UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. DE ELECTRÓNICA Y CIRCUITOS
CIRCUITOS CUÁNTICOS

MINIPROYECTO:

Algoritmo de Shor

Autor:

Ángel Álvarez
16-10031

Julio, 2021

1. Algoritmo Clásico de Shor

Problema

Sea N un número natural producto de dos números primos p y q, el problema a resolver por el algoritmo de Shor radica en encontrar dichos factores de N. Si bien es un problema sencillo de establecer la solución no lo es, incluso muchos sistemas de criptografía clásica se basan en la hipótesis de que no existe forma más rápida de resolver.

En computadores clásicos no toma mucho tiempo resolver para N, pequeños pero a medida que N crece a un orden mayor a los 10^{10}, el tiempo estimado para hallar dicha solución crece a millones de años.

Solución

La mejor solución a este problema es intentar adivinar uno de los factores de N con ensayo y error. Con el algoritmo de Shor podemos mejorar cualquier intento fallido. Veamos:

1. Escojamos un número $a \in [2, N-2]$, donde a se llama base.

2. Calculamos el máximo común divisor entre a y N, esto es
$$\gcd(a, N). \tag{1}$$
Esto se puede hacer rápidamente con el algorimo de Euclides. Ahora:

   - Si $\gcd(a, N) \neq 1$, entonces hemos hallado p o q. Sabiendo alguno de los dos podemos hallar el segundo simplemente haciendo $p = N/q$ o $q = N/p$. Si N es muy grande, esto es muy poco probable que ocurra.
   - Si $\gcd(a, N) = 1$, continuamos al siguiente paso.

3. Ahora, podemos mejorar nuestra base a través de la siguiente ecuación:
$$a^r \equiv 1 \mod N \Rightarrow a^r = k \cdot N + 1, \ \forall k \in \mathbb{N}. \tag{2}$$
Desarrollando la ecuación (2):
$$a^r = k \cdot N + 1 \Rightarrow a^r - 1 = k \cdot N \Rightarrow (a^{r/2} + 1)(a^{r/2} - 1) = k \cdot N.$$
De esta forma, podemos ver que el producto de $(a^{r/2} \pm 1)$ forma un múltiplo de N. Podemos hallar los números primos p y q, ambos vienen dado respectivamente por:
$$\begin{aligned} p &= \gcd(a^{r/2} + 1, N), \\ q &= \gcd(a^{r/2} - 1, N). \end{aligned} \tag{3}$$

Donde r es el mínimo número natural que cumple con la ecuación (2). Sin embargo, r tiene restricciones:

- r debe ser siempre par, si no, la división $r/2$ producidiría un irracional con la base a.
- $\gcd(a^{r/2} \pm 1, N) \neq -1$ y $\gcd(a^{r/2} \pm 1, N) \neq 1$. Si esto ocurre hemos hallado una solución trivial.

Si r no cumple con estas condiciones debemos escoger otra base y empezar de nuevo.

Así, el problema se transforma en hallar un r (periodo) adecuado para nuestra base a, de ser posible.

Veamos un ejemplo;
Sean $N = 35$ y $a = 8$.

1. Calculemos $\gcd(8, 35)$. Con el algoritmo de Euclides nos da que es 1.

2. Realizamos la iteracion para hallar r:

$$\begin{aligned} 8^1 &\mod 35 = 8, \\ 8^2 &\mod 35 = 29, \\ 8^3 &\mod 35 = 22, \\ 8^4 &\mod 35 = 1, \\ 8^5 &\mod 35 = 8, \\ 8^6 &\mod 35 = 29, \\ 8^7 &\mod 35 = 22, \\ 8^8 &\mod 35 = 1. \end{aligned} \tag{4}$$

   De aqui, $r = 4$.

3. Como r es par podemos continuar:

$$\begin{aligned} p &= \gcd(8^{4/2} + 1, 35) = \gcd(8^2 + 1, 35) = \gcd(65, 35) = 5, \\ q &= \gcd(8^{4/2} - 1, 35) = \gcd(8^2 - 1, 35) = \gcd(63, 35) = 7. \end{aligned}$$

   Tambien pudimos haber obtenido $q = N/p$, entonces, $N = 5 \cdot 7$.

2. Algoritmo Cuántico de Shor

Idea general

Como vimos en el algoritmo clásico, puede tomar mucho tiempo evaluar la función $f(x) = a^x \mod N$ para cada x antes de encontrar un candidato al periodo. Por suerte, podemos usar el paralelismo cuántico para evaluar $f(x)$ en todos los x posibles al mismo tiempo.

