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

mayo 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 el campo de la computación, a lo largo de los años se van desarrollando nuevos métodos que permiten una mayor capacidad de procesamiento para resolver problemas del día a día. Algunos ejemplos de los últimos años son el empleo de Big Data, la computación en la nube o el auge de la inteligencia artificial, entre muchos otros.

Necesariamente, a la par que estos avances, el tamaño de los problemas que deben ser resueltos también aumenta, que a su vez requieren de algoritmos más eficientes para su resolución. Dada esta situación aparecen nuevos paradigmas, que generan marcos en los que se puede producir el desarrollo de nuevos algoritmos que superen a los preexistentes. Un caso paradigmático serían los algoritmos basados en heurísticas, los cuales priorizan el rendimiento a la exactitud.

1.1. Motivación

Es en este contexto en el que se desarrolla la computación cuántica, que promete ser una tecnología que permite la creación de algoritmos que resuelvan algunos de estos problemas de forma más eficiente. Al respecto se suele mencionar la revolución en el campo de la encriptación que supondría el algoritmo de Shor [1]. Este invalidaría las tecnologías de ofuscación basadas en algoritmos que utilicen el principio de que la factorización de un entero en números primos es un problema que escala de forma muy ineficiente con respecto al tamaño de la entrada. 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 los problemas de optimización también se proponen varios algoritmos alternativos que permitirían una mejora en la escalabilidad.

Esta superioridad cuántica pfrente a la clásica, 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 hace que tomen importancia los algoritmos híbridos, como QAOA [3] o VQE, que combinan la ejecución de circuitos cuánticos pequeños con el pre y post-procesamiento en un ordenador clásico.

El objetivo de este trabajo es estudiar la implementación en detalle de uno de estos algoritmos

híbridos, QAOA (Quantum Approximation Optimization Algorithm). Conviene mencionar que existe un segundo QAOA (Quantum Alternating Operator Ansatz) [4] que es una generalización del primero, con los operadores que se utilizan y la utilización de qudits (qubits de n-dimensiones) en lugar de qubits. En este trabajo siempre que se mencione QAOA se referirá al primero, definido en el artículo de (Farhi et al., 2014) [3].

Para tener una referencia con la que comparar los resultados se implementarán los mismos problemas aplicados para QAOA en el algoritmo implementado por los sistemas D-Wave, el Quantum Annealing (QA). Se ha escogido QA como comparación porque el tipo de problemas que resuelve es el mismo que QAOA y porque estos algoritmos se basan en principios similares a nivel teórico. 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 de coste sin restricciones de la forma $f : \{0,1\}^n \to \mathbb{R}$ para algún $n \in \mathbb{N}$. En ambos casos, el método para alcanzar el extremo de f consiste en conseguir que al medir el estado fundamental del hamiltoniano que describe el sistema cuántico se obtenga el extremo 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 problema computable (aunque no necesariamente de manera más eficiente) 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 sistemas de uso generalista, que no alcanzan los 1000 qubits manteniendo un funcionamiento tolerante a fallos [1].

Dados estos algoritmos, en las siguientes páginas se busca replicar implementaciones de QAOA obtenidas de diferentes fuentes con el objetivo de estudiar su rendimiento y comprender su funcionamiento.

[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

El punto de partida es el trabajo desarrollado por Urgelles et al. [5], El algoritmo utilizado es QAOA Y TALTALTAL

con la intención de replicar la instancia de QAOA. Este aplica el algoritmo para resolver el problema de hallar el camino más corto en un grafo, con el pretexto de aplicarlo al enrutamiento de paquetes en redes de comunicaciones, y aplica también QAOA para una versión multi-objetivo del mismo problema, versión que no va a ser tratada en este trabajo.

2.1. Complejidad

Al categorizar algoritmos para resolver problemas computacionales se pueden distinguir dos grupos: El primero son algoritmos deterministas, en los que se garantiza obtener el resultado correcto siempre. En segundo caso se encuentran algoritmos basados en heurísticas, los cuales tienen una posibilidad no nula de no obtener el resultado correcto. La idea de utilizar algoritmos falibles radica en que existen situaciones en las que es preferible aceptar un porcentaje de error acotado si eso se traduce en una disminución considerable en la complejidad temporal o espacial.

Esto se aplica también a la computación cuántica, donde hay algoritmos deterministas, y no deterministas. El último es el caso de los algoritmos tratados en este trabajo, tanto QAOA como QA, los cuales dan un resultado óptimo con una probabilidad de error de no ser correcto.

Por supuesto, se contempla la utilización del paradigma de la computación cuántica en problemas en los que la complejidad de un algoritmo cuántico escale más lento que la de los métodos clásicos correspondientes.

En computación los problemas pueden ser divididos en dos grandes grupos:

- *P*: Aquellos problemas que pueden ser resueltos en tiempo polinomial.
- *NP*: Problemas cuyas soluciones son verificables en tiempo polinomial o, de manera equivalente, que pueden ser resueltos en tiempo polinomial por una máquina de Turing no determinista.

De forma similar, al introducir computadoras cuánticas se introducen dos nuevos conjuntos, también de interés para resolver problemas:

- *EQP* (Exact Quantum Polynomial time): Conjunto de problemas que pueden ser resueltos por una computadora cuántica utilizando un algoritmo determinista en tiempo polinómico. Esto sería análogo a los problemas tipo *P*.
- *BQP* (Bounded-error Quantum Polynomial time): Conjunto de problemas resolubles por una computadora cuántica en tiempo polinómico con una probabilidad de error mayor que 0.

2.2. Problemas de optimización combinatoria

Un problema de optimización combinatoria con variables binarias se define una función de coste de la forma:

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

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

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

El objetivo del problema es encontrar el mínimo global de esta función. De la misma forma, el objetivo puede ser definido como hallar el máximo de $-1 * f(x)$.

Además el espacio de estados puede estar restringido, esto es, que la cantidad de x válidos sea $< 2^n$ o, de manera equivalente, que existan cadenas de n bits que no sean válidas en el contexto del problema (en el caso del problema del camino más corto sería, por ejemplo, un camino vacío o discontinuo).

Los problemas aplicados tanto a *QAOA* como a *QA* son problemas tipo *QUBO* (Quadratic Unconstrained Binary Optimization). Estos son problemas en los que el espacio de estados posibles no está restringido, por lo que para resolver un problema utilizando alguno de estos dos algoritmos se debe realizar una conversión entre problemas de optimización genéricos a problemas tipo *QUBO*. Esta conversión se explica en la *sección 2.2*.

2.3. QAOA

Este algoritmo es utilizado para resolver problemas de optimización en formato Ising. Para esto, se construye un hamiltoniano cuyo estado de mínima energía corresponda con el mínimo de la función

de optimización.

Dada una función de coste $C(x)$, un hamiltoniano H y un espacio de resultados 2^n esta igualdad se describe de la siguiente forma:

$$\forall x \in 2^n, C(x) = \langle x| H |x\rangle$$

Como este operador en un problema de optimización no es necesariamente unitario se debe utilizar el operador e^{iH} para ser descrito en un circuito cuántico, en el que en lugar de encontrar el estado de menor energía se debe encontrar el estado de menor fase.

Para esto, en el algoritmo QAOA se utilizan dos operadores unitarios.

- El primer operador es $U(C, \gamma) = e^{-i\gamma H}$, denominado hamiltoniano del problema (problem hamiltonian), correspondiente al operador previamente mencionado.

  Dado un estado con una serie de valores posibles (correspondientes a los estados de la base computacional) este operador separa las fases relativas de dichos valores, de tal forma que la fase de los valores con menor coste disminuye y la de los valores con mayor coste aumenta. Esto no modifica la probabilidad de medición, ya que modificar la fase relativa manteniendo el eje de medición no modifica la probabilidad de medición de estos valores.

