UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio de la aplicación de algoritmos cuánticos de optimización con las tecnologías cuánticas actuales

Autor: Oriol Julián Posada
Tutor: Francisco Gómez Arribas

marzo 2024

Oriol Julián Posada
Estudio de la aplicación de algoritmos cuánticos de optimización con las tecnologías cuánticas actuales

Oriol Julián Posada
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Resumen

***** Rehacer el resumen **********

En un mundo cada vez más conectado, con el crecimiento exponencial del volumen de datos en la red, los ataques en internet, como los ataques de denegación del servicio, son cada vez más frecuentes.

Actualmente es de interés el desarrollo de herramientas de seguridad modulares que, por una parte, permitan añadir fácilmente aportaciones de la comunidad de desarrolladores y, por otra, sean sencillas de utilizar por parte del usuario.

La aportación de este trabajo de fin de grado cosiste en el diseño y desarrollo de un cortafuegos, que permitirá, por un lado, que los programadores añadan sus propias soluciones de forma muy sencilla y, por otro lado, a los usuarios gestionar el cortafuegos fácilmente.

En primer lugar, se analizarán las soluciones y sistemas de cortafuegos existentes en el mercado con el fin de detectar sus fortalezas y debilidades. El propósito es diseñar una aplicación que supla estas carencias y sea verdaderamente útil para el usuario.

Una vez realizado el estudio, se procederá a diseñar la aplicación utilizando el mejor sistema para filtrar paquetes. Entre las distintas posibilidades disponibles actualmente, XDP es una alternativa muy interesante puesto que permite que se ejecute código en el controlador de tarjeta de red. La ventaja de este sistema es poder eliminar paquetes a alta velocidad y con menos recursos, algo muy importante cuando se trata de detener un ataque. La principal desventaja es la complejidad del desarrollo y comunicación debido a la necesidad de utilizar lenguajes de bajo nivel en el funcionamiento interno y de alto nivel en la interfaz.

Por último, se realizarán pruebas de rendimiento y de validación con el fin de comprobar que todos los requisitos se satisfacen y el cortafuegos es viable. Esto es muy importante, ya que es una aplicación corriendo en una zona crítica del sistema.

Palabras clave

Cortafuegos, metodologías ágiles, filtrado de paquetes, XDP, eBPF, Flask Python, ataques de internet, API REST, desarrollo web

Abstract

Nowadays, as the volume of data on the internet increases exponentially, cyber attacks such as distributed denial-of-service, are becoming increasingly common.

The development of modular security tools that, on the one hand, allow easy addition of contributions from the developer community and, on the other hand, are user-friendly is currently of interest.

The contribution of this end-of-degree project consists in the design and development of a firewall, which will allow, on the one hand, programmers to add their own solutions in a very simple way and, on the other hand, users to manage the firewall with ease.

First, the existing firewall solutions and systems on the market will be analysed in order to identify their strengths and weaknesses. The purpose is to design an application that will address these shortcomings and be truly useful to the user.

Once the study is completed, the application will be designed using the best system for filtering packages. Among the various possibilities currently available, XDP is a very interesting alternative since it allows the code to be executed using the network card driver. The advantage of this system is its ability to eliminate packages at high speed and with less resources, which is very important when it comes to stopping an attack. The main disadvantage is the complexity of the development and communication due to the need to use low level languages in the internal functioning and high level languages in the interface.

Finally, performance and validation tests will be carried out in order to verify that all requirements are met and the firewall is viable. This is very important, since it is an application running in a critical area of the system.

Keywords

Firewall, agile methodologies, packet filtering, XDP, eBPF, Flask Python, cyber attacks, API REST, web development

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Introducción

En los últimos años, se encuentra en el centro de la discusión el avance que supone la computación cuántica. Al respecto se suele mencionar la revolución en el campo de la encriptación que supondría el algoritmo de Shor [1], que impediría utilizar algoritmos que se basen en que la factorización en primos de un entero es NP. No obstante, este no es el único caso en el que la computación cuántica podría significar una diferencia con respecto a la computación clásica, ya que en el campo de la optimización combinatoria de problemas también se proponen varios algoritmos alternativos que permitirían una mejora en la complejidad.