Nielsen y Chuang definen (2010, pp. 30-32) el **paralelismo cuántico** como una característica fundamental de muchos algoritmos cuánticos. Heurísticamente, y a riesgo de simplificar demasiado, el paralelismo cuántico permite que las computadoras cuánticas evaluen una función $f(x)$ para muchos valores diferentes de x simultáneamente.

En este proceso también usamos ideas del algoritmo de estimación de fase para obtener $\varphi = s/r$, donde r es el orden que buscamos y s es un número entero al azar entre 0 y $r - 1$. En esta parte es donde el resultado que obtengamos puede fallar, basta con empezar de nuevo la computación.

Finalmente, de manera clásica usando la expansión en fracciones continuas podemos hallar fácilmente r.

Algoritmo

1. Empezamos con dos registros cuánticos, de dimensiones en el espacio de Hilbert L y K respectivamente, y lo inicializamos:

$$|\psi_0\rangle = |0\rangle^{\otimes L} |1\rangle . \tag{5}$$

2. Aplicamos la compuerta Hadamard al primer registro:

$$|\psi_1\rangle = (H^{\otimes L} \otimes \mathbb{1}^{\otimes K}) |0\rangle^{\otimes L} |1\rangle = \frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} |x\rangle |1\rangle . \tag{6}$$

3. Luego aplicamos el operador *controlled*-U sobre $|\psi_1\rangle$, para eso, definamos U:

$$U |y\rangle \equiv |ay \mod N\rangle \tag{7}$$

   De aqui, hacer *controlled*-U, con el primer registro de control y el segundo de target es:

$$CU |x\rangle |y\rangle \equiv |(ay)^x \mod N\rangle \tag{8}$$

   Así:

$$|\psi_2\rangle = CU\,|\psi_1\rangle = \frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} |x\rangle\,|a^x \mod N\rangle\,. \tag{9}$$

Análisis de U:

Si vemos $U\,|y\rangle = |ay \mod N\rangle$, entonces $|ay \mod N\rangle$ es un autovector de U con autovalor $\lambda_1 = 1$.

De la misma forma:

$$U^r\,|1\rangle = |a^r \mod N\rangle\,, \tag{10}$$

donde r es el orden que queremos, entonces siempre $a^r \mod N = 1$.

Si consideramos la descomposición espectral y potencias de U:

$$\begin{aligned}
U\,|y\rangle &= \sum_{k=0}^{M} \lambda_k\,|\lambda_k\rangle\,\langle\lambda_k|\,,\\
U^2\,|y\rangle &= \sum_{k=0}^{M} \lambda_k^2\,|\lambda_k\rangle\,\langle\lambda_k|\,,\\
U^3\,|y\rangle &= \sum_{k=0}^{M} \lambda_k^3\,|\lambda_k\rangle\,\langle\lambda_k|\,,\\
&\vdots\\
U^{r-1}\,|y\rangle &= \sum_{k=0}^{M} \lambda_k^{r-1}\,|\lambda_k\rangle\,\langle\lambda_k|\,,\\
U^r\,|y\rangle &= \sum_{k=0}^{M} \lambda_k^r\,|\lambda_k\rangle\,\langle\lambda_k|\,.
\end{aligned} \tag{11}$$

Como $\lambda_k = 1$ para todo $0 \leq k \leq r$ con $k \in \mathbb{N}$, entonces U comparte los mismos autovectores y autovalores con sus potencias. De esta forma podemos contruir un autovector de U que sea la superposición normalizada de cada uno de ellos:

$$|\lambda_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \lambda_k^k\,\big|(ay)^k \mod N\big\rangle \tag{12}$$

Ahora que el autovalor sea 1 no nos aporta ninguna información, el siguiente paso del algoritmo es aplicar la $QFT^\dagger$, seria más útil ver el estado $|\psi_2\rangle$ como el reultado de una QFT para ver con mayor claridad que obtendremos a la salida. Para eso, hagamos λ_k variante, definimos:

$$\lambda_k \equiv e^{-\frac{2\pi i s}{r}}. \tag{13}$$

Así:

$$|\lambda_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} \left|(ay)^k \mod N\right\rangle . \tag{14}$$

Veamos que pasa si sumamos sobre todos los $|\lambda_s\rangle$ en s, ponderados cada uno por una normalización y una fase variante, es decir:

$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s l}{r}} |\lambda_s\rangle &= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s l}{r}} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} \left|(ay)^k \mod N\right\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} \sum_{s=0}^{r-1} e^{\frac{2\pi i s (l-k)}{r}} \left|(ay)^k \mod N\right\rangle . \end{aligned} \tag{15}$$

Por un lado:

$$\sum_{s=0}^{r-1} e^{\frac{2\pi i s k'}{r}} = \begin{cases} \frac{1-e^{2\pi i k'}}{1-e^{\frac{2\pi k'}{r}}} = \frac{1-1}{1-e^{\frac{2\pi k'}{r}}} = 0 & , \ k' \neq 0 \\ \sum_{s=0}^{r-1} e^{\frac{2\pi i s 0}{r}} = \sum_{s=0}^{r-1} 1^s = r - 1 + 1 = r & , \ k' = 0. \end{cases} \tag{16}$$

Entonces

$$\sum_{s=0}^{r-1} e^{\frac{2\pi i s k'}{r}} = r \delta_{k',0}. \tag{17}$$

Retomando, hacemos $k' = l - k$, entonces $\delta_{k',0} = \delta_{l-k,0} = \delta_{l,k}$

$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s l}{r}} |\lambda_s\rangle &= \frac{1}{r} \sum_{k=0}^{r-1} \sum_{s=0}^{r-1} e^{\frac{2\pi i s k'}{r}} \left|(ay)^k \mod N\right\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} r \delta_{l,k} \left|(ay)^k \mod N\right\rangle \\ &= \frac{1}{r} r \left|(ay)^l \mod N\right\rangle \\ &= \left|(ay)^l \mod N\right\rangle . \end{aligned} \tag{18}$$

Finalmente:

$$\left|(ay)^x \mod N\right\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s x}{r}} |\lambda_s\rangle . \tag{19}$$

Sustituyendo (19) en (9) nos queda:

$$
\begin{aligned}
|\psi_2\rangle = CU\,|\psi_1\rangle &= \frac{1}{\sqrt{2^L}}\sum_{x=0}^{2^L-1}|x\rangle\,|a^x \mod N\rangle \\
&= \frac{1}{\sqrt{2^L}}\sum_{x=0}^{2^L-1}|x\rangle\,\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}e^{\frac{2\pi i s x}{r}}\,|\lambda_s\rangle \\
&= \frac{1}{\sqrt{2^L}}\sum_{x=0}^{2^L-1}e^{\frac{2\pi i s x}{r}}\,|x\rangle\otimes\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\lambda_s\rangle\,.
\end{aligned}
\tag{20}
$$

4. Aplicamos $QFT^\dagger$ al primer registro:

$$
\begin{aligned}
|\psi_3\rangle &= (QFT_L^\dagger\otimes\mathbb{1}^{\otimes K})\,|\psi_2\rangle \\
&= (QFT_L^\dagger\otimes\mathbb{1}^{\otimes K})\left(\frac{1}{\sqrt{2^L}}\sum_{x=0}^{2^L-1}e^{\frac{2\pi i s x}{r}}\,|x\rangle\otimes\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\lambda_s\rangle\right)
\end{aligned}
\tag{21}
$$

   Si hacemos

$$
\frac{s}{r} = \frac{1}{2^L}\frac{\tilde{s}}{\tilde{r}}.
$$

   Entonces

$$
\frac{1}{\sqrt{2^L}}\sum_{x=0}^{2^L-1}e^{\frac{2\pi i\tilde{s}}{2^L\tilde{r}}}\,|x\rangle \xrightarrow{QFT_L^\dagger} \frac{1}{2^L}\sum_{x=0}^{2^L-1}\sum_{z=0}^{2^L-1}e^{-\frac{2\pi i x}{2^L r}}\left(z-\frac{\tilde{s}}{\tilde{r}}\right)|z\rangle\,. \tag{22}
$$

   Finalmente

$$
|\psi_3\rangle = \frac{1}{2^L}\sum_{x=0}^{2^L-1}\sum_{z=0}^{2^L-1}e^{-\frac{2\pi i x}{2^L r}}\left(z-\frac{\tilde{s}}{\tilde{r}}\right)|z\rangle\otimes\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\lambda_s\rangle \tag{23}
$$

5. Midiendo el primer registro en las bases computacionales, la probabilidad alcanza su pico cuando z esta cerca de $\tilde{s}/\tilde{r}$.