- Para que estos cambios en las fases relativas se traduzcan en un aumento de la probabilidad de medición de los valores deseados se utiliza el operador $U(B, \beta) = e^{-i\beta B}$, denominado hamiltoniano de mezcla o interferencia (mixer hamiltonian).

  Este operador realiza una rotación en el estado, de tal forma que los valores con menor fase son más probables de observar que los valores con mayor fase.

En la *sección A* del apéndice se muestra un ejemplo para ilustrar los efectos de los dos operadores descritos.

2.4. Computación adiabática cuántica

La computación cuántica adiabática (**AQC**) es, en contraposición con un enfoque orientado a 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. Pasar de problemas de optimización combinatoria a QUBO

En la *sección 2.2* se explica que los problemas de optimización con restricciones deben ser transformados para su posterior procesamiento por los algoritmos tratados. Una restricción puede ser definida como una igualdad de la forma $A(x) = B(x)$ que debe cumplir toda entrada x de la función de coste para considerarse válida. La forma de convertir un problema con restricciones a uno sin ellas es modificar la función de coste para que las incluya en su definición, por lo que en lugar de tener $f(x)$ como función de coste se tendrá: $f'(x) = f(x) + P * (A(x) - B(x))^2$, 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 deberá tener 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 la función de coste $f(x)$.

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:

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

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:

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

3.2.1. Estado inicial

El estado inicial del qubit se define como

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

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.

[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.

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 $\forall\, |x\rangle = [x_1 x_2 \dots x_n], \langle x| C |x\rangle = f(x)$. 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 coste de la forma $f(x)$, donde $x = x_1 x_2 \dots x_n$ y $x_i \in 0, 1$. Para construirlo la función utilizada debe estar en formato Ising, en lugar de la versión QUBO (caso de $f(x)$), lo cual se traduce en tener como entrada variables con valor $\{-1, 1\}$, en lugar de $\{0, 1\}$. Para ello se debe realizar un cambio de variable:

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

De esta forma $z_i = 1$ si $x_i = 0$ y $z_i = -1$ si $x_i = 1$. La forma de pasar de la función $C(z)$ al operador C es sustituyendo las variables z_i por puertas σ_i^z. La justificación de esto se encuentra en la *sección C* del apéndice.

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 $\mín_x f(x)$

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

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

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. MAX-CUT en grafo de 4 aristas

Para asegurar una implementación correcta del algoritmo QAOA primero se aplica a un problema presentado en la página web de Qiskit [6]. 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.

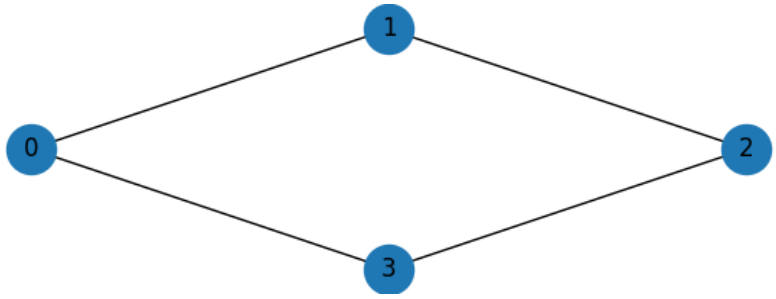


Figura 4.1: Grafo obtenido de *[6]*

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 facilita su implementación.

- **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:**

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

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:

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

Para obtener el operador C se tienen que tener en cuenta que debido al postulado de medición en mecánica cuántica ([7]) 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:

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

De esta forma:

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

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.

Diferencias con el circuito mostrado por Qiskit

El circuito de ejemplo (*Figura 4.3*) corresponde a desarrollar los mismos cálculos solo que con $f'(x) = 2 * f(x)$.
Aunque las puertas *Rzz* tengan coeficientes distintos ambas son correctas. Esto es porque ambos circuitos resultantes tienen un mismo estado fundamental, solo que uno lo tiene con el doble de energía

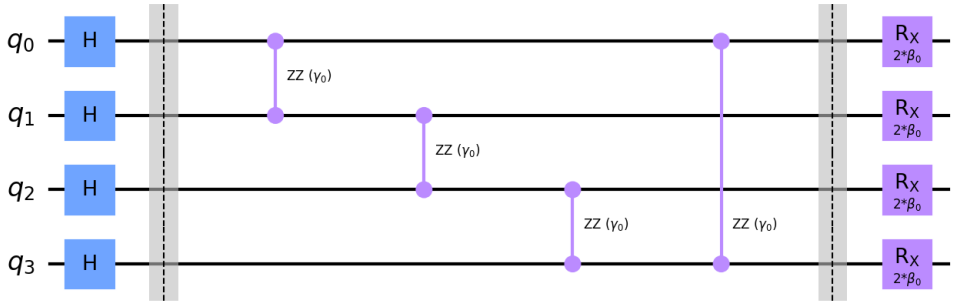


Figura 4.2: Circuito obtenido ($p = 1$)

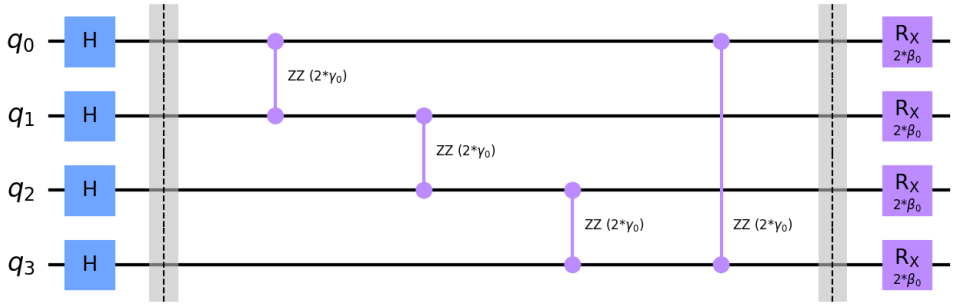


Figura 4.3: Circuito de ejemplo de la página web [6] ($p = 1$)

que el otro. En otras palabras ambas funciones f' y f tienen un mismo x que las minimiza.

4.2. Camino más corto en grafo de 4 nodos

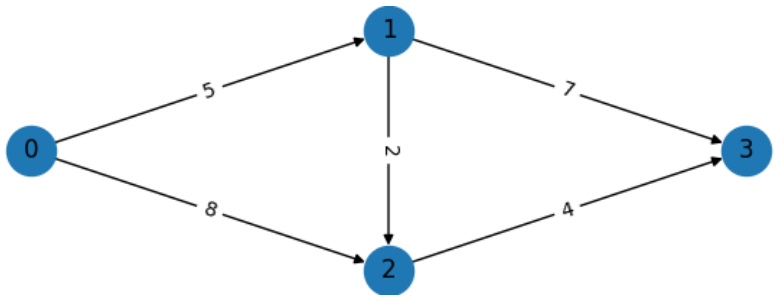


Figura 4.4: Grafo obtenido del artículo [5]

En la *sección 4.1* se consiguieron replicar los cálculos teóricos de una instancia de QAOA simple, sin restricciones y poco profunda. Por ello el siguiente problema a resolver aumenta su complejidad, añadiendo restricciones y aumentando la cantidad de qubits del sistema.
En esta sección se pretenden imitar los resultados obtenidos en la sección 2.2 (*Single-Objective Quantum Routing Optimization*) del *artículo [5]*.

El problema consiste en encontrar el camino más corto que conecte los nodos *0* y *3*.

