor. En este prototipo los *qubits* se codifican en iones (átomos a los que les falta un electrón) confinados mediante campos eléctricos en pequeñas regiones del espacio denominadas trampas. Los estados básicos $|0\rangle$ y $|1\rangle$ se corresponden con los estados de dos niveles energéticos perfectamente identificados: el de menor energía representa al estado $|0\rangle$ y el de mayor energía al estado $|1\rangle$.

Un *qubit* permite almacenar poca información por lo que es necesario trabajar con un conjunto de n *qubits* (un $n-$*qubit*), del mismo modo que en computación clásica se trabaja con una cadena de n *bits*. Combinando los estados básicos de cada uno de los *qubits* obtenemos los estados básicos del $n-$*qubit* que, extendiendo la notación que hemos visto para los estados básicos de un *qubit*, se representan como

$$|k_1\, k_2\, \ldots\, k_n\rangle \qquad \text{con} \qquad k_1,\, k_2,\, \ldots\, k_n \in \{0,1\} \tag{2}$$

Cada uno de los estados básicos de un $n-$*qubit* se corresponde con una cadena de n *bits*. Pero, del mismo modo que ocurre para un solo *qubit*, los $n-$*qubits* pueden estar en estados que son combinación lineal de los 2^n estados básicos. Por ejemplo, el estado de un $2-$*qubit* puede ser

$$\Psi = \frac{1}{\sqrt{3}}|00\rangle + \frac{1}{\sqrt{3}}|01\rangle + \frac{1}{\sqrt{3}}|10\rangle \tag{3}$$

Concretamente, un $n-$*qubit* es un vector unitario en el espacio de Hilbert complejo $\mathcal{H}_n$ generado por los vectores $|k_1\, k_2\, \ldots\, k_n\rangle$, con $k_1,\, k_2,\, \ldots\, k_n \in \{0,1\}$, que determinan una base ortonormal llamada *base de computación*. Los vectores de la base de computación se pueden interpretar como cadenas de *bits* o como números enteros: $|k_1\, k_2\, \ldots\, k_n\rangle \equiv |k\rangle$, donde k es el número con representación binaria $k_1\, k_2\, \ldots\, k_n$. Con esta notación un $n-$*qubit* es un vector

$$\Psi = \sum_{k=0}^{2^n-1} a_k|k\rangle \qquad \text{tal que} \qquad \sum_{k=0}^{2^n-1} |a_k|^2 = 1 \tag{4}$$

La descripción que hemos hecho de los sistemas de $n-$*qubits* es completa. Sin embargo no sabemos qué relación existe entre el espacio vectorial $\mathcal{H}_n$ y los espacios vectoriales $\mathcal{H}$ asociados a cada uno de los *qubits*, considerados como sistemas independientes. Como cabía esperar, $\mathcal{H}_n$ es el producto tensorial $\mathcal{H} \otimes \cdots \otimes \mathcal{H}$ y los vectores de la base de computación se corresponden con los distintos productos tensoriales de n factores $|0\rangle$ ó $|1\rangle$:

$$|k_1\, k_2\, \ldots\, k_n\rangle = |k_1\rangle \otimes |k_2\rangle \otimes \cdots \otimes |k_n\rangle \tag{5}$$

La relación que existe entre $\mathcal{H}_n$ y $\mathcal{H}$ es muy importante pues, como veremos en la subsección siguiente, las puertas cuánticas actúan localmente sobre uno o sobre dos *qubits*, del mismo modo que en el modelo clásico las puertas lógicas actúan sobre uno o sobre dos *bits*.

Transformación de la información cuántica

Un algoritmo cuántico consiste en evaluar una transformación U que, aplicada al estado inicial de un $n-qubit$ Ψ_{in}, obtiene el estado final $\Psi_{out} = U\Psi_{in}$. Si tenemos en cuenta que U, al igual que la física cuántica, es lineal y que conserva la norma, puesto que transforma vectores unitarios en vectores unitarios, llegamos a la conclusión de que U es una tranformación unitaria. Entonces un algoritmo cuántico consiste en la evaluación de una transformación unitaria en $\mathcal{H}_n$. Además, podemos suponer sin pérdida de generalidad que el estado inicial $\Psi_{in} = |0\rangle$.

Sin embargo, no podemos suponer que un ordenador cuántico sea capaz de evaluar una transformación unitaria arbitraria. Por tanto, es preciso descomponer la transformación unitaria U en producto de transformaciones unitarias elementales que denominamos *puertas cuánticas*. Entonces un algoritmo cuántico es una simple secuencia de puertas cuánticas.

Sea U una puerta cuántica de un *qubit*, es decir, una transformación unitaria en $\mathcal{H}$. Si aplicamos U al primer *qubit* el efecto de dicha transformación es $U|k_1\, k_2 \ldots k_n\rangle = U|k_1\rangle \otimes |k_2 \ldots k_n\rangle$ para todo vector de la base de computación. Formalmente la acción de U sobre $\mathcal{H}_n$ corresponde al producto tensorial $U \otimes I \otimes \cdots \otimes I$. Entre las puertas cuánticas más importantes para la construcción de algoritmos se incluyen las siguientes puertas de un *qubit*, siendo $\sigma_k = e^{\frac{2\pi i}{2^k}}$:

$$Not: \begin{cases} |0\rangle \rightarrow |1\rangle \\ |1\rangle \rightarrow |0\rangle \end{cases} \qquad R_k: \begin{cases} |0\rangle \rightarrow |0\rangle \\ |1\rangle \rightarrow \sigma_k|1\rangle \end{cases} \qquad H: \begin{cases} |0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \tag{6}$$

La puerta Not se puede interpretar como la negación lógica, puesto que intercambia los estados $|0\rangle$ y $|1\rangle$. La puerta R_k introduce el factor σ_k en el estado $|1\rangle$. Finalmente la puerta H, que también se llama transformada de Hadamard, se puede interpretar como un giro de 45 grados.