Esta superioridad, es todavía teórica, ya que en la era NISQ (Noisy Intermediate-Scale Quantum era [2]) no es posible el uso de procesadores cuánticos con muchos qubits y poco ruido en sus ejecuciones. Esto provoca la aparición de algoritmos híbridos, como QAOA [3] o VQE, que combinan la ejecución de circuitos cuánticos pequeños con el pre y postprocesamiento en un ordenador clásico.

Esto provoca la aparición de algoritmos híbridos, en los que un procesador cuántico genera un estado cuántico a través de un circuito parametrizado según un conjunto de variables obtenidas de un procesador clásico.

Dentro de los llamados algoritmos híbridos

Tanto el algoritmo QAOA como el algoritmo de quantum annealing tienen como utilidad resolver problemas tipo QUBO (Quadratic Unconstrained Binary Optimization), en los que se busca un extremo (máximo o mínimo global) de una función binaria. La forma en la que ambos realizan esta tarea consiste en conseguir que el estado fundamental del hamiltoniano que describe el sistema cuántico sea el resultado de la función de coste clásica.

Aunque sirvan para el mismo propósito, las técnicas que se utilizan son completamente diferentes.

- El algoritmo QAOA está pensado para ser aplicado en computadores cuánticos de uso generalista basados en puertas, los cuales son el proyecto más ambicioso a día de hoy dentro del campo de la computación cuántica, que servirían para resolver cualquier tipo de problema al ser sistemas Turing completos.

- Los computadores en los que se aplica quantum annealing, como es el caso de los proporcionados por D-Wave , no son Turing-completos y tienen como única utilidad resolver problemas utilizando este algoritmo. Este es el motivo por el que parece haber una superioridad tan grande entre D-Wave, que comercializa sistemas con más de 5000 qubits, y los ssistemas de uso generalista, que no alcanzan los 1000 qubits manteniendo un funcionamiento tolerante a fallos [1].

[1] Es importante recalcar que el auténtico problema de estos computadores no es aumentar el número de qubits, sino aumentarlo sin incrementar el ruido del sistema.

2

Estado del arte

2.1. QAOA

2.1.1. Computación adiabática cuántica

La computación cuántica adiabática (**AQC**) es, en contraposición con el enfoque en los circuitos, un paradigma basado en el teorema adiabático.

Sea un sistema cuántico dado por un hamiltoniano dependiente del tiempo $H(t) = (1 - \frac{t}{T})H_s + \frac{t}{T}H_c$, siendo H_s un hamiltoniano con un estado fundamental conocido, H_c un hamiltoniano cuyo estado fundamental se quiere conocer y T el tiempo total.

En este caso el teorema adiabático afirma que si el estado inicial es el estado fundamental de H_s y se avanza el tiempo lentamente el sistema se mantendrá en el estado fundamental de $H(\alpha)$. Esto significa que al llegar a $t = T$ el sistema se encontrará en el estado fundamental de H_c.

El paradigma seguido por los ordenadores de D-Wave, Quantum Annealing, es una implementación de AQC.

3 Diseño

3.1. Problemas de optimización combinatoria

Un problema de optimización combinatoria se define con la siguiente función de coste:

$$f(x) = \sum_{\alpha=1}^{m} f_\alpha(x)$$

dde $x = x_1 x_2 ... x_n$ y $x_i \in \{0, 1\}$

$$f_\alpha(x) = \begin{cases} 1 & \text{si x satisface } f_\alpha \\ 0 & \text{en otro caso} \end{cases}$$

En este caso el objetivo será encontrar el **mínimo global** de esta función. Para esto mismo se debe encontrar una cadena de bits de tamaño *n* que cumplan para la mayor cantidad de cláusulas $f_\alpha(x) = 0$.

Para que sean aplicados a algoritmos cuánticos estos problemas no pueden tener restricciones añadidas, por lo que el espacio de resultados posibles será de 2^n combinaciones.