- **Objetivo:**

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

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

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

1. $X_{01} + X_{02} = 1$: Debe haber exactamente un eje del camino que involucre al nodo de comienzo. Obliga al camino a comenzar por dicho nodo.
2. $X_{01} = X_{12} + X_{13}$
   $X_{02} + X_{12} = X_{23}$: 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: $\text{máx}_x f_{\text{sin restricc}}(x) = \sum_{(i,j)\in E} w_{ij} = 26$

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 y 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 $g(z)$:

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

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.

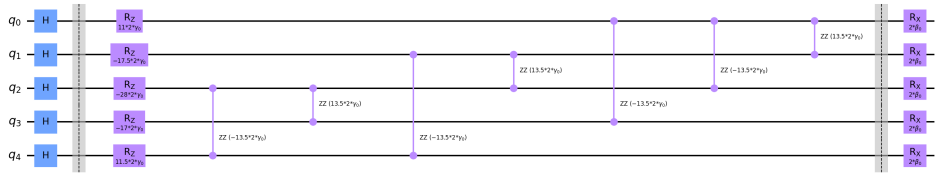


Figura 4.5: Circuito obtenido ($p = 1$)

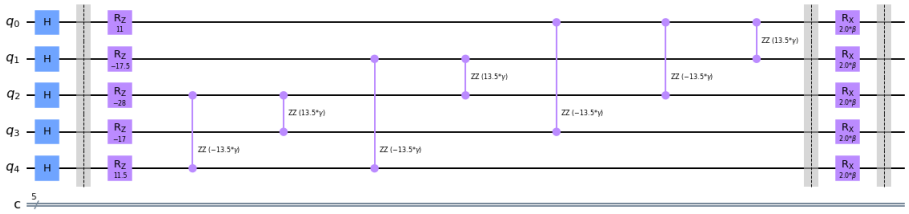


Figura 4.6: Circuito del artículo ($p = 1$)

Diferencias con el artículo

El *circuito 4.6* obtenido teóricamente difiere del que se puede ver en la sección 2.2 del *artículo [5]*. 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.

4.3. Camino más corto para estudiar la variación con el número de capas

Debido a los resultados obtenidos en el problema del camino más corto estudiado anteriormente (*sección 5.2*), se decide aplicar las mismas pruebas sobre otro grafo con el mismo problema.

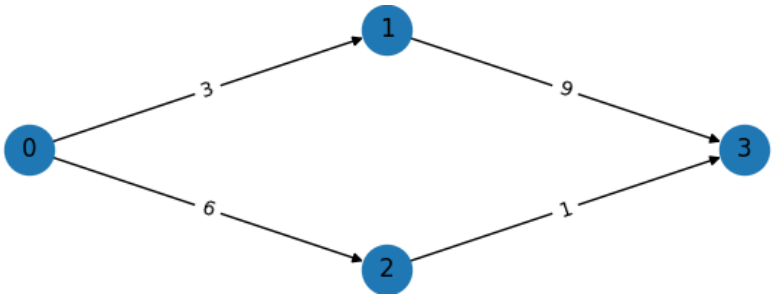


Figura 4.7: Grafo obtenido del *artículo [8]*

Este grafo (*figura 4.7*) tiene la misma cantidad de nodos que el grafo 4.4 y una arista menos. Esto se traduce en una aplicación de QAOA con menos restricciones pero con los mismos qubits. Con esta diferenciación se esperan obtener resultados que reproduzcan o corrijan el comportamiento dado con el grafo anterior al aumentar el número de capas.

- **Objetivo:**

$$\text{mín}(3X_{01} + 6X_{02} + 9X_{13} + 1X_{23})$$

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

- **Restricciones:**

1. $X_{01} + X_{02} = 1$: Equivalente a la *restricción 1.* del problema previo.
2. $X_{01} = X_{13}$
   $X_{02} = X_{23}$: Equivalentes a las *restricciones 2.* del problema previo.

3. $X_{13} + X_{23} = 1$:

Con el fin de mantener las similitudes con el problema anterior se elige el modificador de Lagrange (*P*) utilizando la misma métrica, a saber: $P > \text{máx}_x f_{\text{sin restricc}}(x) = \sum_{(i,j)\in E} w_{ij} = 19$. Por este motivo se escoge $P = 20$

Al añadir las restricciones, como se explica en la *sección 3.1*, la función de coste definitiva en formato QUBO queda como:

$$f(X) = 3X_{01} + 6X_{02} + 9X_{13} + 1X_{23} + \\ + P(X_{01} + X_{02} - 1)^2 + P(X_{13} + X_{23} - 1)^2 + P(X_{01} - X_{13})^2 + P(X_{02} - X_{23})^2$$

Para el paso de la función de la versión QUBO a versión Ising se debe realizar un cambio de variable $X_{ij} \to \frac{1-z_k}{2}$. Con este paso se hará que, como las variables X_{ij} toman valores $\{0, 1\}$, las variables z_k tomen valores $\{-1, 1\}$. Además cada variable z_k corresponderá al qubit *k*-ésimo. La correspondencia entre variables X_{ij} y z_k es la siguiente: X_{01} corresponde con z_0, X_{02} corresponde con z_1, X_{13} corresponde con z_2 y X_{23} corresponde con z_4.

Al igual que para los problemas anteriores, y de acuerdo con la sección cada variable z_i corresponde a un operador Pauli-Z en el qubit i.

La versión Ising de la función de coste queda de esta forma: [1]

$$\begin{aligned} g(z) = & \, 3\frac{1-z_0}{2} + 6\frac{1-z_1}{2} + 9\frac{1-z_2}{2} + 1\frac{1-z_3}{2} + \\ & + P(\frac{1-z_0}{2} + \frac{1-z_1}{2} - 1)^2 + P(\frac{1-z_2}{2} + \frac{1-z_3}{2} - 1)^2 + \\ & + P(\frac{1-z_0}{2} - \frac{1-z_2}{2})^2 + P(\frac{1-z_1}{2} - \frac{1-z_3}{2})^2 = \\ = & -1{,}5z_0 - 3z_1 - 4{,}5z_2 - 0{,}5z_3 + \\ & + 10 * (z_0z_1 - z_0z_2 - z_1z_3 + z_2z_3) + 49{,}5 \end{aligned}$$

El **problem hamiltonian** (descrito en la *sección 3.2*) partiendo de $g(z)$ es:

$$\begin{aligned} & U(C, \gamma) = \exp(-i\gamma C) = \\ & = Rz_0(-1{,}5 * 2\gamma) \cdot Rz_1(-3 * 2\gamma) \cdot Rz_2(-4{,}5 * 2\gamma) \cdot Rz_3(-0{,}5 * 2\gamma) \cdot \\ & \cdot Rz_0z_1(10 * 2\gamma) \cdot Rz_0z_2(-10 * 2\gamma) \cdot Rz_1z_3(-10 * 2\gamma) \cdot Rz_2z_3(10 * 2\gamma) \end{aligned}$$

Con el **mixing hamiltonian** y el vector inicial, definidos en la *sección 3.2* y el **problem hamiltonian** obtenido se puede construir el circuito cuántico.

[1] Dado $z_i \in \{-1, 1\}$, se cumple $z_i^2 = 1$

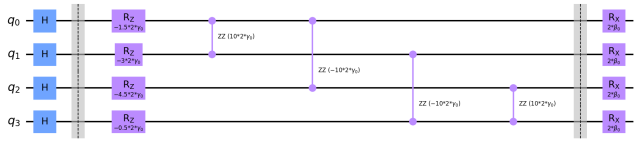


Figura 4.8: Circuito obtenido ($p = 1$)

En el artículo del que se ha obtenido el problema no se muestra la construcción del circuito de QAOA, por lo que el resultado obtenido en la *figura 4.8* no puede ser comparado.