Sea U una puerta cuántica de dos *qubit*, es decir, una transformación unitaria en $\mathcal{H}_2$. Si, por ejemplo, aplicamos U a los dos primeros *qubits* el efecto de dicha transformación es $U|k_1\, k_2\, k_3 \ldots k_n\rangle = U|k_1\, k_2\rangle \otimes |k_3 \ldots k_n\rangle$ para todo vector de la base de computación. Formalmente la acción de U sobre $\mathcal{H}_n$ corresponde al producto tensorial $U \otimes I \otimes \cdots \otimes I$ que contiene $n-2$ veces la identidad. Entre las puertas cuánticas de dos *qubits* más importantes para

la construcción de algoritmos se incluyen las siguientes:

$$CNot: \begin{cases} |00\rangle \rightarrow |00\rangle \\ |01\rangle \rightarrow |01\rangle \\ |10\rangle \rightarrow |11\rangle \\ |11\rangle \rightarrow |10\rangle \end{cases} \quad CR_k: \begin{cases} |00\rangle \rightarrow |00\rangle \\ |01\rangle \rightarrow |01\rangle \\ |10\rangle \rightarrow |10\rangle \\ |11\rangle \rightarrow \sigma_k|11\rangle \end{cases} \quad Swap: \begin{cases} |00\rangle \rightarrow |00\rangle \\ |01\rangle \rightarrow |10\rangle \\ |10\rangle \rightarrow |01\rangle \\ |11\rangle \rightarrow |11\rangle \end{cases} \tag{7}$$

La puerta $CNot$ (*Controlled Not*) se puede ver como la negación del segundo *qubit* si el primero está en estado $|1\rangle$. Del mismo modo la puerta CR_k (*Controlled* R_k) introduce el factor σ_k en el estado $|1\rangle$ del segundo *qubit*, si el primero está en estado $|1\rangle$. Por último, la puerta $Swap$ intercambia los estados de los dos *qubits*.

No se necesitan puertas cuánticas de más de dos *qubits* para generar cualquier transformación unitaria. En este sentido Barenco y otros [1] probaron que la puerta cuántica de dos *qubits* $Cnot$ y las puertas cuánticas de un *qubit* son suficientes para definir cualquier transformación unitaria en $\mathcal{H}_n$. En definitiva, un algoritmo cuántico es una secuencia de puertas cuánticas de un *qubit* y puertas $CNot$ (véase la figura 2).

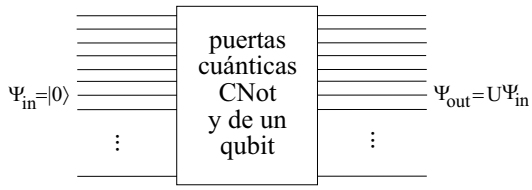

Figura 2: Algoritmo cuántico

Finalmente vamos a introducir la transformada de Walsh-Hadamard que consiste en aplicar la transformada de Hadamard a cada uno de los *qubits*, $W = H \otimes \cdots \otimes H$. Esta transformación unitaria permite, a partir del estado $|0\rangle$, generar un estado mucho más interesante para la computación cuántica:

$$\begin{aligned} \Psi &= (H \otimes \cdots \otimes H)\,(|0\rangle \otimes \cdots \otimes |0\rangle) = (H|0\rangle) \otimes \cdots \otimes (H|0\rangle) \\ &= \tfrac{1}{\sqrt{2^n}}\,(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle \end{aligned} \tag{8}$$

Esta transformada es una de las herramientas más importantes en el diseño de algoritmos cuánticos. La expresión explícita de $W|k\rangle$ para un vector arbitrario $|k\rangle$ de la base de computación, en términos de la forma bilineal simétrica

$(k|j) = (k_1 \odot j_1) \oplus \cdots \oplus (k_n \odot j_n)$ definida en el espacio vectorial $(\mathbb{Z}_2^n, \oplus, \odot_{\mathbb{Z}_2})$, es la siguiente:

$$W|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{(k|j)} |j\rangle \tag{9}$$

Lectura de la información cuántica

Una vez realizado el cálculo es preciso medir el $n-$*qubit* Ψ_{out} para obtener los resultados. En el modelo clásico de computación el proceso de lectura de la información es tan simple que no merece ningún comentario. Sin embargo en el modelo cuántico la situación es mucho más compleja. Se sabe que la medida de dos estados cuánticos idénticos puede dar resultados diferentes. Por tanto un proceso cuántico de medida debe interpretarse como un experimento aleatorio. Además, también se sabe que las medidas cuánticas modifican el estado que se está midiendo.

A pesar de este comportamiento tan extraño, el proceso de medición cuántica tiene una interpretación matemática relativamente simple. Supongamos que tenemos el $n-$*qubit* Ψ_{out} y queremos medir el primer *qubit*, es decir, queremos saber en cuál de los estados $|0\rangle$ ó $|1\rangle$ está dicho *qubit*. Decimos que el resultado de la medida es $k_1 = 0$ si la respuesta es que está en estado $|0\rangle$ y que es $k_1 = 1$ si la respuesta es que está en estado $|1\rangle$. Por tanto de la medida de un *qubit*, en este caso el primero, obtenemos un *bit* de información, k_1.

Hay estados de Ψ_{out} en los que el primer *qubit* está, con toda seguridad, en el estado $|0\rangle$. La descripción de estos estados es sencilla. Son todos los que pertenecen al subespacio $\mathcal{H}_{0,n-1} = |0\rangle \otimes \mathcal{H}_{n-1}$. Análogamente los estados en los que el primer *qubit* está, con toda seguridad, en el estado $|1\rangle$ son los que pertenecen al subespacio $\mathcal{H}_{1,n-1} = |1\rangle \otimes \mathcal{H}_{n-1}$.

Si Ψ_{out} está en $\mathcal{H}_{0,n-1}$ o en $\mathcal{H}_{1,n-1}$ el resultado de la medida del primer *qubit* es determinista: $k_1 = 0$ en el primer caso y $k_1 = 1$ en el segundo. Para estudiar el resultado si Ψ_{out} es un estado cualquiera vamos a tener en cuenta que los subespacios $\mathcal{H}_{0,n-1}$ y $\mathcal{H}_{1,n-1}$ generan una suma directa ortogonal: $\mathcal{H}_n = \mathcal{H}_{0,n-1} \oplus \mathcal{H}_{1,n-1}$. Por tanto siempre podemos expresar Ψ_{out} del siguiente modo:

$$\Psi_{out} = \Phi_0 + \Phi_1 \qquad \text{con} \qquad \Phi_0 \in \mathcal{H}_{0,n-1} \text{ y } \Phi_1 \in \mathcal{H}_{1,n-1} \tag{10}$$

Si Φ_0 y Φ_1 son no nulos, el primer *qubit* no está en un estado $|0\rangle$ ó $|1\rangle$ definido. Por tanto, en este caso la medida será un experimento aleatorio con dos posibles resultados: $k_1 = 0$ ó $k_1 = 1$. La distribución de probabilidad del resultado de la medida del primer *qubit* se describe mediante una regla muy simple:

$$P(k_1 = i) = \|\Phi_i\|^2 \tag{11}$$

Para que la regla anterior sea consistente es imprescindible que los estados se correspodan con vectores unitarios, puesto que la suma de probabilidades $\|\Psi_{out}\|^2 = \|\Phi_0\|^2 + \|\Phi_1\|^2$ debe valer 1. Teniendo en cuenta la propiedad de suma directa ortogonal, resulta que los vectores Φ_0 y Φ_1 son las proyecciones ortogonales de Ψ_{out} sobre los subespacios $\mathcal{H}_{0,n-1}$ y $\mathcal{H}_{1,n-1}$ respectivamente.