   Donde $\tilde{s}/\tilde{r}$ es una aproximación de $2L-1$ bits del real s/r. Utilizando el algoritmo de expansión en fracciones continuas podemos hallar r con facilidad, si este r no cumple con las mismas condiciones de la parte clásica debemos escoger otra base y empezar de nuevo.

Finalmente el circuito cuántico encargado de realizar el algoritmo de Shor es:

3. Diseño de los circuitos

En nuestro caso, se nos pide factorizar los numeros $N = \{15, 35, 55\}$ con el algoritmo cuántico de Shor y además ver el deempeño de 3 computadores cuánticos diferentes de IBM (IBMQ).

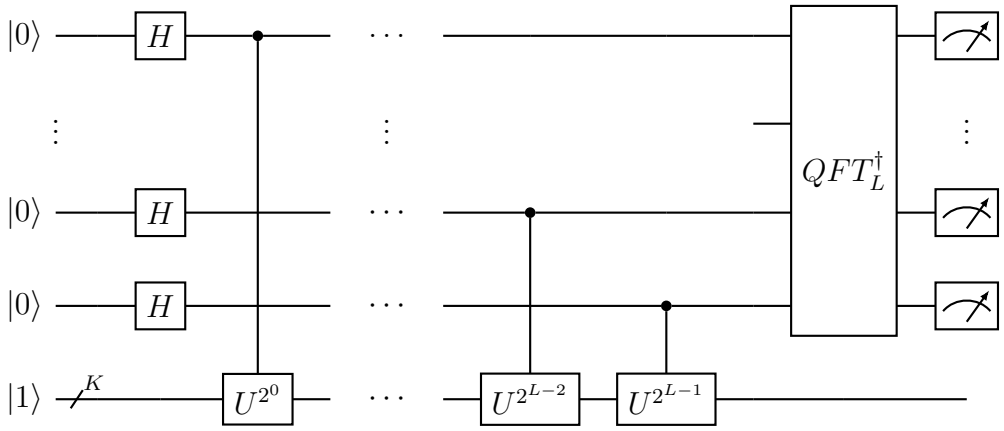

Figura 1: Circuito para el algoritmo cuántico de Shor.

Considerando que la magnitud de L y K suele ser bastante grande incluso para N pequeños, en el orden de 10 $(L + K)$. Lamentablemente tenemos que emular en computadores cuánticos de máximo 5 qubits.

Para reducir el tamaño de los qubits necesarios en los circuitos, hay que tomar ciertas consideraciones, que por suerte esto será suficiente para demostrar que el algoritmo cuántico de Shor funciona.

Lo que determinará en esta implementación L y K será el diseño de la compuerta U. Hay que ver además el valor de r de antemano. En estos 3 casos se consideran $r = \{2, 4\}$. Aun cuando r sea el mismo se intentara diseñar una U diferente para ver cómo esto afecta en la emulación.

- U para $N = 15$:

  Sea $U|y\rangle = |11y \mod 16\rangle$, entonces

$$\begin{aligned} U|1\rangle &= |11 \mod 15\rangle = |11\rangle\,, \\ U^2|1\rangle &= \left|11^2 \mod 15\right\rangle = |1\rangle\,. \end{aligned} \tag{24}$$

  De aqui, $r = 2$, entonces U solo trabaja con 2 estados diferentes $|1\rangle$ y $|11\rangle$, entonces si definimos estos estados en las bases computacionales:

$$\begin{aligned} |11\rangle &\equiv |0\rangle\,, \\ |1\rangle &\equiv |1\rangle\,. \end{aligned} \tag{25}$$

  y sutituimos esto en (24), tenemos:

$$\begin{aligned} U|1\rangle &= |0\rangle\,, \\ U^2|1\rangle &= U|0\rangle = |1\rangle\,. \end{aligned} \tag{26}$$

Entonces $U = X$. Además, $U^2 = XX = \mathbb{1}$, $U^4 = XXXX = \mathbb{1}$ y $U^8 = \mathbb{1}$. De esta simplificacion podemos reducir aun más el circuito a $L = 1$.

Así, $K = 1$ y $L = 4$. De aquí, nuestro circuito para factorizar $N = 15$ con base $a = 11$ es:

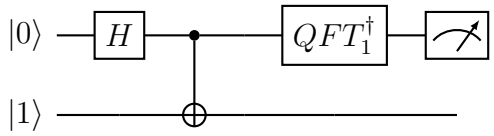

Figura 2: Circuito para $N = 15$ y $a = 11$.