5

Integración, pruebas y resultados

En las ejecuciones de QAOA, con el fin de medir la eficacia de los resultados obtenidos entre métricas distintas, se han realizado los siguientes tipos de pruebas:

Debido a la naturaleza probabilística de la computación cuántica, toda ejecución de un circuito se debe realizar varias veces, para obtener una aproximación de la probabilidad de los valores del estado. Así, cuando se habla del resultado de la ejecución de un circuito cuántico, como el mostrado en la *figura 5.6*, se refiere a probabilidades asociadas a cada valor, que han sido calculadas mediante sucesivas ejecuciones del programa. La cantidad de veces que se ejecuta cada programa, viene dado por las *muestras* (*shots*).

1. **NA/TE**: Con este método se busca despreciar el ruido presente en cada ejecución. Para ello se realizan *n* ejecuciones distintas del algoritmo para un número *p* de capas y se calcula el porcentaje de ejecuciones en los que se ha obtenido como resultado el camino óptimo del problema.

$$\text{NA/TE} = \frac{NA}{TE}$$

   Donde *NA* (número de aciertos) es la cantidad de ejecuciones en las que se ha encontrado el camino óptimo y *TE* es el total de ejecuciones realizadas.

   Ejemplo:

   Tomando como ejemplo de salida de QAOA la mostrada en la *figura 5.6* esto sumaría al cálculo de **NA/TE** $\frac{1}{n}$, ya que el camino óptimo es el obtenido en la mayor cantidad de muestras en la ejecución del circuito.

2. **MM/TE**:

   Lo que se busca con esta métrica es la media de veces que el camino óptimo aparece en la ejecución del circuito de QAOA.

   Como ya ha sido explicado, al ejecutar el circuito de QAOA en el último paso con $(\gamma_{opt}, \beta_{opt})$ el resultado obtenido tendrá tantos valores como *muestras* tenga la instrucción (por defecto 1024)

   Es el caso de gráficas como la de la *figura 5.6* en la que el eje vertical muestra la probabilidad

aproximada de cada valor.

De esta forma la estadística **MM**/**TE** se calcula como:

$$\frac{1}{TE}\sum_{i=1}^{TE}(MM_i)$$

Donde *TE* es el total de ejecuciones y MM_i es la media de las muestras en las que se han encontrado el resultado óptimo en la ejecución i.

Ejemplo:

En la *figura 5.6* la media de muestras del óptimo sería 0.59.

3. **Función gamma:** Se ha utilizado para comprobar la forma general que tiene la función *execute_circuit*, a minimizar por el optimizador clásico. Para ello se han realizado circuitos de una sola capa y se ha mantenido el parámetro $\beta = 1$. Se ha decidido así porque dicho parámetro se encarga del ángulo de rotación de los operadores σ_x, de construcción trivial en comparación con los operadores dependientes de γ. Al variar γ y graficar la función resultante se puede intuir la probabilidad de encontrar mínimos locales en lugar del mínimo global. Esto se traduce como la posibilidad de encontrar un resultado subóptimo para el problema, es decir, que el algoritmo falle.

A continuación se mostrarán los resultados de ejecución utilizando ambos paradigmas, esto es, QAOA y Quantum Annealing, además de explorar el rendimiento de las ejecuciones en Qiskit variando los métodos para construir la función de coste.

Para las ejecuciones de QAOA se utilizarán dos métodos distintos: una implementación basada en los cálculos desarrollados en la *sección 4* y un enfoque de caja negra, en el que se empleará la función *QAOAAnsatz* de *Qiskit* para generar un circuito cuántico de manera automática. Este último método se utilizará para comparar con el primero, para así verificar el funcionamiento de la implementación realizada.

5.1. MAX-CUT en grafo de 4 aristas

Esta sección corresponde a los resultados obtenidos con el problema descrito como ejemplo en la página web de Qiskit, replicado en la *sección 4.1*.

Dada la naturaleza del problema Max-Cut existen siempre al menos dos resultados (en este caso “1010” y “0101”) que devuelven el máximo de la función de coste ($C(x) = 4$). Un resultado “abcd” sería equivalente a dar un valor “a” al nodo 3, “b” al nodo 2 y así sucesivamente.

5.1.1. Resultados con QAOA

Las estadísticas obtenidas son las siguientes:

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	100 %	52.14 %
p = 2	100 %	98.17 %
p = 3	100 %	96.10 %
p = 4	100 %	98.70 %
p = 5	100 %	98.90 %

Tabla 5.1: Resultados de la ejecución del problema [6]

Según esta tabla, se encuentra el camino óptimo con cualquier número de capas el 100 % de las ejecuciones.

Analizando los resultados, se concluye que son inmejorables, ya que siempre se encuentra el resultado óptimo y aumentar el número de capas disminuye la aparición de otros valores en la ejecución del algoritmo.

Función gamma

Se ha graficado la función gamma para poder visualizar posibles mínimos locales que se pudieran encontrar al ejecutar el optimizador clásico:

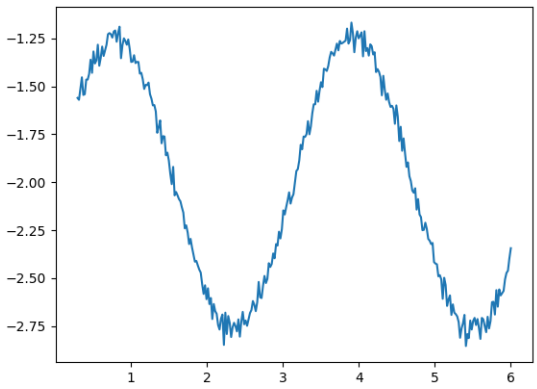


Figura 5.1: Función gamma con $\beta = 1{,}0$ y variando γ

Como se puede ver, la función toma una forma sinusoidal y, como se ve en gráficas posteriores, tiene mucho menos ruido. Esto hace ver que el resultado óptimo es fácil de encontrar, lo que se ve reafirmado comprobando los resultados de la *tabla 5.1*.

5.1.2. Resultados de D-Wave

El resultado de ejecutar el algoritmo de Quantum Annealing utilizando los sistemas de D-Wave es mostrado en la *tabla 5.2*.

Camino	Energía	Muestras
0101 (**Óptimo**)	-4.0	618
1010 (**Óptimo**)	-4.0	406

Tabla 5.2: Resultados de la ejecución de MAX-CUT en D-Wave

Al igual que en la implementación de QAOA, que corresponde a los resultados de la *tabla 5.1*, se obtienen unos resultados perfectos, ya que para el problema mencionado tanto *0101* como *1010* son los resultados óptimos.

5.1.3. Resultados con librería de QAOA

La *tabla 5.3* son estadísticas calculadas utilizando la función de Qiskit *QAOAAnsatz*.

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	100 %	51.30 %
p = 2	100 %	98.13 %
p = 3	100 %	95.83 %
p = 4	100 %	97.71 %
p = 5	100 %	99.50 %

Tabla 5.3: Resultados utilizando *QAOAAnsatz* con el problema de la *figura 4.1*

Esto ha sido realizado para comparar con la implementación de QAOA, cuyos resultados se muestran en la *tabla 5.1*. En ambas tablas se puede ver que siempre se ha encontrado el resultado óptimo y con el aumento de capas disminuye la aparición de otros resultados en el algoritmo en general. Con esta comparación se concluye que la implementación de QAOA de la *sección 4.1* es correcta.