Hemos analizado cómo se obtiene el resultado de la medida del primer *qubit*, tanto cuando la respuesta es determinista como cuando es aleatoria. Falta por estudiar el estado $\tilde{\Psi}_{out}$ que queda después de realizar la medida. Para ello nos preguntamos ¿qué ocurre si medimos por segunda vez el estado del primer *qubit*? En este caso el resultado de la segunda medida coincide con toda seguridad con el de la primera. Esto significa que si en la primera medida la respuesta fue k_1 el estado resultante $\tilde{\Psi}_{out}$ tiene el primer *qubit* en estado $|k_1\rangle$, es decir, $\tilde{\Psi}_{out} \in \mathcal{H}_{k_1,n-1}$. Concretamente el estado que resulta es

$$\tilde{\Psi}_{out} = \frac{\Phi_{k_1}}{\|\Phi_{k_1}\|} \tag{12}$$

El experimento aleatorio correspondiente a la medida del primer *qubit* que hemos descrito tiene, desde el punto de vista matemático, una interpretación bastante sencilla. Además, desde el punto de vista de la física cuántica resulta ser el mecanismo más simple que permite modelizar la cuantificación de los observables físicos como, por ejemplo, la energía.

La medida de cualquiera de los *qubits* restantes se describe en los mismos términos que la correspondiente al primer *qubit*. Generalmente, una vez calculado Ψ_{out} mediante un algoritmo cuántico, el resultado se obtiene midiendo, uno a uno, un determinado conjunto de *qubits*. Por ejemplo, si en un $(n+m)-$*qubit* medimos los n primeros *qubits* obtendremos como resultado una cadena de n *bits*, $k_1\, k_2\, \ldots\, k_n$. Esta cadena de *bits* se puede interpretar como la representación binaria de un número natural k, que sería el resultado del algoritmo, y que se obtiene con probabilidad

$$P(k) = \left\| |k\rangle \otimes \sum_{j=0}^{2^m-1} a_{k,j}|j\rangle \right\|^2 = \sum_{j=0}^{2^m-1} |a_{k,j}|^2 \text{ si } \Psi_{out} = \sum_{k=0}^{2^n-1}\sum_{j=0}^{2^m-1} a_{k,j}|k\rangle \otimes |j\rangle \tag{13}$$

Problema de Simon

Una de las primeras cuestiones que se resolvieron en computación cuántica, con ganancia exponencial respecto a los algoritmos clásicos, fue el problema de Simon. En el espacio vectorial $(\mathbb{Z}_2^n, \oplus, \odot_{\mathbb{Z}_2})$ una función booleana $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ es *periódica de periodo* $T \in \mathbb{Z}_2^n$ si para todo $k \in \mathbb{Z}_2^n$ se cumple $f(k \oplus T) = f(k)$ y es *2 a 1* si para todo $k \in \mathbb{Z}_2^n$ se cumple que $f^{-1}(k)$ tiene cardinal 0 ó 2. El problema consiste en determinar el periodo de f con el menor número

posible de evaluaciones de la función. En el modelo cuántico de computación en lugar de la función f se utiliza la transformación unitaria de $2n$ *qubits* U_f definida del siguiente modo:

$$U_f\left(|k\rangle \otimes |j\rangle\right) = |k\rangle \otimes |j \oplus f(k)\rangle \tag{14}$$

Desde el punto de vista clásico hay que evaluar f sobre la mitad más uno de los elementos del dominio, es decir sobre $2^{n-1}+1$ elementos, para estar seguros de encontrar dos elementos k y j tales que $f(k) = f(j)$. A partir de estos valores k y j se puede calcular el periodo de la función: $T = k \oplus j$. Si sólo buscamos una solución probabilística, con cota de error ϵ, habría que evaluar la función $2^{(n-1)/2}\sqrt{1-\epsilon}$ veces. Sin embargo, cuánticamente son suficientes $O((n-1)\log(\epsilon^{-1}))$ evaluaciones. Los detalles de estos dos resultados se pueden ver en el apéndice A.

Algoritmo 1:

1. Inicializar el $2n-$*qubit* $\Psi = |0\rangle \otimes |0\rangle$.
2. Aplicar la transformada de Walsh-Hadamard W a los n primeros *qubits*.
3. Aplicar U_f.
4. Medir los n últimos *qubits*: $j_1, \ldots, j_n$ (resultado $j = j_1 \ldots j_n$).
5. Aplicar de nuevo W a los n primeros *qubits*.
6. Medir los n primeros *qubits*: $k_1, \ldots, k_n$. Devolver $k = k_1 \ldots k_n$.

El resultado final es un número entero k tal que $(T|k) = 0$. Para comprobarlo basta desarrollar el algoritmo paso a paso. La primera medida da como resultado cualquier valor de la imagen de f con probabilidad 2^{-n+1}, por ser una función *2 a 1*, y la segunda cualquier valor k que cumple $(T|k) = 0$

también con probabilidad 2^{-n+1}.

$$
\begin{aligned}
|0\rangle \otimes |0\rangle \quad &\xrightarrow{2} \quad \frac{1}{\sqrt{2^n}} \sum_{0\le k<2^n} (|k\rangle \otimes |0\rangle) \\
&\xrightarrow{3} \quad \frac{1}{\sqrt{2^n}} \sum_{0\le k<2^n} (|k\rangle \otimes |f(k)\rangle) \\
&\xrightarrow{4} \quad \frac{1}{\sqrt{2}} (|l\rangle + |l \oplus T\rangle) \otimes |j\rangle \\
&\qquad \big(\text{medida } j, \text{ por tanto } \{\, l, l \oplus T \,\} = f^{-1}(j)\big) \\
&\xrightarrow{5} \quad \frac{1}{\sqrt{2^{n+1}}} \sum_{0\le k<2^n} \left((-1)^{(l|k)} + (-1)^{(l\oplus T|k)}\right) |k\rangle \otimes |j\rangle \\
&= \quad \frac{1}{\sqrt{2^{n+1}}} \sum_{0\le k<2^n} (-1)^{(l|k)} \left(1 + (-1)^{(T|k)}\right) |k\rangle \otimes |j\rangle \\
&\xrightarrow{6} \quad k \quad \text{tal que} \quad (T|k) = 0
\end{aligned}
$$

Al aplicar el algoritmo evaluamos una vez la transformación unitaria U_f y obtenemos una ecuación lineal homogénea, $(T|k) = 0$, con n incógnitas y coeficientes en el cuerpo $\mathbb{Z}_2$. Debemos repetir el algoritmo hasta obtener un sistema lineal homogéneo de rango $n-1$. La solución no nula de este sistema será el periodo de la función. Aplicando el algoritmo $n-1$ veces la probabilidad de obtener un sistema de rango $n-1$ es mayor que $1/5$, por tanto son suficientes $O((n-1)\log(\epsilon^{-1}))$ repeticiones.