3.1.1. Añadir restricciones

Si el problema que se está intentando representar tiene restricciones de la forma $A(x) = B(x)$ la forma de añadirlas a la función de coste sería $f'(x) = f(x) + P * (A(x) - B(x))^2$, donde

$$P * (A(x) - B(x))^2 \begin{cases} = 0 & \text{si se cumple la restricción} \\ \geq P & \text{en otro caso} \end{cases}$$

El parámetro P se denomina *modificador de Lagrange* y tiene un valor lo suficientemente grande como para que el castigo en caso de romper una restricción aumente lo suficiente el valor de la función de coste como para que sea mayor que cualquier otro resultado en el que no se rompa. Esto sería: $P > \text{máx}_x f(x)$

3.2. Circuito de QAOA

El sistema cuántico en el algoritmo se desarrolla sobre un espacio de Hilbert de 2^n dimensiones, donde *n* es el número de bits de entrada en la función de coste clásica. Esto quiere decir que se tendrán tantos qubits como bits tenga la entrada de *f(x)*.

La base computacional se representa como $\{|x\rangle : x \in \{0,1\}^n\}$.

La idea general de QAOA se basa en preparar un estado $|\psi(\vec{\beta},\vec{\gamma})\rangle$ tal que, con los valores adecuados $(\vec{\beta_{opt}},\vec{\gamma_{opt}})$, el estado $|\psi(\vec{\beta_{opt}},\vec{\gamma_{opt}})\rangle$ encuentre la solución al problema. Los parámetros de dicho estado son:

$$\vec{\beta} = [\beta_0, \beta_1, ..., \beta_{p-1}]$$
$$\vec{\gamma} = [\gamma_0, \gamma_1, ..., \gamma_{p-1}]$$

Donde *p* es el número de capas del circuito y $\beta_i, \gamma_i \in [0, 2\pi]$.

Este estado consta de tres componentes: el **estado inicial** ($|\psi_0\rangle$); y dos operadores denominados **problem hamiltonian** ($U(C,\gamma)$) y **mixing hamiltonian** ($U(B,\beta)$). Estos se combinan de la siguiente forma:

$$|\psi(\vec{\beta},\vec{\gamma})\rangle = U(B,\beta_{p-1})U(C,\gamma_{p-1})U(B,\beta_{p-2})U(C,\gamma_{p-2})...U(B,\beta_0)U(C,\gamma_0)\,|\psi_0\rangle$$

3.2.1. Estado inicial

El estado inicial del qubit se define como

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle = (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))^{\otimes n} = H^{\otimes n}\,|0\rangle^{\otimes n}$$

Este estado inicial se construye añadiendo operadores de Hadamard a *n* qubits inicializados a $|0\rangle$, lo que genera un estado equiprobable, es decir, donde la probabilidad de obtener una cadena dada de *n* bits al medir el estado sería en cualquier caso $\frac{1}{2^n}$.

3.2.2. Problem hamiltonian y mixing hamiltonian

Los operadores se definen como:

$$U(B,\beta) = e^{-i\beta B}$$
$$U(C,\gamma) = e^{-i\gamma C}$$

La operación $U(X) = e^{-iX}$ sirve para garantizar la unitariedad del operador ya que, como se verá en su definición, *B* y *C* no son necesariamente unitarios [1].

Tanto la noción de hamiltoniano como la construcción de un operador unitario proceden de la ecuación de Schrödinger de la mecánica cuántica.

Operador *B*

B se construye a partir de puertas Pauli-X (σ^x) y se define de la siguiente forma:

$$B = \sum_{i=0}^{n-1} \sigma_i^x$$

$$U(B,\beta) = e^{-i\beta B} = \prod_{j=0}^{n-1} e^{-i\beta\sigma_j^x} = \prod_{j=0}^{n-1} Rx_i(2 * \beta)$$

Como se puede ver únicamente depende de la cantidad de qubits del sistema.