- U para $N = 35$

  Sea $U|y\rangle = U|13\rangle$, entonces

$$\begin{aligned} U|1\rangle &= |13 \mod 35\rangle = |13\rangle\,, \\ U^2|1\rangle &= \left|13^2 \mod 35\right\rangle = |29\rangle\,, \\ U^3|1\rangle &= \left|13^3 \mod 35\right\rangle = |27\rangle\,, \\ U^4|1\rangle &= \left|13^4 \mod 35\right\rangle = |1\rangle\,. \end{aligned} \tag{27}$$

  De aquí $r = 4$, entonces U trabaja con 4 estado diferentes, si definimos estos estados en las bases computacionales:

$$\begin{aligned} |1\rangle &\equiv |01\rangle\,, \\ |13\rangle &\equiv |10\rangle\,, \\ |29\rangle &\equiv |11\rangle\,, \\ |27\rangle &\equiv |00\rangle\,. \end{aligned} \tag{28}$$

  Sustituyendo (28) en (27) tenemos:

$$\begin{aligned} U|01\rangle &= |10\rangle\,, \\ U|10\rangle &= |11\rangle\,, \\ U|11\rangle &= |00\rangle\,, \\ U|00\rangle &= |01\rangle\,. \end{aligned} \tag{29}$$

  Si bien existen muchos circuitos capaces de realizar esta transformación, utilizaremos el siguiente:

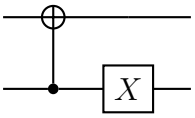

Figura 3: Diseño de U.

De aqui, U^2 es:

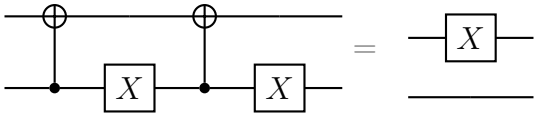

Figura 4: Simplificación de U^2

Facilmente, podemos ver que $U^4 = \mathbb{1} \otimes \mathbb{1}$. Asi $K = 2$ y con la simplificación anterior no necesitamos el 3 qubit de control, asi $L = 2$

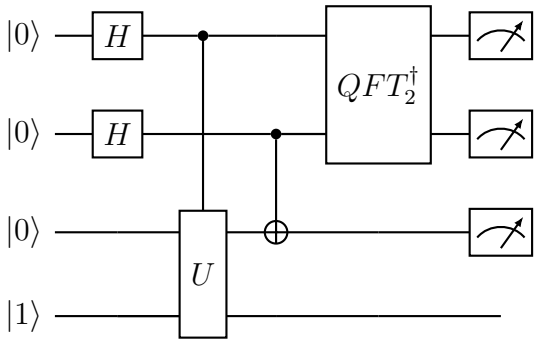

Figura 5: Circuito para $N = 35$ y $a = 12$.

- U para $N = 55$

  Si escogemos $a = 12$ con un calculo similar al anterior obtenemos $r = 4$ y de nuevo pudieramos usar el circuito anterior sin ningun problema. Sin embargo, en esta oprtunidad diseñaremos U de forma diferente.

  Si bien el circuito anterior pudimos simplificar las potencias de U ese no siempre es el caso, y evaluar dichas potencias puede tomar bastante tiempo. Siguiendo esto, podemos diseñar una U que realice la misma acción usando lógica circuital clásica. Veamos la siguiente tabla:

x	$f(x) = 12^x \mod 55$
0	1
1	12
2	34
3	23
4	1
5	12
6	34
7	23

En binario seria:

$x\,(x_0x_1x_2)$	$f(x)\,(f_0f_1f_2f_3f_4f_5)$	f_4f_5
000	000001	01
001	001100	00
010	100010	10
011	010111	11
100	000001	01
101	001100	00
110	100010	10
111	010111	11

De esta tabla, utilizando algebra booleana se concluye que las expresiones para f_4 y f_5 son:

$$\begin{aligned} f_4 &= x_1, \\ f_5 &= \bar{x_1}\bar{x_2} + x_1x_2. \end{aligned} \tag{30}$$

De estas expresiones, podemos prescindir del primer qubit entonces porque no lo utilizamos. Con ayuda la compuerta Toffoli podemos crear el siguiente circuito para U:

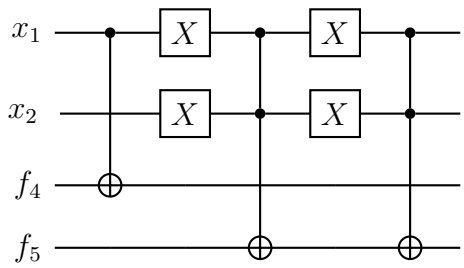

Figura 6: Diseño de U para $N = 55$ y $a = 13$.