Problema clásico de factorización

La factorización de números enteros es un problema muy actual. No porque hayamos encontrado un método eficiente para resolverlo sino más bien por todo lo contrario. La dificultad de este problema, infructuosamente atacado hasta el momento, nos permite considerar la multiplicación de números enteros como una función de dirección única. La aplicación más importante de este hecho es el sistema criptográfico de clave pública RSA [18], introducido por Rivest, Shamir y Adleman en 1978. La seguridad de este sistema radica en la imposibilidad práctica de factorizar números grandes, con centenares de dígitos.

El algoritmo clásico de factorización con mejor complejidad demostrada es el obtenido por Pollard y Strassen en 1976 que tiene complejidad $O(2^{n/4}n^2)$ [11], siendo n el número de dígitos. Sin embargo existen otros algoritmos más eficientes, aunque no se haya conseguido probar rigurosamente su complejidad.

Por ejemplo, se conjetura que el algoritmo de Lenstra, Lenstra, Manasse y Pollard [17] tiene complejidad

$$e^{O\left(\sqrt[3]{n\log^2(n)}\right)} \tag{15}$$

Para encontrar un factor propio de un número N, impar y con al menos dos factores primos distintos, vamos a trabajar con el grupo multiplicativo de unidades de $\mathbb{Z}_N$, que denotaremos U_N. Se trata de un grupo conmutativo de cardinal $\phi(N)$, donde ϕ es la función de Euler. Un elemento $a \in \mathbb{Z}_N$ es una unidad, es decir pertenece a U_N, si y solo si $\mathrm{mcd}(a, N) = 1$. Con este planteamiento podemos reducir el problema de obtener un factor propio de N al de encontrar el orden de un elemento $a \in U_N$, es decir, al de calcular el menor número natural t tal que $a^t \equiv 1 \bmod N$.

Algoritmo 2:

1. Elegir aleatoriamente a entre 1 y $N-1$, ambos incluidos.
2. Si $\mathrm{mcd}(a, N) \neq 1$ **devolver** $\mathrm{mcd}(a, N)$.
3. Calcular el orden t de a en U_N.
4. Si t es impar **devolver** fallo.
5. Si $\mathrm{mcd}(a^{t/2}+1, N) \neq N$ **devolver** $\mathrm{mcd}(a^{t/2}+1, N)$.
6. **Devolver** fallo.

Si el algoritmo no falla entonces devuelve un factor propio de N. Si termina en el paso 2 el resultado es obviamente un factor propio de N. Si termina en el paso 5 entonces se cumple que a es una unidad de orden par y $\mathrm{mcd}(a^{t/2}+1, N) \neq N$. En este caso obtenemos dos divisores de cero en $\mathbb{Z}_N$: $m_1 = a^{t/2}-1$ y $m_2 = a^{t/2}+1$. En efecto $m_1 \not\equiv 0 \bmod N$ pues en caso contrario $a^{t/2} \equiv 1 \bmod N$ y el orden de a no sería t, $m_2 \not\equiv 0 \bmod N$ ya que estamos suponiendo que $\mathrm{mcd}(m_2, N) \neq N$ y $m_1 m_2 = a^t - 1 \equiv 0 \bmod N$ puesto que t es el orden de a. Finalmente cada uno de los divisores de cero, y en particular m_2, contiene un factor propio de N.

El algoritmo propuesto para reducir el problema es un algoritmo probabilístico. Pero, para que ambos problemas sean polinomialmente equivalentes, debe funcionar con probabilidad mayor que una constante positiva independiente del dato de entrada N. Efectivamente, el algoritmo calcula un factor propio de N con probabilidad mayor que

$$1 - \left(\frac{1}{2}\right)^{h-1} \tag{16}$$

donde h es el número de factores primos distintos que tiene N. La demostración de este resultado se incluye en el apéndice B.

Antes de abordar cuánticamente el cálculo del orden de una unidad $a \in U_N$, vamos a modificar de nuevo el problema. Consideremos la función $f : \mathbb{Z} \to \mathbb{Z}_N$ definida por $f(k) = a^k \bmod N$. Se trata de una función periódica cuyo periodo T coincide con el orden t de a. En efecto:

1. $f(k+t) = a^{k+t} \bmod N = a^k a^t \bmod N = a^k \bmod N = f(k)$, por lo tanto $T \leq t$.

2. $a^T \bmod N = f(T) = f(0) = a^0 \bmod N = 1 \bmod N$, por lo tanto $t \leq T$.

De este modo el problema de encontrar un factor propio de N se ha reducido al de encontrar el periodo de la función $f(k) = a^k \bmod N$ para una unidad $a \in U_N$. Con este propósito Shor introduce en su algoritmo la transformada cuántica de Fourier (QFT). Esta herramienta le permite calcular la transformada discreta de Fourier (DFT) de forma mucho más eficiente que con los algoritmos clásicos conocidos, por ejemplo el de la transformada rápida de Fourier (FFT).