5.2. Camino más corto en grafo de 4 nodos

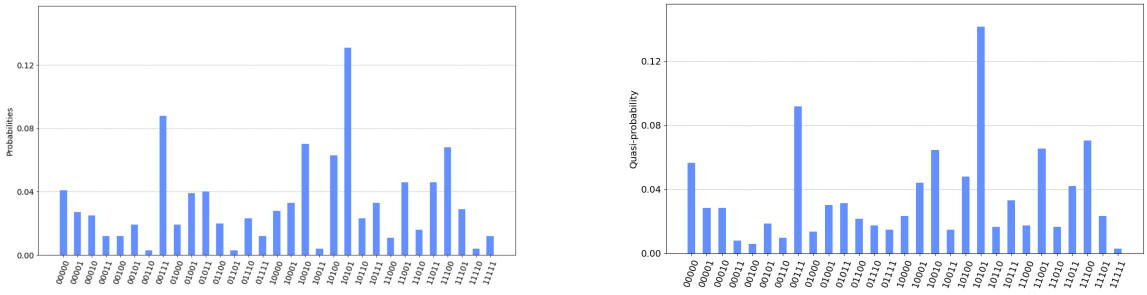
Como en la sección anterior se ha logrado una implementación correcta de QAOA para un problema simple se continúa con un caso en el que se aplica el algoritmo a un problema distinto, con más qubits y con restricciones.

Estos resultados recopilados corresponden con el problema descrito en la *sección 4.2*, en el que se resuelve el problema del hallar el camino más corto entre los nodos 0-3 para el grafo de la *figura 4.4*.

5.2.1. Resultados con QAOA

Solución del artículo

En las siguientes muestras se ha buscado replicar los resultados del *artículo [5]*, del que se ha obtenido el problema. Esto ha sido probado ya que, empleando los parámetros $\beta = 0{,}28517317$ y $\gamma = -5{,}05969577$ dados como óptimos, se obtiene un gráfico muy similar al del artículo:



(a) Resultado del artículo

(b) Resultado obtenido

Figura 5.2:

De esta forma, se asume que los resultados del artículo deberían ser equivalentes a los obtenidos en esta instancia del algoritmo.

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	91.3 %	39.34 %
p = 2	64.6 %	24.16 %
p = 3	63.4 %	18.82 %
p = 4	9.4 %	5.38 %
p = 5	67.9 %	19.45 %
p = 6	29.8 %	12.59 %
p = 7	28.9 %	9.12 %
p = 8	36.7 %	12.49 %

Tabla 5.4: Resultados de la ejecución de la versión de QAOA del artículo

El primer resultado a resaltar en la *tabla 5.4* es un empeoramiento de los resultados a medida que se aumenta el número de capas, lo cual es contrario a lo esperado teóricamente.

Además, la gran diferencia entre los resultados dados por las estadísticas **NA/TE** y **MM/TE** denotan una gran cantidad de ruido al ejecutar el algoritmo, lo cual se corrobora viendo los resultados de ejecuciones concretas, como los dados en la *figura 5.2*.

El resultado de la función gamma es el siguiente:

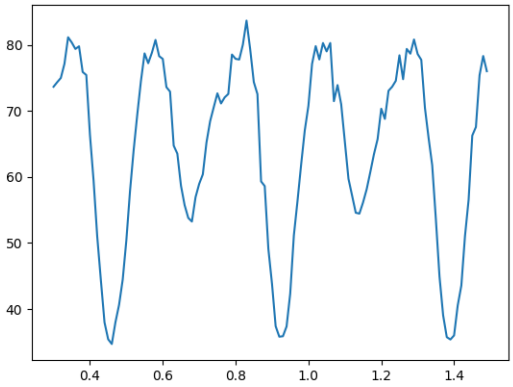


Figura 5.3: Función gamma. Caso del artículo

Se puede ver que existen un gran número de mínimos locales, lo cual dificulta la tarea del optimizador clásico. Esto se corrobora ya que, al emplear como parámetros iniciales $(\beta = 1{,}0, \gamma = 0{,}5)$, para $p = 1$ se obtiene el camino óptimo el 100 % de las ejecuciones.

Este proceso de inicializar los parámetros con valores concretos no sería una solución válida, ya que se trata de una metodología no automática en la que, para ejecutar correctamente el algoritmo, se necesitaría conocer antes su propio resultado. Además la ejecución correcta sucede para $p = 1$, pero al igual que el caso por defecto $(\beta = 1{,}0, \gamma = 1{,}0)$, no escala correctamente al aumentar el número de capas.

Modificaciones a la solución del artículo

Partiendo de la solución del artículo descrita en la *sección 5.2.1*, se han realizado cambios a la función de coste y a la construcción del circuito cuántico con el fin de encontrar el mejor resultado. Esto se ha realizado para poder comparar el rendimiento de la mejor solución encontrada con el rendimiento del algoritmo de Quantum Annealing de D-Wave.

Las modificaciones empleadas han sido las siguientes:

Ángulos correctos teóricamente

Al construir el circuito, utilizar para los operadores lineales los ángulos obtenidos teóricamente, en lugar de los vistos en el artículo (discutido en la *sección 4.2*).

Las estadísticas obtenidas en la *tabla 5.5* no dan ningún resultado satisfactorio. Se aprecia un comportamiento similar al visto en los resultados del grafo de la *tabla 5.9* en cuanto a que se mejoran las estadísticas hasta las 3 capas y luego disminuyen.

La función gamma resultante se muestra en la *figura 5.4*. Tiene un mínimo global menos claro con

nº Capas	NA/TE [1.]	MM/TE [2.]
p = 1	0.5 %	4.05 %
p = 2	9.2 %	5.83 %
p = 3	54.8 %	12.98 %
p = 4	11.1 %	7.00 %
p = 5	6 %	5.71 %

Tabla 5.5: Resultados de la ejecución de QAOA utilizando los ángulos obtenidos teóricamente

respecto a la función gamma de la versión del grafo del paper, en la *figura 5.3*. Además a diferencia de dicha gráfica, en esta no se aprecia periodicidad alguna.

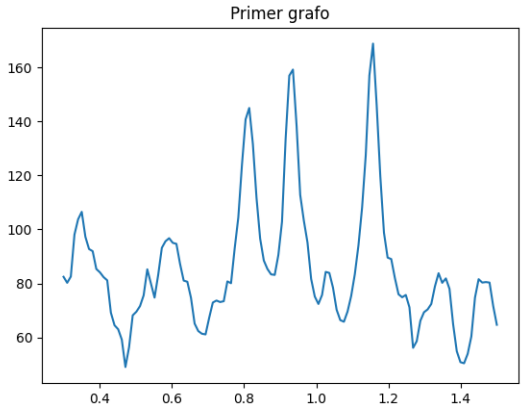


Figura 5.4: Función gamma. Con ángulos correctos teóricamente.

P=40

Incrementar el valor del parámetro P (correspondiente al modificador de Lagrange) al construir la función de coste, para así aumentar la penalización en caso de fallo.

Restricción extra

Añadir una restricción a la función de coste, que especifique que solo se puede acceder al último nodo por una de las aristas que lleguen a este.

$$X_{13} + X_{23} = 1 \tag{5.1}$$

Los resultados (*tabla 5.6*) presentan una ligera mejora con respecto a los presentes en la *tabla 5.4*, tanto para la estadística **NA/TE** como para **MM/TE**.

La función gamma resultante (*figura 5.5*) toma, en comparación con la original (*figura 5.3*), unos

nº Capas	NA/TE [1.]	MM/TE [2.]
p = 1	93.8 %	37.83 %
p = 2	64.6 %	26.16 %
p = 3	84.8 %	27.82 %
p = 4	56.0 %	23.47 %
p = 5	88.1 %	46.40 %
p = 6	88.1 %	21.83 %

Tabla 5.6: Resultados de la ejecución de QAOA añadiendo la restricción de la *fórmula 5.1*

valores elevados. Se distingue un incremento en la diferencia entre los mínimos globales y los mínimos locales (no globales).

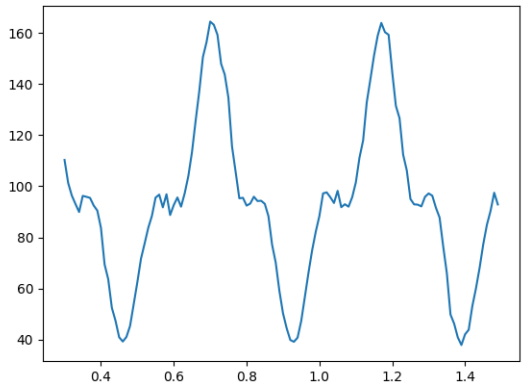


Figura 5.5: Función gamma. Con restricción extra

5.2.2. Resultados con QAOA en ordenador cuántico real

Utilizando la herramienta Runtime de Qiskit se han realizado ejecuciones del código para comprobar sus resultados en ordenadores reales.

El funcionamiento ha consistido en obtener los parámetros γ y β en dichos ordenadores y después ejecutar una vez más en un simulador para obtener el resultado al problema.

$p = 1$

El código ha sido ejecutando con el error encontrado en el artículo (*sección 4.2*), tratando así de asemejarse a los resultados obtenidos en el mismo.

El resultado obtenido ha sido el siguiente:

```
message: Optimization terminated successfully.
success: True
 status: 1
    fun: 52.81120901157566
      x: [ 7.624e-01  4.636e-01]
   nfev: 30
  maxcv: 0.0
```

Este mensaje es devuelto por el optimizador clásico scipy.optimize.minimize y significa que se han encontrado los parámetros $\gamma = 0{,}7624$ y $\beta = 0{,}4636$, con los que la función *execute_circuit* devuelve un valor de $52{,}81$. Esto está muy lejos de ser una media óptima, ya que esta se daría al obtener en todos los casos el camino más corto, lo que daría una media de 11 (el coste de dicho resultado). El otro valor a remarcar, *nfev*=30, se refiere al número de ejecuciones de la función *execute_circuit* realizadas.

Los parámetros obtenidos, ejecutados en un simulador, dan el siguiente resultado:

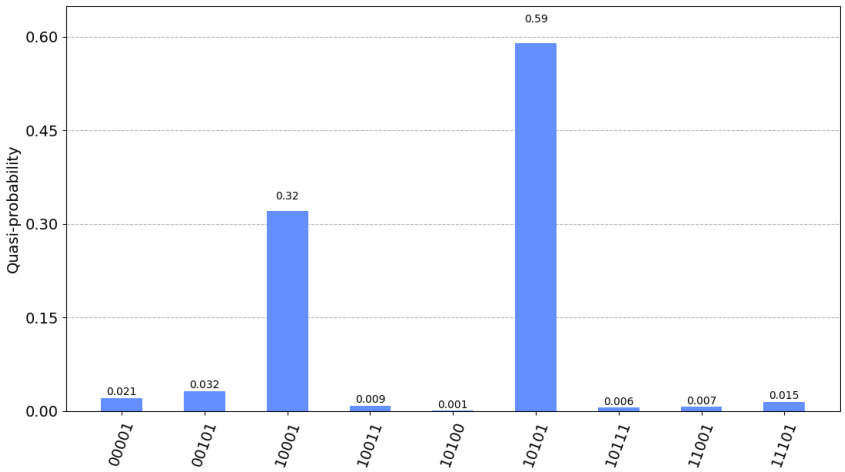


Figura 5.6: Ejecución de γ, β en simulador

Esto indica que el motivo de obtener un resultado inesperadamente alto de la función se debe a la presencia de un camino, *10001*, de coste elevado obtenido un gran número de veces. Este camino es costoso, con valor de 63, debido a que rompe varias restricciones de la función de coste.

Estos mismos parámetros resultados de la ejecución en *ibm_nairobi* han sido ejecutados en el ordenador de *ibm_lagos*, dando los resultados de la *figura 5.7*.

Comparando las *figuras 5.6* y *5.7* se aprecia la diferencia de ejecutar unos mismos parámetros en un simulador y un ordenador cuántico real. Aunque los resultados sean similares, el número de muestras en las que se obtuvo la cadena de bits correspondiente con el camino óptimo disminuye notablemente en la ejecución real, debido al ruido presente en toda ejecución de un circuito cuántico.

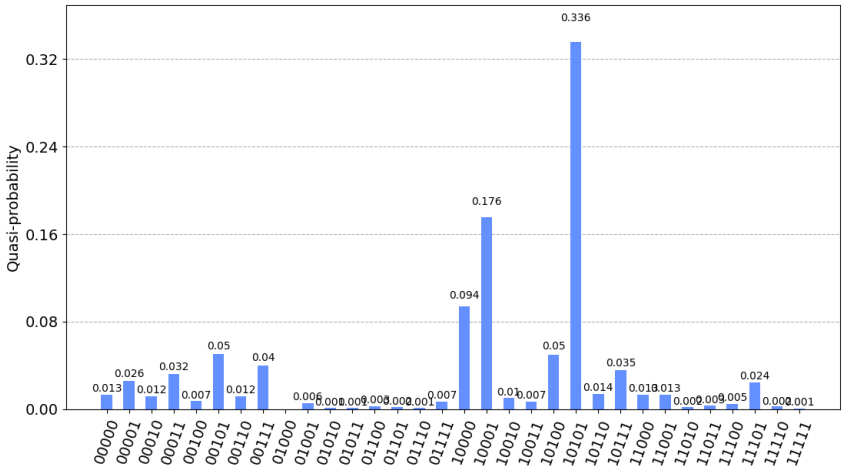


Figura 5.7: Ejecución de γ, β en *ibm_lagos*

5.2.3. Resultados de D-Wave

Con respecto a los resultados de aplicar Quantum Annealing utilizando los sistemas de D-Wave se ha obtenido el resultado de la *tabla 5.7*

Camino	Energía	Muestras
10101 (**Óptimo**)	11	348
10010	12	373
01001	12	294
00000	54	4
10000	58	1
00001	59	1
10100	60	1
00101	61	1
01010	69	1

Tabla 5.7: Resultados de la ejecución del camino más corto en grafo de 4 nodos en D-Wave

La energía se refiere al coste de dicho camino de acuerdo con la función de coste utilizada. Se puede ver cómo, aunque se encuentre el camino óptimo un menor número de veces que en QAOA, existe una mayor coherencia en los resultados. Esto es así porque los caminos que han salido en una mayor cantidad de muestras son los que tienen menor energía, mientras en las ejecuciones de Qiskit se pueden ver ejemplos, como es el camino 00111 en la *figura 5.2*, que han aparecido en una gran cantidad de muestras pero también tiene una energía elevada. [1]

[1] Con la formulación del problema dada, la energía del camino "00111" es 150. Esto se debe a que se rompen varias restricciones presentes en la función de coste. Este valor puede variar dependiendo del multiplicador de Lagrange empleado y las restricciones utilizadas.

5.2.4. Resultados con librería de QAOA

En la siguiente sección se muestran unas estadísticas calculadas en la implementación de QAOA de la librería de Qiskit, *QAOAAnsatz*. Se utilizará para comparar con la implementación propia de QAOA, para así poder asegurarse de que el funcionamiento es el esperado. La función mencionada, *QAOAAnsatz*, construye un circuito a partir de un hamiltoniano, que en este caso será equivalente al operador *C* correspondiente al **problem hamiltonian**.

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	1.0 %	5.54 %
p = 2	5.0 %	4.58 %
p = 3	47.0 %	12.36 %
p = 4	1.0 %	6.91 %
p = 5	5.0 %	6.34 %

Tabla 5.8: Resultados utilizando *QAOAAnsatz* con el problema de la *figura 4.4*

Comparando la tabla generada con la *tabla 5.5* se puede ver que son muy similares. Esto garantiza que la implementación del algoritmo desarrollada en la *sección 4.2* es correcta. También, se apoya la hipótesis de que los cálculos seguidos en el artículo del que se ha obtenido el problema (artículo [5]) son incorrectos (resultados de la tabla 5.4).

5.3. Camino más corto para estudiar la variación con el número de capas

Dadas las incoherencias encontradas en los resultados de la sección anterior, se aplica QAOA al problema descrito en el *artículo [8]*, con el fin de seguir analizando el comportamiento del algoritmo al aumentar el número de capas.

Este problema consiste, de nuevo, en obtener el camino más corto para un grafo, en este caso el de la *figura 4.7*.

5.3.1. Resultados con QAOA

A diferencia de las ejecuciones del grafo correspondiente a la *tabla 5.4*, aquí se ve una clara mejora con el aumento del número de capas.

Además, para el caso $p = 3$ se aprecia una distancia mucho mayor a la habitual entre las estadísticas **NA/TE** y **MM/TE**. A efectos prácticos se toma como medida de fiabilidad **NA/TE**, ya que significa

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	0.6 %	5.9 %
p = 2	30.7 %	16.9 %
p = 3	93.8 %	26.0 %
p = 4	66.9 %	39.4 %
p = 5	1.6 %	15.0 %
p = 6	81.0 %	32.9 %
p = 7	36.5 %	26.4 %
p = 8	64.2 %	32.8 %

Tabla 5.9: Resultados de la ejecución de QAOA del tercer grafo. P=20. Modificadores del paper (omitir $2 * \gamma$)

que se ha encontrado el camino óptimo el 93.8 % de las ejecuciones realizadas.

Función gamma

El resultado de la función gamma es el siguiente:

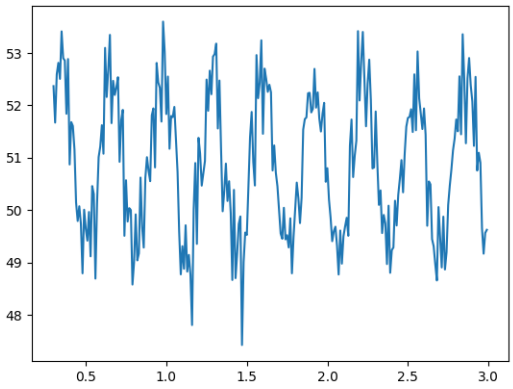


Figura 5.8: Función gamma del tercer grafo. P=20

A diferencia de las funciones gamma de los problemas de MAX-CUT y el camino más corto en grafo de 5 aristas, en esta gráfica se ve mucho más ruido.

Esto se compara también con la gráfica resultante de ejecutar este mismo problema utilizando, para los operadores lineales del circuito, los ángulos obtenidos teóricamente (en lugar de basarse en los obtenidos del artículo).

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	0.1 %	9.79 %
p = 2	10.0 %	20.20 %
p = 3	38.1 %	19.47 %
p = 4	0.7 %	27.33 %
p = 5	40.6 %	24.29 %
p = 6	30.7 %	29.29 %
p = 7	49.3 %	24.60 %
p = 8	71.4 %	36.34 %

Tabla 5.10: Resultados de la ejecución de QAOA del tercer grafo. Ángulos teóricos

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	0 %	7.43 %
p = 2	40.3 %	14.69 %
p = 3	62.0 %	21.9 %

Tabla 5.11: Resultados de la ejecución de QAOA del tercer grafo. P=40

Camino	**Energía**	**Muestras**
1010 (**Óptimo**)	7	776
0101	12	247
0010	46	1

Tabla 5.12: Resultados de la ejecución del tercer grafo en D-Wave

5.3.2. Resultados de D-Wave

Al igual que en anteriores ejecuciones utilizando Quantum Annealing, se mantiene la coherencia en los resultados. El resultado obtenido un mayor número de veces es el camino óptimo y el siguiente camino más producido es el otro único camino válido (sin romper ninguna restricción).

5.3.3. Resultados con librería de QAOA

Las estadísticas ejecutadas con la función *QAOAAnsatz* de Qiskit, al igual que en las secciones anteriores, muestran un comportamiento igual al obtenido en la implementación propia de QAOA (*tabla 5.10*).

nº Capas	**NA/TE** [1.]	**MM/TE** [2.]
p = 1	0.2 %	9.86 %
p = 2	1.2 %	20.01 %
p = 3	40.6 %	19.62 %
p = 4	0.5 %	27.51 %
p = 5	42.7 %	24.88 %

Tabla 5.13: Resultados utilizando *QAOAAnsatz* con el problema de la *figura 4.7*

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] S. Hadfield, Z. Wang, B. O’Gorman, E. Rieffel, D. Venturelli, and R. Biswas, “From the quantum approximate optimization algorithm to a quantum alternating operator ansatz,” *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019. [Online]. Available: http://dx.doi.org/10.3390/a12020034

[5] 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.

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

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

[8] Z. Fan, J. Xu, G. Shu, X. Ding, H. Lian, and Z. Shan, “Solving the shortest path problem with qaoa,” *SPIN*, vol. 13, 12 2022.

Apéndices

A

Ejemplo de aplicación de operadores de QAOA

En esta sección se muestra un ejemplo práctico simple, para facilitar la comprensión de la función de los dos operadores de QAOA.

A.1. Ejemplo de modificación de fases

En un espacio de Hilbert de 2^2 dimensiones, dado un hamiltoniano H definido de la siguiente forma:

$$H = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} = 3 * |00\rangle \langle 00| + 1 * |01\rangle \langle 01| + 2 * |10\rangle \langle 10| + 4 * |11\rangle \langle 11|$$

Como H es diagonal en la base computacional se puede calcular $U(H,\gamma)$ como se explica en la *sección B.1.2*

El estado de mínima energía de H sería $\min_{x\in\{0,1\}^2} \langle x| H |x\rangle$, por lo que el estado es $|x\rangle = |01\rangle$, con energía 1. De esta forma, el **problem hamiltonian** sería $U(H,\gamma) = e^{-i\gamma H}$. Tomando un valor de $\gamma = 0{,}1$, el resultado de aplicarlo a un estado en superposición equiprobable $|\psi\rangle$ es el siguiente:

$$|\psi\rangle = \frac{1}{2} * (|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$\gamma = 0{,}1$$

$$U(H,\gamma)|\psi\rangle = \begin{bmatrix} e^{-i*0{,}1*3} & 0 & 0 & 0 \\ 0 & e^{-i*0{,}1*1} & 0 & 0 \\ 0 & 0 & e^{-i*0{,}1*2} & 0 \\ 0 & 0 & 0 & e^{-i*0{,}1*4} \end{bmatrix} * \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} e^{-i*0{,}3} \\ e^{-i*0{,}1} \\ e^{-i*0{,}2} \\ e^{-i*0{,}4} \end{bmatrix}$$

Como se puede ver las fases relativas en $|\psi\rangle$ son de 0 para todos los valores, mientras que para $U(H,\gamma)|\psi\rangle$ el estado de mayor fase relativa es $|01\rangle$.