Finalmente, $L = K = 2$. Nuestro circuito para $N = 55$ y $a = 12$ es:

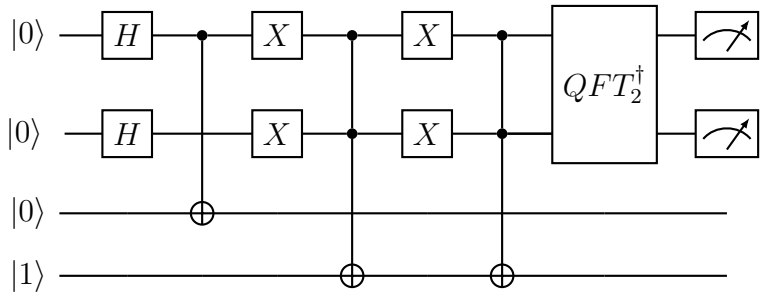

Figura 7: Circuito para $N = 55$ y $a = 13$.

4. Análsis de resultados.

Retomando los resultados de las simulaciones y emulaciones podemos construir la siguiente tabla:

IBMQ	N					
	15		35		55	
	Prob.	σ	Prob.	σ	Prob.	σ
Simulación	0,501	0,000	0,257	0,000	0,249	0,000
Santiago	0,479	0,015	0,267	0,005	0,227	0,020
Quito	0,462	0,026	0,209	0,015	0,233	0,008
Bogotá	0,444	0,035	0,258	0,010	0,228	0,016

Figura 8: Tabla de resultados de simulación y emulación.

Seguido de esta información podemos hacer una tabla de los errores relativos entre cada IBMQ y el valor real. El valor real para $N = 15$ es 0.5 y para $N = 35$ y $N = 55$ es 0.25, así:

IBMQ	N		
	15	35	55
	Error	Error	Error
Santiago	4%	7%	9%
Quito	8%	16%	7%
Bogotá	11%	16%	9%

Figura 9: Tabla de errores relativos.

También podemos hacer una tabla de los tiempos de ejecución más rápidos:

IMBQ	N		
	15	35	55
	Tiempo (s)	Tiempo (s)	Tiempo (s)
Santiago (2)	26,217	27,452	27,766
Quito (1)	23,164	24,492	24,785
Bogotá (3)	26,205	27,621	27,748
Promedio	25,195	26,522	26,766

Figura 10: Tabla de tiempos de ejecución.

De esta información podemos concluir lo siguiente:

- El computador que se desempeña mejor entre estos tres es el de IBMQ antiago, si bien no es el más rápido en tiempos de ejecución es el que menos errores relativos presenta de alrededor de 7 % y es el segundo más rápido entre el grupo, por un poco más de 3 s.
- El computador más rápido es el de IBMQ Quito por una diferencia media de 3 s con respecto al segundo lugar, y siempre por debajo del promedio, pero lamentablmente tiene errores tiene alrededor del 10 %.
- En general, el computador que peor se desempeño fue el de IBMQ Bogotá, si bien no es el que más errores tiene es el más lento de los tres y sus errores son alrededor del 12 %.
- En general, no hubo la presencia de muchos ruidos en las emulaciones, esto se puede deber a las simplificaciones que se le hicieron a los circuitos para representar a U y sus potencias.
- El circuito con más errores relativos fue el de $N = 35$, con alrededor del 13 %, el de menos errores fue el de $N = 15$ alrededor del 8 %.
- Para el cambio de la compuerta U en $N = 55$ el tiempo de ejecución incremento en todo el grupo alrededor de 3 decimas de segundo. Sin embargo, los errores relativos en dicha ejecución disminuyeron a casi la mitad en IBMQ Quito y IBMQ Bogotá, pero aumento un 2 % en IBMQ Santiago.

5. Referencias

1. Nielsen, M. y Chuang, I. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press

2. Nakahara, M. y Ohmi, T. (2008). *Quantum Computing*. CRC Press.

3. Monz, T. (2021). Realization of a scalable Shor algorithm. *Science Magazine*. 1068-1070. https://science.sciencemag.org/content/351/6277/1068

4. Cheung, D. (2003). *Using Generalized Quantum Fourier Transforms in Quantum Phase Estimation Algorithms*. University of Waterloo.