Transformada cuántica de Fourier

La transformada cuántica de Fourier de un *n-qubit* es el operador lineal $F_n : \mathcal{H}_n \to \mathcal{H}_n$ que se define sobre el vector $|j\rangle$ de la base de computación, $0 \leq j < 2^n$, del siguiente modo:

$$F_n|j\rangle = \frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} \sigma_n^{jk} |k\rangle \tag{17}$$

donde σ_n es la raíz 2^n-ésima de la unidad $e^{\frac{2\pi i}{2^n}}$ y $Q = 2^n$. Realmente se trata de la transformada discreta de Fourier cuya expresión habitual $(\hat{f}_0, \cdots, \hat{f}_{Q-1}) = F_n(f_0, \cdots, f_{Q-1})$, en coordenadas de la base de computación, es la siguiente:

$$\hat{f}_k = \frac{1}{\sqrt{Q}} \sum_{j=0}^{Q-1} \sigma_n^{kj} f_j \tag{18}$$

La transformada cuántica de Fourier es una transformación unitaria. Para obtener su inversa basta sustituir en la expresión de F_n el parámetro $\sigma_n = e^{\frac{2\pi i}{2^n}}$ por $\sigma_n^* = e^{-\frac{2\pi i}{2^n}}$. Y actúa sobre el vector $|0\rangle$ del mismo modo que la transformada de Walsh-Hadamard:

$$F_n|0\rangle = \frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} |k\rangle \tag{19}$$

Si denominamos periodo de la función discreta f al menor número T, $1 \leq T \leq Q$, que verifica $f_{j+T} = f_j$ para todo $0 \leq j < Q$, considerando los índices módulo Q, entonces T es un divisor de Q. Si no lo fuera $T' = \mathrm{mcd}(T, Q) < T$ cumpliría $f_{j+T'} = f_j$ para todo $0 \leq j < Q$, contradiciendo la minimalidad de T. En efecto, según el teorema de Bèzout, existirían $m_1, m_2 \in \mathbb{Z}$ tales que $T' = m_1 T + m_2 Q$ y, por tanto, se satisfaría $f_{j+T'} = f_{j+m_1 T+m_2 Q} = f_j$. Entonces, dada una función f de periodo T su transformada cuántica de Fourier $\hat{f}$ se anula en todos los elementos del dominio salvo en los multiplos de la frecuencia w de la función, definida en términos del periodo por la relación $wT = Q$. Es decir

$$F_n \left(\sum_{j=0}^{Q-1} f_j |j\rangle \right) = \sum_{k=0}^{T-1} \hat{f}_{wk} |wk\rangle \tag{20}$$

Esta propiedad permite obtener fácilmente la frecuencia w de la función f y, en consecuencia, también el periodo T. Para ello se aplica la transformada cuántica de Fourier a f y, a continuación, se miden todos los *qubits*. De este modo obtenemos un valor wk tal que $0 \leq k < T$ y devolvemos como resultado $w' = \mathrm{mcd}(wk, Q)$. Si se cumple que el $\mathrm{mcd}(k, T) = 1$ entonces $w' = w$, puesto que $Q = wT$. En caso contrario w' es un múltiplo de w. Si todos los valores de k entre 0 y $T - 1$ son equiprobables entonces se verifica

$$P(w' = w) \log\log(T) = \frac{\phi(T)}{T} \log\log(T) \quad \xrightarrow[\liminf]{} \quad e^{-\gamma} = 0.561459... \tag{21}$$

siendo $\gamma = 0.577215...$ la constante de Euler. En el cálculo del límite inferior anterior se ha tenido en cuenta el siguiente resultado clásico de teoría de números [4]:

$$\liminf_{T \to \infty} \frac{\phi(T) \log\log(T)}{T} = e^{-\gamma} \tag{22}$$

Para conseguir una probabilidad positiva, independiente de T y de Q, habría que repetir el proceso $O(\log\log(Q))$ veces. Por tanto, tendríamos un algoritmo polinomial en el número de dígitos de Q. La transformada cuántica de Fourier se puede definir para valores de Q arbitrarios, en particular para Q igual al número $\phi(N)$. Sin embargo, para evaluarla de forma eficiente, por ejemplo mediante el algoritmo de la transformada rápida, se necesitaría factorizar previamente tanto N como $\phi(N)$.

La elección más simple para implementar la transformada cuántica de Fourier consiste en tomar $Q = 2^n$. En la siguiente figura se muestra el primer algoritmo que se propuso para calcular la transformada cuántica de Fourier en este caso. La demostración de su corrección se incluye en el apéndice C.

Las líneas horizontales representan *qubits* que evolucionan temporalmente de izquierda a derecha y se numeran desde arriba. El símbolo H sobre una línea

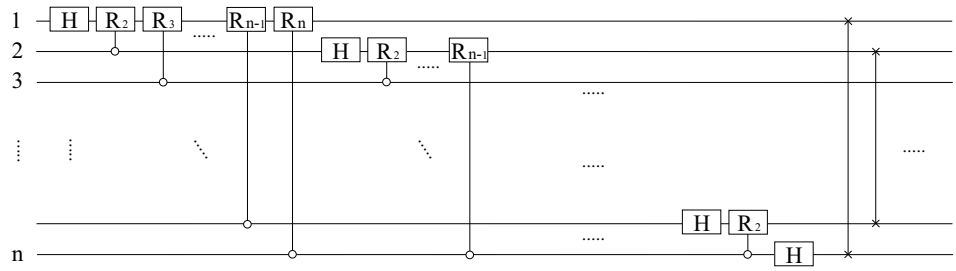

Figura 3: Algoritmo para la transformada cuántica de Fourier

especifica la aplicación de la puerta cuántica H sobre el *qubit* correspondiente a la línea. La aplicación de la puerta CR_k (*Controlled* R_k) se indica uniendo con un segmento vertical los símbolos $\circ$ y R_k que se colocan sobre el *qubit* de control y el *qubit* afectado respectivamente. Finalmente la puerta S (*Swap*) se representa uniendo con un segmento vertical dos símbolos $\times$ colocados sobre los dos *qubits* afectados.

El algoritmo anterior calcula la transformada de Fourier de un vector de longitud $Q = 2^n$ aplicando $O(n^2)$ puertas cuánticas mientras que, clásicamente, el algoritmo de la transformada rápida de Fourier realiza $O(Q\log(Q)) = O(n2^n)$ operaciones. Shor utiliza la ganancia exponencial del algoritmo cuántico para obtener un factor propio del número natural N en tiempo del orden $O(\log^4(N)\text{loglog}(N))$.

Algoritmo de factorización de Shor

El algoritmo de Shor consiste en realizar cuánticamente la parte más costosa del algoritmo clásico descrito anteriormente (algoritmo 2). Realiza cuánticamente el cálculo del orden t de a en U_N (paso 3) o, equivalentemente, el cálculo del periodo T de la función $f(k) = a^k \bmod N$ definida sobre $\mathbb{Z}_{\phi(N)}$.