También es importante recalcar que las probabilidades de medición de los valores no han sido modificadas. Es el caso de 01:

$$P_\psi(01) = \langle\psi| M_{01}^\dagger M_{01} |\psi\rangle = \langle\psi| M_{01} |\psi\rangle = |\frac{1}{2} * e^{-0,1i}|^2 = \frac{1}{4}$$

Donde $M_{01} = \langle 01|01\rangle$ es un operador de medición para el valor 01. Se han tomado las definiciones del postulado de la medición explicado en [7].

A.2. Ejemplo de modificación de probabilidades

Tomando el resultado de la *sección A.1* se puede ver cómo el operador **mixer hamiltonian** aumenta la probabilidad de medición de 01.

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\beta = 0{,}1$$

$$B = \sigma^x \otimes I + I \otimes \sigma^x = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$U(B,\beta) = e^{-i\beta B} = \begin{bmatrix} \cos(\beta)^2 & -i * \sin(\beta)\cos(\beta) & -i * \sin(\beta)\cos(\beta) & -\sin(\beta)^2 \\ -i * \sin(\beta)\cos(\beta) & \cos(\beta)^2 & -\sin(\beta)^2 & -i * \sin(\beta)\cos(\beta) \\ -i * \sin(\beta)\cos(\beta) & -\sin(\beta)^2 & \cos(\beta)^2 & -i * \sin(\beta)\cos(\beta) \\ -\sin(\beta)^2 & -i * \sin(\beta)\cos(\beta) & -i * \sin(\beta)\cos(\beta) & \cos(\beta)^2 \end{bmatrix} =$$

$$= \begin{bmatrix} 0{,}99 & -0{,}0993j & -0{,}0993j & -0{,}01 \\ -0{,}0993j & 0{,}99 & -0{,}01 & -0{,}0993j \\ -0{,}0993j & -0{,}01 & 0{,}99 & -0{,}0993j \\ -0{,}01 & -0{,}0993j & -0{,}0993j & 0{,}99 \end{bmatrix}$$

$$U(B,\beta) * \frac{1}{2} \begin{bmatrix} e^{-i*0,3} \\ e^{-i*0,1} \\ e^{-i*0,2} \\ e^{-i*0,4} \end{bmatrix} = \begin{bmatrix} 0{,}4535 - 0{,}2424i \\ 0{,}4536 - 0{,}1416i \\ 0{,}4462 - 0{,}191i \\ 0{,}4364 - 0{,}2894i \end{bmatrix}$$

Donde σ^x es conocida como puerta Pauli-X, que equivale a una rotación en el eje X de la esfera de Bloch.

B

Desarrollo de operaciones

B.1. Exponente de una matriz

En esta sección se muestra el desarrollo necesario para operar sobre operador $e^{i*\theta*A}$, dados dos casos:

B.1.1. A*A = I

- **Definiciones:**

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, A^2 = I$$

- **Desarrollo:**

$$\begin{aligned} \exp(i*\theta*A) &= \sum_{n=0}^{\infty} \frac{(i*\theta)^n * A^n}{n!} = \\ &= (1*A^0 - \frac{\theta^2*A^2}{2!} + \frac{\theta^4*A^4}{4!} - \ldots) + i*(\theta*A^1 - \frac{\theta^3*A^3}{3!} + \frac{\theta^5*A^5}{5!} - \ldots) = \\ &= I*(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \ldots) + i*A*(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \ldots) = \cos(\theta)*I + i*\sin(\theta)*A \end{aligned}$$

B.1.2. A es diagonalizable

Un operador A sobre un espacio V es diagonalizable si es diagonal con respecto a alguna base ortonormal de V. La representación diagonal de A tiene la siguiente forma:

$$A = \sum_{\lambda} \lambda \left|\lambda\right\rangle \left\langle\lambda\right|$$

Donde λ son los autovalores de A y $|\lambda\rangle$ sus respectivos autovectores.

Dada una función f definida sobre los números complejos se puede definir una función correspondiente sobre A utilizando su representación diagonal:

$$f(A) = \sum_{\lambda} f(\lambda) |\lambda\rangle \langle\lambda|$$

Concretamente, para $e^{i\theta A}$:

$$\exp(-i\theta A) = \sum_{\lambda} \exp(\lambda) |\lambda\rangle \langle\lambda|$$

C

Paso de función clásica a hamiltoniano

Esta sección está destinada a demostrar el paso de una función clásica $f(x)$ en su versión QUBO a su hamiltoniano correspondiente C.

- Definiciones:
  - $f(x)$: Función clásica en formato QUBO (*Quadratic Unconstrained Binary Optimization*), donde $x \in \{0,1\}^n$.
  - $C(z)$: Función clásica en formato Ising equivalente a $f(x)$, pero donde $z \in \{-1,1\}^n$.
  - σ_z: Operador Pauli-Z.

$$\sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Concretamente se pretende demostrar que sustituyendo las variables z_i en $C(z)$ por puertas σ^z en el qubit *i* se puede obtener el operador C tal que se cumpla:

$$f(x) = \langle x| C |x\rangle, \forall x \in \{0,1\}^n$$

Esto es equivalente a decir que el operador C es diagonal, con valores correspondientes a la función clásica $f(x)$:

$$C|x\rangle = \begin{bmatrix} C(0\ldots00) & 0 & 0 & 0 \\ 0 & C(0\ldots01) & 0 & 0 \\ \vdots & & \ddots & \\ 0 & 0 & 0 & C(1\ldots11) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = f(x)|x\rangle \qquad \forall x \in \{0,1\}^n$$

Los autovalores de σ^z son $-1, +1$, y sus respectivos autovectores la base computacional:

$$\sigma^z |0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = +1 * |0\rangle$$
$$\sigma^z |1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -1 * |1\rangle$$

Esto puede ser generalizado como:

$$\sigma^z |x\rangle = (-1)^x |x\rangle, x \in \{0, 1\}$$

En las funciones de coste utilizadas existen dos términos distintos en los que pueden aparecer variables z_i:

- $c * z_i$

  Un operador σ^z actuando en el qubit *i* se desarrolla de esta forma:

  $$\sigma_i^z |x_0 \dots x_{n-1}\rangle = I \otimes \dots \otimes \sigma_i^z \otimes \dots \otimes I |x_0 \dots x_{n-1}\rangle = (-1)^{x_i} |x_0 \dots x_{n-1}\rangle \quad x_i \in \{0, 1\}, i \in 1, \dots n \tag{C.1}$$

- $c * z_j * z_k, j \neq k$

  De forma similar al caso anterior, dos operadores σ^z actuando en dos qubits *i*, *j* se desarrollan de esta forma:

  $$\begin{aligned} \sigma_i^z \sigma_j^z |x_0 \dots x_{n-1}\rangle &= I \otimes \dots \otimes \sigma_i^z \otimes \sigma_j^z \otimes \dots \otimes I |x_0 \dots x_{n-1}\rangle = \\ &= (-1)^{x_i}(-1)^{x_j} |x_0 \dots x_{n-1}\rangle \qquad x_i \in \{0, 1\}, i \in 1, \dots n \end{aligned} \tag{C.2}$$

- $c * z_i * z_i = c$

  Porque $z_i \in \{-1, 1\} \forall i \in 0, \dots n-1$

Esto permite que se realice un cambio de variable a la función $C(z)$ tal que $z_i \to (-1)^{x_i}$, donde se cumple que $x_i \in \{0, 1\}$ teniendo en cuenta que $z_i \in \{-1, 1\}$.

Dado este cambio de variable, a su vez se podrán utilizar las ecuaciones *C.1* y *C.2* para sustituir esos términos $(-1)^{x_i}$ por σ_i^z


UAM
UNIVERSIDAD AUTONOMA
DE MADRID