Operador *C*

C debe ser un operador diagonal en la base computacional $|x\rangle$ con los valores de la función objetivo. De esta forma, siendo $|x\rangle$ un vector en la base computacional se cumple que $\forall\, |x\rangle = [x_1 x_2 ... x_n], \langle x| \, C \, |x\rangle = f(x)$. Se define a partir de la función de coste clásica del problema a resolver y se construye utilizando puertas Pauli-Z (σ^z).

Para obtener este operador se parte de una función de la forma $f(x)$, donde $x = x_1 x_2 ... x_n$ y $x_i \in 0, 1$ (como la definida en la sección 3.1). Para construirlo la función utilizada debe tener como entrada variables con valor $\{-1, 1\}$, en lugar de $\{0, 1\}$, para lo que se debe realizar un cambio de variable:

$$f(x) \to C(z)$$
$$x_i \to \frac{1 - z_i}{2}$$

De esta forma $z_i = \begin{cases} 1 \text{ si } x_i = 0 \\ -1 \text{ si } x_i = 1 \end{cases}$. La forma de pasar de la función *C(z)* al operador *C* es sustituyendo las variables z_i por puertas σ_i^z. Esto es debido a que los autovalores de σ^z son $\{-1, 1\}$.

[1] En computación cuántica los operadores lineales deben ser unitarios. Tiene relación con la idea $\sum_{|x\rangle} P(|x\rangle) = 1$, donde $|x\rangle$ itera sobre los vectores de la base computacional del sistema y $P(|x\rangle)$ se refiere a la probabilidad de medir i sobre dicha base.

3.3. Algoritmo de QAOA

La estructura general del algoritmo es la siguiente:

1. Inicializar $\vec{\beta}$ y $\vec{\gamma}$ a valores arbitrarios.
2. Repetir hasta que se cumpla cierto criterio de convergencia, obteniendo $\vec{\beta_{opt}}, \vec{\gamma_{opt}}$
   2.1. Construir el circuito correspondiente al estado $|\psi(\vec{\beta}, \vec{\gamma})\rangle$
   2.2. Medir el estado en la base computacional, lo que devuelve una cadena de bits $\{0, 1\}^n$
   2.3. Computar el valor esperado de C: $\langle\psi(\vec{\beta}, \vec{\gamma})|C|\psi(\vec{\beta}, \vec{\gamma})\rangle$. Esto puede ser calculado de manera equivalente evaluando la cadena de bits, obtenida de medir el estado, con la función de coste clásica *f(x)*.
   2.4. Hallar nuevos $\vec{\beta_{new}}, \vec{\gamma_{new}}$ usando un algoritmo clásico de optimización y volver al paso 2..
3. Calcular el valor esperado: $\langle\psi(\vec{\beta_{opt}}, \vec{\gamma_{opt}})|C|\psi(\vec{\beta_{opt}}, \vec{\gamma_{opt}})\rangle$. Este valor aproximará $\min_x f(x)$

Para el estado construido se cumple que para un número de capas $p \to \infty$:

$$\min_{\beta,\gamma}\langle\psi(\vec{\beta}, \vec{\gamma})|C|\psi(\vec{\beta}, \vec{\gamma})\rangle = \min_x f(x)$$

Esto es equivalente a decir que el estado fundamental del sistema corresponde al valor mínimo de la función de coste.

De esta forma, el optimizador clásico llamará en repetidas iteraciones a una función *execute_circuit* con una entrada $(\vec{\beta}, \vec{\gamma})$ que devolverá como salida un valor mayor o igual al mínimo de la función de coste. Este resultado es el que el optimizador debe minimizar.

En este caso el optimizador clásico que se utilizará es COBYLA (Constrained Optimization BY Linear Approximation), de la librería de *Python scipy.minimize*.

4

Desarrollo

4.1. Tutorial de Qiskit - Max Cut

Para asegurar una implementación correcta del algoritmo QAOA primero se aplica al problema presentado en el tutorial de Qiskit. [4] Este problema consiste en resolver MAX-CUT para el grafo 4.1, que consiste en, asignando un valor 0 o 1 a cada nodo maximizar la cantidad de aristas entre nodos de distinto valor.