Ya hemos comentado que si $Q = \phi(N)$ no podemos calcular la transformada cuántica de Fourier. El motivo es evidente: no conocemos el valor de $\phi(N)$ y su cálculo es tan costoso como la factorización de N. Esto significa que no podemos tomar $\mathbb{Z}_{\phi(N)}$ como dominio de la función f. En su lugar consideramos el conjunto $\mathbb{Z}_Q$ donde $Q = 2^n$ es la única potencia de dos que verifica $N^2 < Q < 2N^2$. Generalmente f no será una función periódica en $\mathbb{Z}_Q$, puesto que T no tiene por qué ser un divisor de Q. Por tanto, vamos a trabajar con una extensión no periódica de la función f.

Para representar los valores que toma la función f necesitamos exactamente m *qubits*, siendo m el único número entero tal que $N < 2^m < 2N$, puesto que f toma valores en $\mathbb{Z}_N$. Por tanto, la parte cuántica del algoritmo

de Shor trabajará con un $(n+m)-$*qubit* y con el siguiente operador unitario, asociado a la función f:

$$U_f\left(|k\rangle \otimes |j\rangle\right) = |k\rangle \otimes |j \oplus f(k)\rangle = |k\rangle \otimes |j \oplus (a^k \bmod N)\rangle \tag{23}$$

para todo $0 \leq k < Q$ y $0 \leq j < 2^m$.

En el $(n+m)-$*qubit* se distinguen dos conjuntos de *qubits* o registros. El primero está formado por los n primeros y permite representar todos los elementos del dominio $\mathbb{Z}_Q$ la función. El segundo está constituido por los m últimos y en él se codifican los valores que toma la función en los distintos elementos del dominio. El papel de estos dos registros y del operador U_f se apreciará mejor viendo el funcionamiento del algoritmo.

Algoritmo 3: *(Shor)*

1. Elegir aleatoriamente a entre 1 y $N-1$, ambos incluidos.
2. Si $\mathrm{mcd}(a, N) \neq 1$ **devolver** $\mathrm{mcd}(a, N)$.
3. Calcular el periodo T de la función $f(k) = a^k \bmod N$ definida sobre $\mathbb{Z}_{\phi(N)}$:
   (a) Inicializar el $(n, m)-$*qubit* $\Psi = |0\rangle \otimes |0\rangle$.
   (b) Aplicar F_n al primer registro, es decir aplicar $F_n \otimes I$.
   (c) Aplicar el operador U_f asociado a la función f.
   (d) Aplicar F_n al primer registro, es decir aplicar $F_n \otimes I$.
   (e) Obtener la medida $k \in \{0, 1, \ldots, Q-1\}$ del primer registro.
   (f) Si es posible, calcular T a partir de k y si no **devolver** fallo.
4. Si T es impar **devolver** fallo.
5. Si $\mathrm{mcd}(a^{T/2}+1, N) \neq N$ **devolver** $\mathrm{mcd}(a^{T/2}+1, N)$.
6. **Devolver** fallo.

Para ver el funcionamiento del algoritmo vamos a mostrar, a lo largo del paso 3, la evolución del $(n+m)-$*qubit*. Este seguimiento nos permitirá determinar posteriormente la probabilidad de que el algoritmo encuentre un

factor propio de N.

$$
\begin{aligned}
|0\rangle \otimes |0\rangle \quad &\xrightarrow{(b)} \quad \frac{1}{\sqrt{Q}} \sum_{j=0}^{Q-1} |j\rangle \otimes |0\rangle \\
&\xrightarrow{(c)} \quad \frac{1}{\sqrt{Q}} \sum_{j=0}^{Q-1} |j\rangle \otimes |f(j)\rangle \\
&\xrightarrow{(d)} \quad \frac{1}{Q} \sum_{j=0}^{Q-1} \sum_{k=0}^{Q-1} \sigma_n^{jk} |k\rangle \otimes |f(j)\rangle \;=\; \frac{1}{Q} \sum_{k=0}^{Q-1} |k\rangle \otimes |A(k)\rangle \\
&\qquad\qquad \left(A(k) = \sum_{j=0}^{Q-1} \sigma_n^{jk} |f(j)\rangle \right) \\
&\xrightarrow{(e)} \quad k \in \{\, 0, 1, \ldots, Q-1 \,\} \text{ con probabilidad } P(k) = \tfrac{\|A(k)\|^2}{Q^2}
\end{aligned}
$$

La primera transformada de Fourier sirve exclusivamente para inicializar el $(n+m)-qubit$, poniéndolo como combinación lineal (superposición) de todos los elementos del dominio de f. Si tomásemos como dominio $\mathbb{Z}_{\phi(N)}$, sustituyendo Q por $\phi(N)$ y σ_n por $e^{\frac{2\pi i}{\phi(N)}}$, la probabilidad de obtener k al medir el primer registro verificaría

$$
P(k) = \frac{\|A(k)\|^2}{\phi^2(N)} = \begin{cases} \frac{1}{T} & \text{si } k \text{ es múltiplo de } w \\ \\ 0 & \text{si } k \text{ no es múltiplo de } w \end{cases} \tag{24}
$$

siendo w la frecuencia definida por la expresión $wT = \phi(N)$. Si $Q = 2^n$ la función f no tiene por qué ser periódica en el dominio $\mathbb{Z}_Q$ pues, generalmente, T no será divisor de Q. Si embargo, la transformada de Fourier tiene picos muy pronunciados en los elementos del dominio próximos a los valores jw', $0 \leq j < T$, siendo

$$
w' = \frac{Q}{T} \tag{25}
$$

el valor formal de la frecuencia en la extensión no periódica de f. En la siguiente figura se observa el efecto característico de dicha extensión: reducción de la altura y esanchamiento de los picos de la tranformada de Fourier. A pesar de este efecto, los valores próximos a los multiplos de la frecuencia formal acumulan una parte importante de la probabilidad.

En el apéndice D se demuestra que la probabilidad de que el resultado k de la medida sea el entero más cercano a un múltiplo dado de la frecuencia formal

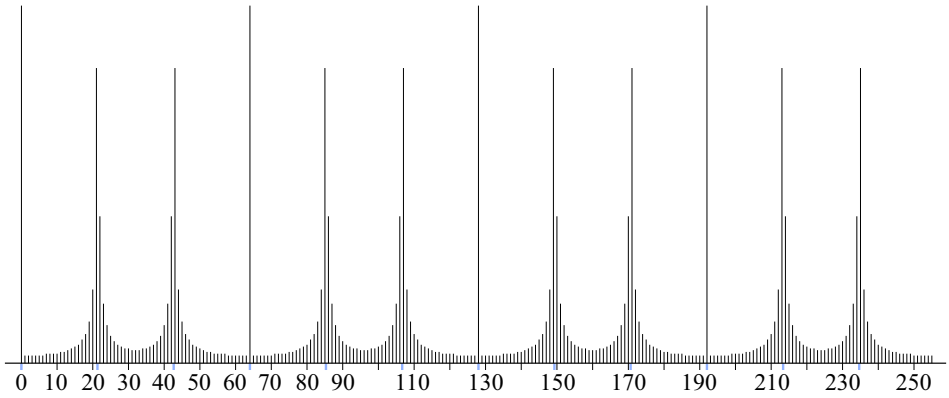

Figura 4: Gráfica de $\sqrt{P(k)}$ para $N = 13$, $a = 2$ ($T = 12$), $n = 8$ ($Q = 256$) y $w' = 21.33$

jw' $(0 \leq j < T)$, es decir que para dicho valor de j se verifique $|k - jw'| \leq 1/2$, satisface

$$P(k)\,T = \frac{\|A(k)\|^2}{Q^2}\,T \geq \frac{4}{\pi^2}\left(1 - \frac{1}{N}\right)^2 \frac{cos^2\left(\frac{\pi}{2N}\right)}{\left(1 + \frac{1}{N}\right)^2} \quad \xrightarrow[N\to\infty]{} \quad \frac{4}{\pi^2} \qquad (26)$$

Entonces, la probabilidad de obtener el entero más cercano a uno cualquiera de los múltiplos de la frecuencia formal tiene una cota inferior que se aproxima, cuando N tiende a infinito, a $4/\pi^2$; una constante independiente de N. Esto significa que el efecto de la extensión no periódica de f es irrelevante. El motivo por el que estamos especialmente interesados en esos valores de k es que permiten obtener el periodo T de la función f con probabilidad alta. En efecto, teniendo en cuenta que $T < N$ y $N^2 < Q$, se concluye que

$$\left|k - jw'\right| \leq 1/2 \implies |Tk - jQ| \leq \frac{T}{2} \implies \left|\frac{k}{Q} - \frac{j}{T}\right| \leq \frac{1}{2Q} < \frac{1}{2T^2} \qquad (27)$$

Finalmente, aplicando un teorema clásico de teoría de fracciones continuas (véase el apéndice E), se concluye que j/T es una de las convergentes de la fracción continua del número k/Q, del que conocemos tanto el numerador como el denominador. Si, además, se cumpliese que $\mathrm{mcd}(j, T) = 1$ obtendríamos el periodo T de la función f sin más que probar con los denominadores de todas las convergentes. Puesto que existe una biyección entre los valores de j y los valores de k y hay $\phi(T)$ valores de j que son primos relativos con T, entonces exiten $\phi(T)$ valores de k que permiten calcular el periodo de la función. Teniendo en cuenta este hecho y los resultados expresados por las

fórmulas 16 y 26, la probabilidad de éxito del algoritmo de Shor, P, verifica:

$$\begin{aligned} P\,\mathrm{loglog}(T) &\geq \left(1-\left(\frac{1}{2}\right)^{h-1}\right)\frac{4}{\pi^2}\frac{\phi(T)}{T}\,\mathrm{loglog}(T)\left(1-\frac{1}{N}\right)^2\frac{cos^2\left(\frac{\pi}{2N}\right)}{\left(1+\frac{1}{N}\right)^2} \\ &\geq \frac{2}{\pi^2}\frac{\phi(T)}{T}\,\mathrm{loglog}(T)\left(1-\frac{1}{N}\right)^2\frac{cos^2\left(\frac{\pi}{2N}\right)}{\left(1+\frac{1}{N}\right)^2} \end{aligned} \tag{28}$$

En la segunda desigualdad hemos utilizado la hipótesis de que el número h de divisores primos de N distintos verifica $h \geq 2$. Tomando límites en la expresión anterior, tanto en N como en T, obtenemos el resultado clave del algoritmo de Shor: la probabilidad de éxito P verifica

$$P\,\mathrm{loglog}(T) \quad \xrightarrow[N\to\infty]{} \quad \frac{2}{\pi^2}\frac{\phi(T)}{T}\,\mathrm{loglog}(T) \quad \xrightarrow{\liminf} \quad \frac{2}{\pi^2}e^{-\gamma} \tag{29}$$

Para conseguir una probabilidad de éxito independiente de N y de T es suficiente repetir el algoritmo $O(\mathrm{loglog}(N))$ veces. Además, veremos un poco más adelante que el número de operaciones que realiza el algoritmo es polinomial respecto del número de dígitos de N. Por tanto, el algoritmo de Shor realiza uno de los sueños más antiguos de las matemáticas: factorizar números enteros de forma eficiente.

A modo de ejemplo, la siguiente figura muestra la probabilidad de éxito del algoritmo de Shor para todos los números compuestos e impares entre 9 y 255. Para calcular estas probabilidades se ha simulando la ejecución del algoritmo de Shor en un ordenador clásico utilizando números complejos en coma flotante con doble precisión. Como hecho curioso se observa que la probabilidad es sensiblemente más pequeña para potencias de números primos, en especial para cuadrados (véase el apéndice B).

Una vez demostrado que la probabilidad de éxito del algoritmo de Shor es razonablemente alta, sólo queda por determinar el número de operaciones que requiere su ejecución. Las complejidades de las operaciones que necesitamos, tanto clásicas como cuánticas, verifican:

1. Aritmética básica para dos números naturales A y B tales que $0 < A, B < 2N^2$:
   (a) Expresión booleana $A = B$ o $A \neq B$: $O(\log(N))$.
   (b) Suma $A + B$ o diferencia $A - B$: $O(\log(N))$.
   (c) Producto $A \cdot B$, cociente A div B o módulo A mod B: $O(\log^2(N))$.
   (d) Máximo común divisor $\mathrm{mcd}(A, B)$: $O(\log^3(N))$.
   (e) Exponenciación modular $A^B \bmod N$: $O(\log^3(N))$.