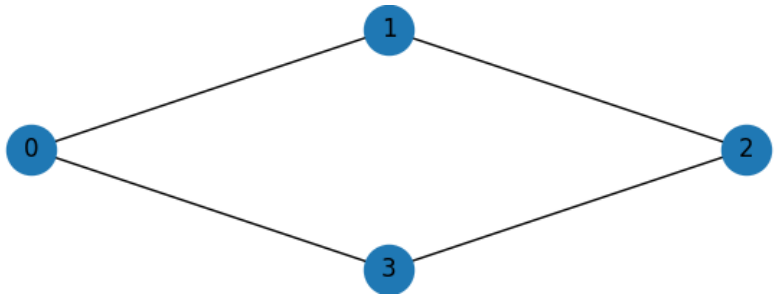

Figura 4.1:

Se parte de este caso por tratarse de uno sencillo. Esto es porque la cantidad de qubits es pequeña y no existen restricciones en este problema, por lo que el espacio de estados válidos disponibles es de 2^n, siendo *n* el número de qubits del sistema. Estas condiciones hacen que el algoritmo trabaje sobre un circuito pequeño, lo que reduce el ruido, y fácil de implementar.

- **Formulación:**
  Sea $G = (V, E)$ el grafo de la figura 4.1, con $V = [0, 1, 2, 3]$ y $E = [(0, 1), (1, 2), (2, 3), (0, 3)]$ los nodos y aristas de G respectivamente

- **Objetivo:**

$$max(\sum_{(i,j)\in E} (x_i * (1 - x_j) + x_j * (1 - x_i))) \tag{4.1}$$

Como *QAOA* sirve para encontrar el mínimo de una función, se tomará la ecuación [4.1] $* -1$.

De esta forma se tiene que la función de coste a minimizar para este problema en particular es la siguiente:

$$f(x) = \sum_{(i,j)\in E} (x_i * (x_j - 1) + x_j * (x_i - 1))$$

El cambio de variable ($x_i \to \frac{1-z_i}{2}$) de acuerdo con la sección genera la siguiente función en formato Ising:

$$\begin{aligned} g(z) &= \sum_{(i,j)\in E} (\frac{1-z_i}{2} * (\frac{1-z_j}{2} - 1) + \frac{1-z_j}{2} * (\frac{1-z_i}{2} - 1)) = \\ &= \sum_{(i,j)\in E} \frac{z_i z_j - 1}{2} \end{aligned}$$

Para obtener el operador C se tienen que tener en cuenta que debido al postulado de medición en mecánica cuántica [5] la fase global es despreciable. Esto significa que dado un operador lineal A y $n \in \mathbb{R}$:
$e^{i\gamma n} \cdot e^{i\gamma A} = e^{i\gamma A}$.

También se emplean las siguientes definiciones:

- $Rz_i(\lambda) = exp(-i\frac{\lambda}{2}\sigma_i^z)$
- $Rz_iz_j(\lambda) = exp(-i\frac{\lambda}{2}\sigma_i^z \otimes \sigma_j^z)$

De esta forma:

$$\begin{aligned} U(C,\gamma) = \exp(-i*\gamma*C) &= \exp\left(-i*\gamma*\sum_{(i,j)\in E} \frac{\sigma_i^z \otimes \sigma_j^z - 1}{2}\right) \\ &= \prod_{(i,j)\in E} \exp\left(-i*\gamma*\frac{\sigma_i^z \otimes \sigma_j^z - 1}{2}\right) \\ &= \prod_{(i,j)\in E} [\exp\left(-i*\frac{\gamma}{2}*\sigma_i^z \otimes \sigma_j^z\right) * \exp\left(i*\frac{\gamma}{2}\right)] \\ &= \prod_{(i,j)\in E} Rz_iz_j(\gamma) \end{aligned}$$

Circuito obtenido.

Con el operador $U(B,\beta)$ y el vector inicial, definidos en la sección 3.2, y el operador $U(C,\gamma)$ obtenido se puede construir el circuito cuántico.