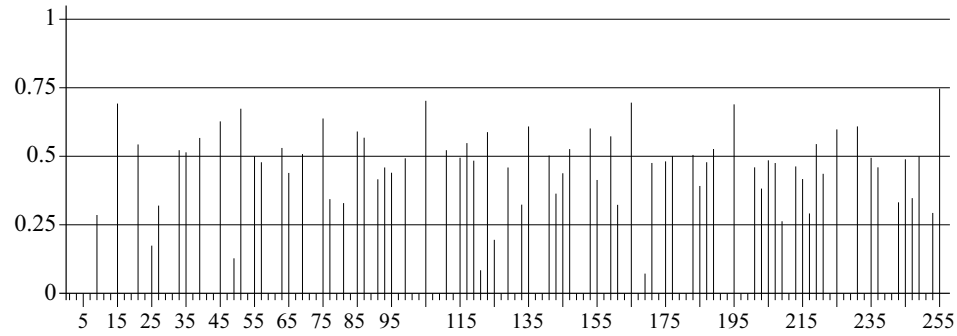

Figura 5: Probabilidad de éxito del algoritmo de Shor

(f) Expresión booleana $A^B = 1 \bmod N$: $O(\log^3(N))$.

2. Elección aleatoria de un número entre 1 y $N-1$: $O(\log(N))$.
3. Determinar si el denominador de alguna convergente de k/Q verifica $a^x = 1 \bmod N$: $O(\log^4(N))$.
4. Tranformada cuántica de Fourier F_n: $O(\log^2(N))$
5. Exponenciación modular cuántica U_f: $O(\log^3(N))$.
6. Medida cuántica del primer registro: $O(\log(N))$.

Entre estos resultados hay dos que requieren una explicación más detallada: puntos 3 y 5. Respecto al primero de ellos, para determinar la complejidad basta observar que el tamaño de la fracción continua de k/Q es $O(\log(N)$ (véase el apéndice E). Además, no es preciso determinar si el periodo T de la función coincide con el denominador de alguna de las convergentes puesto que, en tal caso, deberíamos resolver un problema tan complicado como el problema original de factorizar N. En realidad, basta con verificar si algún denominador cumple la ecuación $a^x = 1 \bmod N$. En cuanto al segundo punto, la exponenciación modular cuántica se puede implementar a partir de algoritmos cuánticos de aritmética entera y modular [22] que, esencialmente, imitan los algoritmos clásicos.

Finalmente, para obtener una probabilidad de éxito próxima a $2e^{-\gamma}/\pi^2 = 0.1137...$, o mayor, habría que repetir el algoritmo $\text{loglog}(N)$ veces. Por tanto, extrictamente hablando, la complejidad del algoritmo de Shor es del orden $O(\log^4(N)\text{loglog}(N))$. Se trata de un algoritmo probabilístico polinomial para factorizar números enteros que, además, se puede adaptar fácilmente para resolver el problema del logaritmo discreto. Estos resultados tienen una gran trascendencia para las matemáticas, especialmente en teoría de números y en criptografía de clave pública.

Apéndices A, B, C, D y E

Se pueden leer en el archivo electrónico
`http://www.eui.upm.es/~jglopez/ApendicesShor.pdf`

Referencias

[1] A. Barenco y otros, Elementary gates for quantum computation, *Physical Review A*, **52**, 5, pp. 3457–3467, (1995).

[2] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines, *J. Stat. Phys.* **29**, 515–546, (1982).

[3] C.H. Bennett, G. Brassard and A.K. Ekert, Quantum cryptography, *Sci. Am.* **267**, 4, 50, (1992).

[4] J. Cilleruelo y A. Córdoba, *La teoría de los números*, Biblioteca Mondadori, 1992.

[5] J.I. Cirac and P. Zoller, Quantum Computation with Cold Trapped Ions, *Phys. Rev. Lett.* **74**, 4091, (1995).

[6] D.G. Cory, M.D. Price and T.F. Havel, Nuclear Magnetic Resonance Spectroscopy: An Experimentally Accesible Paradigm for Quantum Computing, *PhysComp96, New Englang Complex Systems Institute*, **91**, 87, (1996), (`arXiv:quant-ph/9709001`).

[7] A.R. Calderbank, y P.W. Shor, Good Quantum Error-Correcting Codes Exist, *Phys. Rev. A* **54**, 1098–1105, (1996), (`arXiv:quant-ph/9512032`).

[8] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proc. of the Royal Society of London Ser. A* **400**, 97–117, (1985).

[9] P. Domokos, J.M. Raimond and M. Brune, Simple Cavity-QED Two-bit Universal Quantum Logic Gate: The Principle and Expected Performances, *Phys. Rev. A* **52**, 3554, (1995).

[10] R. Feynman, Simulating physics with computers, *International Journal of Theoretical Physics* **21**, 6-7, 467–488, (1982).

[11] J. Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, 1999.

[12] N.A. Gershenfeld and I.L. Chuang, Bulk Spin-Resonance Quantum Computation, *Science* **257**, 350, (1997).

[13] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1997.

[14] L.K. Grover, *A fast quantum mechanical algorithm for database search*, `arXiv:quant-ph/9605043`, (1996).

[15] G.A. Jones and J.M. Jones, *Elementary Number Theory*, Springer, 1998.

[16] B.E. Kane, A Silicon-Based Nuclear Spin Quantum Computer, *Nature* **393**, 133–137, (1998).

[17] A.K. Lenstra, H.W. Lenstra, M.S. Manasse and J.M. Pollard, *The number field sieve*, Proceedings of the Twenty-Second Annual ACM Symposium of the Theory of Computing, Baltimore MD, ACM Press, 564–572, (1990).

[18] R.L. Rivest, A. Shamir and L.M. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM* **21** (2), [550], 120–126, (1978).

[19] P.W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, `arXiv:quant-ph/9508027`, (1994).

[20] P.W. Shor, Scheme for Reducing Decoherence in Quantum Computer Memory, *Phys. Rev. A* **52**, 2493–2496, (1995).

[21] A.M. Steane, Multiple Particle Interference and Quantum Error Correction, `Proc. R. Soc. London A` **452**, 2551–2576, (1996).

[22] V. Vedral, A. Barenco and A. Ekert, *Quantum Networks for Elementary Arithmetic Operations*, `arXiv:quant-ph/9511018`, (1995).

[23] R. Vrijen, E. Yablonovitch, K. Wang, H.W. Jiang, A. Baladin, V. Roychowdhury, T. Mor, D. DiVincenzo, *Electron Spin Resonance Transistors for Quantum Computing in Silicon-Germanium Hetero-Structures*. `arXiv:quant-ph/9905096`, (1996).

[24] W.K. Wootters and W.H. Zurek, A Single quantum cannot be cloned, *Nature* **299**, 802, (1982).

Jesús García López de Lacalle
Dep. de Matemática Aplicada
Escuela Universitaria de Informática
Universitad Politécnica Madrid
Página Web: `http://www.upm.es/~jglopez`
Correo electrónico:`jglopez@eui.upm.es`