4.2. Grafo simple - Camino más corto

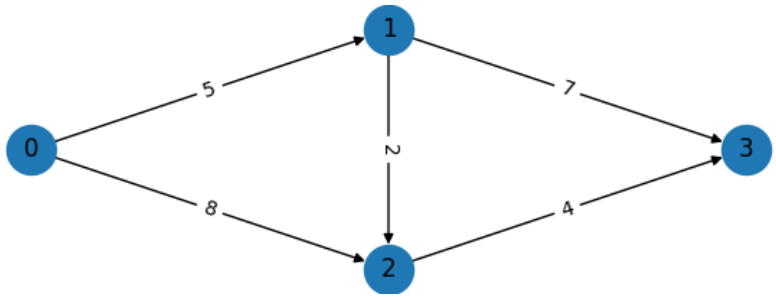

Figura 4.2:

Con este grafo se pretenden analizar los resultados obtenidos en la sección 2.2 (*Single-Objective Quantum Routing Optimization*) del artículo [6].

El problema a resolver es encontrar el camino más corto que conecte los nodos *0* y *3*.

- **Objetivo:**

$$min(5X_{01} + 8X_{02} + 2X_{12} + 7X_{13} + 4X_{23})$$

$$\text{dde } X_{ij} = \begin{cases} 1 & \text{si el camino contiene la arista del nodo } i \text{ al } j \\ 0 & \text{en otro caso} \end{cases}$$

- **Restricciones:** También se deben añadir una serie de restricciones para evitar caminos triviales o incongruentes.

$$X_{01} + X_{02} = 1 \tag{4.2}$$

$$X_{01} = X_{12} + X_{13} \tag{4.3}$$

$$X_{02} + X_{12} = X_{23}$$

Estas restricciones se pueden justificar como:

1. Debe haber exactamente un eje del camino que involucre al nodo de comienzo. Obliga al camino a comenzar por dicho nodo.
2. Para cada nodo intermedio debe haber en el resultado tantas aristas entrantes como salientes. Evita caminos incongruentes y hace que el único nodo posible para finalizar sea el nodo final.

Siguiendo el caso del artículo se elige como valor del *modificador de Lagrange* $P = 27$, ya que debe ser estrictamente mayor que el máximo de la función objetivo: $\max_x f_{\text{sin restricc}}(x) = \sum_{(i,j)\in E} w_{ij} = 26$

De acuerdo con los pasos descritos en la sección 3.1, la función de coste clásica en su versión QUBO es:

$$\begin{aligned} f(X) =& 5X_{01} + 8X_{02} + 2X_{12} + 7X_{13} + 4X_{23} + \\ & P(X_{01} + X_{02} - 1)^2 + P(X_{01} - X_{12} - X_{13})^2 + P(X_{02} + X_{12} - X_{23})^2 \end{aligned}$$

El número de qubits del sistema cuántico es igual a la cantidad de variables de la función de coste, esto es, la cantidad de ejes del grafo.
Las variables X de la función de coste tienen valores 0 o 1 y para la conversión a su correspondiente función de coste implementada en un circuito cuántico se utiliza la formulación Ising. Estas nuevas variables (z) van a tomar valores -1 y 1 y cada variable z_k estará asociada al qubit k-ésimo del circuito. En este caso la correspondencia entre X_{ij} y z_k es la siguiente: X_{01} corresponde con z_0, X_{02} corresponde con z_1, X_{12} corresponde con z_2, X_{13} corresponde con z_3, X_{23} corresponde con z_4.

Como ya se ha visto en la sección cada variable z_i va a corresponder con una puerta Pauli-Z en el qubit i.

De acuerdo con la sección 3.2.2 la versión Ising de la función de coste queda como:

$$\begin{aligned} g(z) =& 5\frac{1-z_0}{2} + 8\frac{1-z_1}{2} + 2\frac{1-z_2}{2} + 7\frac{1-z_3}{2} + 4\frac{1-z_4}{2} + \\ & P(\frac{1-z_0}{2} + \frac{1-z_1}{2} - 1)^2 + P(\frac{1-z_0}{2} - \frac{1-z_2}{2} - \frac{1-z_3}{2})^2 + \\ & P(\frac{1-z_1}{2} + \frac{1-z_2}{2} - \frac{1-z_4}{2})^2 = \\ =& 11z_0 - 17{,}5z_1 - 28z_2 - 17z_3 + 11{,}5z_4 + \\ & 13{,}5(z_0z_1 - z_0z_2 - z_0z_3 + z_1z_2 - z_1z_4 + z_2z_3 - z_2z_4) + \\ & 80{,}5 \end{aligned}$$

Esta igualdad solo se cumple para variables con valores $\{-1, 1\}$, ya que $z_i^2 = 1$.

De forma similar a la sección 4.1 se obtiene el hamiltoniano $U(C, \gamma)$ a partir de $g(z)$:

$$\begin{aligned} & U(C,\gamma) = exp(-i\gamma C) = \\ & Rz_0(11 * 2\gamma) \cdot Rz_1(-17{,}5 * 2\gamma) \cdot Rz_2(-28 * 2\gamma) \cdot Rz_3(-17 * 2\gamma) \cdot Rz_4(11{,}5 * 2\gamma) \cdot \\ & Rz_0z_1(+13{,}5 * 2\gamma) \cdot Rz_0z_2(-13{,}5 * 2\gamma) \cdot Rz_0z_3(-13{,}5 * 2\gamma) \cdot Rz_1z_2(+13{,}5 * 2\gamma) \cdot \\ & Rz_1z_4(-13{,}5 * 2\gamma) \cdot Rz_2z_3(+13{,}5 * 2\gamma) \cdot Rz_2z_4(-13{,}5 * 2\gamma) \end{aligned}$$

Con el operador $U(B, \beta)$ y el vector inicial, definidos en la sección 3.2, y el operador $U(C, \gamma)$

obtenido se puede construir el circuito cuántico.

- **Circuito obtenido ($p = 1$):**

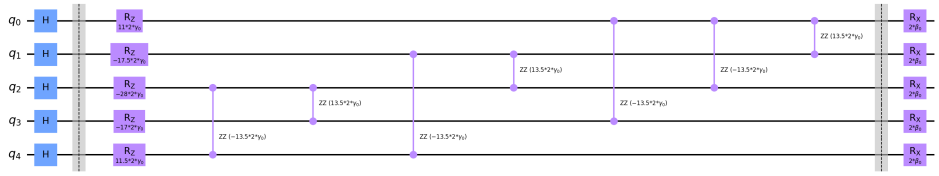

Figura 4.3:

- **Circuito del artículo ($p = 1$):**

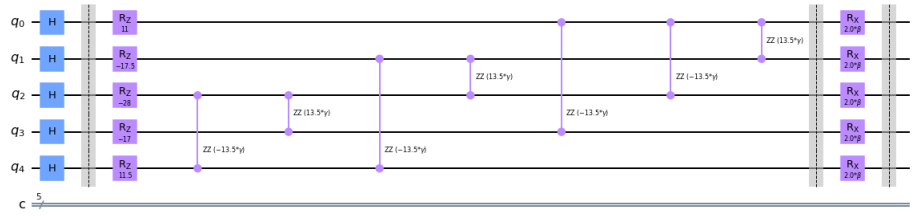

Figura 4.4:

4.2.1. Diferencias con el artículo

El circuito obtenido teóricamente difiere del que se puede ver en la sección 2.2 del artículo [6]. En la sección anterior se obtienen operadores con la forma $Rz(n * 2\gamma)$, mientras que en la imagen del circuito en el artículo aparecen como $Rz(n)$.

Debido a esto, y como se busca replicar los resultados del artículo, se modifica el hamiltoniano para que sea igual al de la imagen.

5

Integración, pruebas y resultados

En este capítulo se va a mostrar cómo se ha realizado la integración de la aplicación en los dispositivos de los usuarios y algunas de las pruebas que se han realizado sobre la aplicación.

Para las pruebas se ha aplicado la técnica de evaluación heurística. El objetivo de esta técnica es tener una primera toma de contacto con la aplicación, para verificar tanto la calidad como la usabilidad de esta, así como identificar posibles errores en la aplicación desarrollada y así poder solventarlos.
El procedimiento que se ha llevado a cabo ha sido el testeo por un usuario externo. Este usuario no tenía conocimiento alguno de la aplicación. El usuario ha comprobado todas las funcionalidades que ofrece la aplicación, así como la observación de la usabilidad, documentación y calidad del producto a probar. Además, se ha comprobado que la aplicación cargada en el núcleo no tenía ningún problema, como por ejemplo un uso excesivo de memoria o bucles infinitos que podrían dejar colgado al núcleo. También se ha comprobado la correcta liberación de memoria en C. Por último, se ha lanzado la aplicación en producción y se han hecho pruebas de estrés creando flujos de tráfico mediante la herramienta perf de Linux. La aplicación ha pasado todas las pruebas en Ubuntu, Linux y funciona en los principales navegadores adaptándose al tamaño de la pantalla.

Figura 5.1: Se puede observar como la aplicación apenas usa un 6 % de la memoria del ordenador y muy poco procesador, con un tiempo en línea de 3 días.

6

Conclusiones y trabajo futuro

En este capítulo de la memoria se darán unas conclusiones sobre el aprendizaje que ha supuesto este proyecto y un planteamiento de trabajo futuro para mejorar la aplicación creada.

Conclusiones

La principal idea de este proyecto ha sido crear un cortafuegos muy sencillo de utilizar para el usuario y con la capacidad de que un desarrollador pueda incorporar sus propias huellas y configuraciones fácilmente. Esto puede ser complejo ya que hay que tener especial cuidado con el código que se inserta en el núcleo, una parte crítica del sistema, y se le añade la dificultad de comunicar la aplicación del núcleo con la del espacio usuario y con la interfaz escrita en un lenguaje de alto nivel.
La elaboración de este proyecto me ha permitido apreciar las diferentes fases de un proyecto de programación, como la investigación, el diseño y la metodología de desarrollo para construir una aplicación funcional en el menor tiempo posible. También he comprendido la importancia del uso de una metodología ágil con el fin de mejorar la calidad y productividad dividiendo el proyecto en pequeñas partes.
Esta aplicación combina el bajo nivel con el alto nivel y usa tecnologías novedosas, como XDP, que hace que los errores no sean tan fáciles de resolver debido al pequeño tamaño de la comunidad de desarrolladores y las constantes actualizaciones al tratarse de una tecnología nueva.
Por último, se ha construido una aplicación cortafuegos para Linux completamente funcional y que cumple con todos los requisitos funcionales y no funcionales. Esta aplicación es capaz de utilizar una tecnología novedosa a bajo nivel, en el núcleo de Linux y, utilizando el alto nivel proporciona una interfaz sencilla para el usuario y facilita el desarrollo de soluciones de otros programadores mediante el uso de una API REST. El diseño de la interfaz se adapta al tamaño de la pantalla.

Trabajo futuro

En cuanto al trabajo futuro se pretende traducir la aplicación a otros idiomas, como el inglés, con el fin de que pueda ser utilizada por más personas. Por otro lado, se quiere añadir al repositorio GitHub las huellas y soluciones aportadas por otros desarrolladores para que puedan ser utilizadas por toda la comunidad. También es posible mejorar la aplicación creando nuevas soluciones de detección de huellas, actualizándola y mejorando su compatibilidad con todos los sistemas operativos.

Bibliografía

[1] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] J. Preskill, “Quantum computing in the nisq era and beyond,” *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: http://dx.doi.org/10.22331/q-2018-08-06-79

[3] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” 2014.

[4] Qiskit, “Qaoa,” accessed: 2024-03-11. [Online]. Available: https://github.com/Qiskit/textbook/blob/main/notebooks/ch-applications/qaoa.ipynb

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[6] H. Urgelles Pérez, P. Picazo-Martinez, D. Garcia-Roger, and J. Monserrat, “Multi-objective routing optimization for 6g communication networks using a quantum approximate optimization algorithm,” *Sensors*, vol. 22, p. 7570, 10 2022.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID