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 J. Gómez Arribas

junio 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

La computación cuántica, que aplica principios de la mecánica cuántica a la computación, promete ser una herramienta para tratar problemas que, por la gran cantidad de datos o por la dimensión del espacio de búsqueda, no son resolubles actualmente con los recursos disponibles en computación clásica, a la vez que se consigue disminuir el tiempo de ejecución.

Este TFG se desarrolla en el contexto de los problemas de optimización y estudia la aplicación de algoritmos en los que se busca el extremo de una función de coste. Se ha evaluado la utilización de dos tecnologías cuánticas diferentes para tratar de mejorar las soluciones clásicas.

Concretamente, se han utilizado tres problemas de optimización para validar el funcionamiento híbrido del algoritmo QAOA (*Quantum Approximate Optimization Algorithm*) que combina computación cuántica, para obtener el resultado óptimo de la función de coste, y computación clásica, para encontrar los parámetros que optimizan el funcionamiento del circuito que calcula dicha función de coste.

Se han comparado los resultados obtenidos con QAOA en ordenadores cuánticos generalistas, con los obtenidos por QA (*Quantum Annealing*) en sistemas cuánticos D-Wave, específicos para problemas de optimización.

Adicionalmente se han realizado análisis estadísticos de los resultados obtenidos, comparando los valores publicados en dos artículos internacionales con los que se obtienen en este TFG y con los que resultan de utilizar métodos de librería disponibles en el entorno de programación Qiskit. Para esto, se ha realizado la implementación completa del algoritmo QAOA.

Este TFG ha sido desarrollado con dos objetivos distintos:

- Un objetivo formativo y didáctico, con una explicación completa de cómo y por qué funciona el algoritmo QAOA y cómo se compara con QA.
- La evaluación de la aplicación práctica de estos algoritmos con las tecnologías cuánticas actuales, realizando un análisis estadístico de la validez de los resultados obtenidos.

Palabras clave

Computación cuántica, Problemas QUBO, Ising, QAOA, Quantum annealing, D-Wave, Qiskit, optimización cuadrática, Camino más corto, Max cut

Abstract

The paradigm of quantum computation, which applies principles of quantum mechanics to computation, promises to be a tool to manage problems that, due to the big amount of data or the extent of the search space, are not solvable at the moment with the available resources in classic computation, at the same time that a reduction of the execution time is achieved.

This end-of-degree project is developed in the context of optimization problems and studies the application of algorithms in which the maximum or minimum of a cost function is searched. Trying to improve the available classical solutions, two quantum technologies have been used.

Specifically, three optimization problems have been used to validate the hybrid functioning of QAOA (*Quantum Approximate Optimization Algorithm*), that combines quantum computing, to obtain the optimal result of the cost function, and classical computing, to find the parameters that optimize the functioning of the circuit that evaluates said cost function.

The result obtained with QAOA in generic quantum computers has been compared with the results obtained by QA (*Quantum Annealing*) in D-Wave's quantum systems, specifically designed to solve optimization problems.

Additionally, statistical analysis have been carried out on the obtained results, comparing the values published in two international articles with the ones obtained in this work and with the ones resulting from using methods available in the Qiskit programming environment.

This end-of-degree project has been developed with two different objectives:

- A training and didactic objective, with a complete explanation of how and why QAOA works and how it compares with QA.
- The evaluation of the practical application of these algorithms with current quantum technologies, doing a statistical analysis to estimate the validity of the obtained results.

Keywords

Quantum computing, QUBO problems, Ising, QAOA, Quantum annealing, D-Wave, Qiskit, quadratic optimization, Shortest path, Max cut

Índice

Listas

Lista de figuras

Lista de tablas

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 resolver 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 la factorización de un entero en números primos al ser 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 también se proponen varios algoritmos de optimización alternativos que permitirían una mejora en la escalabilidad. Esta superioridad cuántica todavía no ha podido verse realizada, 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.

Para la resolución de problemas de optimización con variables binarias en cuántica se busca que el hamiltoniano que describe el sistema cuántico represente la función de coste del problema que se

quiere optimizar, para que el mínimo de energía del sistema cuántico coincida con el mínimo de la función de coste.

Para esto es necesario que el formato del problema de optimización sea QUBO (Quadratic Unconstrained Binary Optimization), en los que se busca el mínimo global de una función de coste. Además, el problema no puede tener restricciones, por lo que el tamaño del espacio de estados disponibles en el problema es 2^n. La función de coste tiene la forma $f : \{0, 1\}^n \to \mathbb{R}$ para $n \in \mathbb{N}$.

Dos efectos de la mecánica cuántica, la superposición y el entrelazamiento, permiten a las tecnologías basadas en computación cuántica resolver problemas que están más allá del alcance de las computadoras clásicas.
Los computadores cuánticos realizan ciertos tipos de cálculos exponencialmente más rápido que las computadoras clásicas, lo que podría conducir a mejoras significativas en la precisión y eficiencia de los algoritmos de optimización. Esto es debido a que es posible realizar cálculos en paralelo y explorar una mayor cantidad de escenarios posibles.

Hoy en día existen dos categorías principales de tecnologías basadas en computación cuántica:

- Computadoras cuánticas basadas en puertas universales.
- Optimizadores cuánticos basados en *Quantum Annealing* (recocido cuántico).

IBM-Q y Google ofrecen computadores cuánticos basados en puertas universales. Estos sistemas tienen la propiedad de ser Turing-completos, en los que se interconectan un conjunto de puertas universales.
En una aplicación práctica se codifican los datos de entrada a través de los estados iniciales de los qubits; se llevan a un estado de superposición; se aplica un algoritmo en todos los estados, mediante un circuito cuántico implementado a través de una serie de puertas que se aplican a los qubits; y finalmente se mide uno o varios qubits, colapsando el estado cuántico. Esta medición tiene un resultado probabilístico y la finalidad es que el estado solución quede reflejado con el de mayor probabilidad y sea interpretable como la solución del problema.

Por el contrario, los sistemas D-Wave se fabrican según la teoría del recocido cuántico (*quantum annealing*). Este sistema es un dispositivo cuya función es resolver problemas de tipo QUBO. La formulación QUBO se ha convertido en un formato de entrada estándar para los *quantum annealers*, al que se convierten los problemas de aplicaciones prácticas de interés.
El cálculo se realiza mediante un algoritmo heurístico de optimización cuántica. Las ideas de la computación cuántica adiabática (Farhi *et al.*, 2000 [4]) son similares al algoritmo de recocido simulado clásico (simulated annealing), donde las fluctuaciones térmicas permiten que el sistema salte entre diferentes mínimos locales en la función de coste.
En el *quantum annealing*, las transiciones del sistema son impulsadas por el efecto túnel cuántico. Una propiedad que todas las QPU (*Quantum Processing Units*) tienen es que la disposición física de los

qubits es fija. La arquitectura de hardware de los *quantum annealers*, como el de D-Wave, emplea una red de qubits y acopladores dispuestos para mapear de manera eficiente los problemas de optimización al hardware cuántico. Este diseño permite la traducción efectiva de problemas de optimización a operadores hamiltonianos (definidos en la *sección A.3*) y la posterior búsqueda del mínimo energético de estos mediante la aplicación de un proceso de recocido.

Las computadoras cuánticas basadas en puertas universales también se pueden usar para el recocido, aunque de manera menos eficiente. Los *quantum annealers*, como el de D-Wave, tienen un diseño específico orientado a resolver problemas de optimización. Esta es también la razón por la que D-Wave tiene máquinas de más de miles de qubits (King *et al.*, 2022 [5]) a día de hoy, mientras que la computadora cuántica universal de última generación tiene solo unos cientos de qubits [6].

En este trabajo se tratarán sendos algoritmos de las dos tecnologías de computación cuántica mencionadas:

- El primero, **Quantum Approximate Optimization Algorithm** (QAOA) [3], es un algoritmo híbrido (con pre y post-procesamiento en un ordenador clásico) que es aplicado en computadoras que siguen el modelo basado en puertas. La forma de implementarlo será al más bajo nivel, es decir, a nivel de puertas cuánticas, que es el equivalente a puertas lógicas en álgebra booleana, utilizando las librerías de Qiskit (IBM-Q) para su ejecución. En este trabajo será el algoritmo de mayor importancia, ya que el segundo se utilizará como comparación con este. [1]
- El segundo algoritmo, ya introducido, será el de **Quantum Annealing** (QA). Esto se hará mediante los sistemas de la empresa D-Wave y se le dará un uso de caja negra, es decir, no se va a realizar una implementación del mismo, sino que se utilizará a través de la interfaz proporcionada por estos sistemas.

1.2. Objetivos

El objetivo de este trabajo es estudiar la implementación en detalle de QAOA, utilizando los resultados obtenidos con el algoritmo de QA como comparación. El motivo por el que se han escogido estos algoritmos es porque, a pesar de las diferencias entre ellos, el tipo de problemas a resolver es el mismo, a saber, problemas tipo QUBO. Además, tanto en QAOA como en QA, para alcanzar el mínimo de la función de coste clásica f se debe hacer que el estado fundamental (*sección A.4*) del hamiltoniano que describe el sistema cuántico contenga ese mínimo de f.

[1] Existe un segundo significado de QAOA, Quantum Alternating Operator Ansatz [7], 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].

2

Estado del arte

El punto de partida es el trabajo desarrollado por Urgelles *et al.* (2022) [8]. En este se exploran algoritmos cuánticos para la optimización del enrutamiento de paquetes en redes de comunicaciones, con el objetivo de aplicarlo a tecnologías 6G. Concretamente, se explora el rendimiento de QAOA al resolver el problema del camino más corto en grafos dirigidos pesados.
Para esto se resuelve una versión de este problema en un grafo simple, mostrando los resultados obtenidos por QAOA, y después se aplica el algoritmo a un grafo multi-objetivo. En este segundo caso se parte de un grafo donde las aristas tienen dos o más pesos distintos, relacionados con distintas propiedades que quieren verse minimizadas. Por esto, no siempre es posible obtener un mismo camino que constituya el mínimo global para todas estas propiedades, por lo que en su defecto se busca el camino que constituya un óptimo de Pareto y, por lo tanto, su coste para cada propiedad sea suficientemente bajo.
Como el objetivo de este trabajo es la comprensión y análisis de resultados de QAOA, la aplicación del mismo a grafos multi-objetivo no será explorada. Esto es porque se basa en realizar sucesivas ejecuciones de QAOA, variando la entrada de ejecuciones posteriores basándose en las anteriores, sin modificar el funcionamiento del algoritmo en sí.

Como los algoritmos a tratar pertenecen a un paradigma distinto a la computación clásica, es conveniente describir las clases de complejidad necesarias dentro de la computación cuántica, así como su relación con las clases tradicionales.

Además, los problemas de optimización a resolver con QAOA y QA son de tipo QUBO, por lo que es necesario definir qué cualidades posee un problema de este tipo.

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. El segundo son 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 si-

tuaciones 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 tienen probabilidad no nula de no dar el resultado correcto.

Por supuesto, solo 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. Tanto QAOA como QA caen en este segundo conjunto de problemas, *BQP*.

2.2. Problemas de optimización combinatoria

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

$$f(x) = \sum_{\alpha=1}^{m} f_\alpha(x) \tag{2.1}$$

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

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

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 3.1*.

Una estrategia infalible para resolver un problema de este tipo es aplicar la fuerza bruta, evaluando todas las cadenas de bits posibles para encontrar la de menor coste. Este es un método muy costoso, ya que la complejidad crece de forma exponencial con n. De la misma forma, si se precomputaran todos los valores posibles de la función de coste la memoria aumentaría de forma exponencial, ya que para cadenas de n bits se necesitarían almacenar 2^n valores distintos.
Por naturaleza, todo circuito cuántico de n qubits tiene una matriz asociada de tamaño $2^n \times 2^n$ denominada el hamiltoniano, que describe la energía del sistema (*sección A.3*). De esta forma si se construye un circuito cuyo hamiltoniano tenga en su diagonal los valores de una función de coste clásica se pueden lograr almacenar los 2^n valores distintos con tan solo n qubits. De esta forma es posible definir de forma implícita la función de coste del problema y revertir ese incremento exponencial de precomputar los valores.
Esta estrategia es seguida tanto por QAOA como por QA.

2.3. QAOA

La idea fundamental de QAOA es partir de un hamiltoniano C con los valores de una función de coste $f(x)$ en su diagonal (sección A.5) es decir:

$$\forall x \in \{0,1\}^n, f(x) = \langle x| C |x\rangle \tag{2.2}$$

De esta forma al medir el estado fundamental (sección A.4) del sistema, es decir, el estado de mínima energía, se obtendrá el valor x para el que $f(x)$ es mínimo.
No obstante el operador C no es necesariamente unitario, la cual es una propiedad necesaria de un operador para ser implementado en un sistema cuántico [1]. Para ello se debe utilizar el operador e^{iC},

[1] Todo operador lineal en un sistema cuántico debe ser unitario para que el sistema sea coherente y cumpla que $\sum_{|x\rangle} P(|x\rangle) = 1$, donde $|x\rangle$ itera sobre los vectores de la base computacional del sistema y $P(|x\rangle)$ se refiere a la probabilidad de medir x sobre dicha base.

que es unitario para todo operador hermítico C.

Esta expresión surge de la resolución de la ecuación de Schrödinger (*ecuación 2.3*), la cual es una ecuación diferencial que describe la evolución de un estado cuántico a partir de su hamiltoniano.

$$i\frac{d\left|\psi(t)\right\rangle}{dt} = H\left|\psi(t)\right\rangle \rightarrow \left|\psi(t)\right\rangle = e^{-iHt}\left|\psi(0)\right\rangle \tag{2.3}$$

Esta expresión será utilizada en la construcción de los operadores de QAOA, ya que si en lugar de utilizar el tiempo se varía el estado a razón de un ángulo α se obtiene una simulación de la evolución del estado.

Siguiendo con la notación del artículo original [3], en el algoritmo QAOA se utilizan dos operadores unitarios:

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

  Dado un estado con una serie de valores posibles (correspondientes a los estados de la base computacional) este operador separa las fases relativas (*sección A.7*) de dichos valores. De esta forma al aumentar γ la fase de los valores con menor coste aumenta y la de los valores con mayor coste disminuye. Esto no tiene ningún efecto en los resultados, 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 al variar β los valores con menor fase son más probables de observar que los valores con mayor fase.

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

Como se indicará con más detalle en la *sección 3.2.2*, tanto γ como β representan ángulos que describen rotaciones en torno a los ejes del sistema. Es por ello que tienen un carácter periódico y si se aumentan lo suficiente pueden provocar un efecto de desbordamiento.

La función de los operadores hamiltonianos es fundamental para el funcionamiento de QAOA, pero una sola aplicación de estos sobre un estado puede no dar un buen resultado. Por este motivo se repiten sucesivamente los dos operadores variando β y γ para lograr mejores resultados, donde un par $U(B,\beta_i)U(C,\gamma_i)$ constituirá la capa i-ésima del algoritmo. QAOA será definido con un número de capas p.

2.4. Quantum Annealing

El nombre de *quantum annealing* viene de su relación con el algoritmo de *simulated annealing*. En ambos se busca el mínimo de una función de coste asociada a un problema de optimización combinatoria.
En el algoritmo de *simulated annealing* se busca el mínimo partiendo de un estado inicial y moviéndose sucesivamente a estados vecinos en el espacio de estados. Se parte de una temperatura alta al principio, lo que hace que el radio de aceptación de estados vecinos sea mayor, y disminuyéndola gradualmente hasta encontrar el mínimo. Este algoritmo tiene la posibilidad de finalizar en un estado que constituya un mínimo local no global de la función de coste, si la energía inicial no es lo suficientemente fuerte.
De forma similar, QA también parte de un estado inicial y lo evoluciona hasta encontrar el estado de mínima energía.

La idea general de QA es la siguiente:
Sea un sistema cuántico dado por un hamiltoniano dependiente del tiempo $H(t)$:

$$H(t) = (1 - \frac{t}{T})H_s + \frac{t}{T}H_c \tag{2.4}$$

T representa el tiempo total y H_s es un hamiltoniano con un estado fundamental $|\psi_s\rangle$ conocido. H_c es un hamiltoniano equivalente al operador C de QAOA cuyo estado fundamental $|\psi_c\rangle$ se quiere conocer ya que, al igual que C, es equivalente al estado de menor valor en la función de coste clásica. De acuerdo con el teorema adiabático [9] si el estado $|\psi(t=0)\rangle$ del sistema es el estado fundamental de H_s y se avanza el tiempo lentamente el sistema se mantendrá en el estado fundamental de $H(t)$, de tal forma que $|\psi(t=T)\rangle$ será el estado fundamental de H_c y, por lo tanto, el resultado de la función de coste.

Como ya se ha visto en la sección anterior, la evolución de un estado viene dada por la ecuación de Schrödinger(*ecuación 2.3*), de tal forma que para QA el hamiltoniano $H(t)$ del sistema afecta de la siguiente forma:

$$|\psi_c\rangle = e^{-iH(t)t}\,|\psi_s\rangle \tag{2.5}$$

Una similitud esencial entre QAOA y QA es que ambos siguen esta noción de evolucionar de un estado inicial al estado de mínima energía del sistema, que en ambos casos corresponde con el mínimo de la función de coste, solo que donde en QA varía el tiempo (t) en QAOA varían los ángulos β y γ.
La diferencia esencial es que aunque en QAOA sería posible utilizar los ángulos β y γ para que el estado siguiera una evolución de la misma forma que QA, es preferible variar los ángulos utilizando una heurística clásica, como el descenso por gradiente o equivalentes.

3

Diseño

Antes de explicar el diseño concreto de cada problema y cómo se implementa en cada tecnología cuántica, en la *fig. 3.1* se indica el flujo de trabajo realizado en este TFG. Marcado en color verde se muestran las aportaciones de este trabajo, mientras que en azul se muestran recursos obtenidos de otras fuentes.

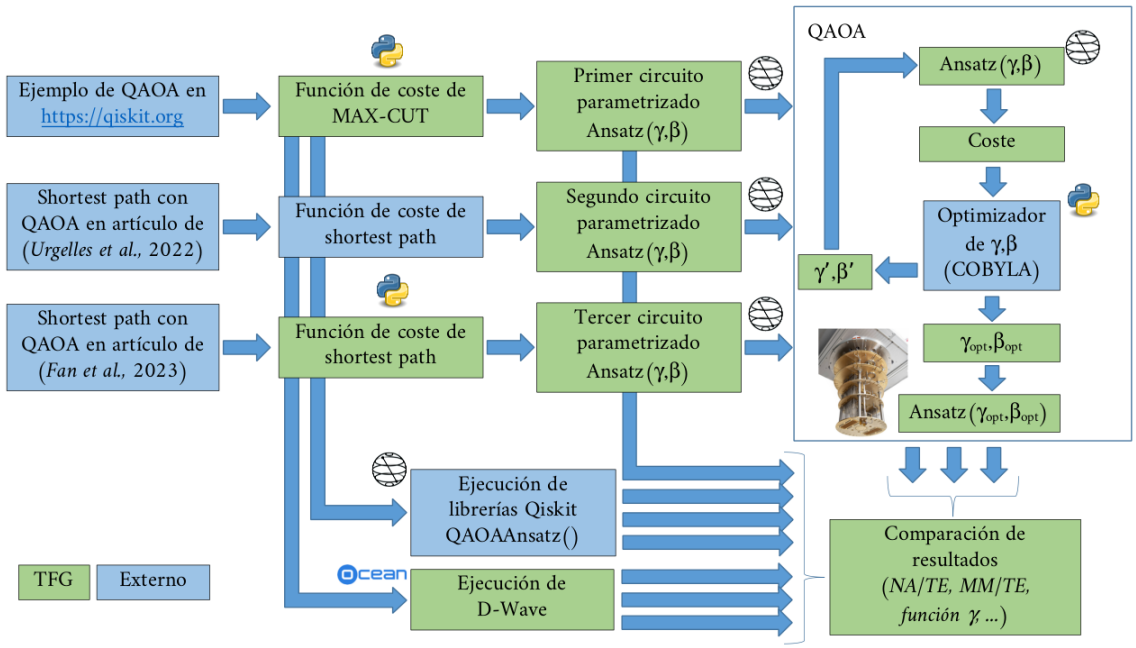


Figura 3.1: Flujo de trabajo

Los dos algoritmos del trabajo, QAOA y QA, han sido ejecutados en tres problemas distintos:

- Max-cut en grafo de 4 aristas:

  Este problema, obtenido en la página web tutorial de Qiskit [10], ha sido elegido como el primero a resolver para asegurar una correcta implementación de QAOA, debido a su simplicidad. Además, en la fuente se aporta el código utilizado, por lo que se puede comparar completamente con el código propio. Los problemas max-cut son un estándar en la resolución de QAOA.

- Camino más corto en grafo de 4 nodos:

Tras lograr unos resultados satisfactorios en el problema anterior, se resuelve el problema del artículo de Urgelles *et al.* (2022) [8]. En este caso se aumenta la complejidad con respecto a la resolución del problema de max-cut. Para comparar con los resultados obtenidos, en el artículo se incluyen un circuito de prueba y estadísticas de QAOA ejecutadas para una y varias capas.

- Camino más corto para estudiar la variación con el número de capas:

  Tras obtener unos resultados incongruentes en el problema anterior al aumentar el número de capas del algoritmo se decide evaluar otra instancia del problema del camino más corto, del artículo de Fan *et al.* (2023) [11], para analizar el comportamiento del algoritmo para $p > 1$.

Para estos tres casos se ha seguido el mismo proceso de análisis.
Se ha definido una función de coste clásica apta para el problema. Después se ha utilizado esta función para calcular el circuito parametrizado ansatz. Este se ha aplicado en una implementación realizada en Python del algoritmo haciendo uso de la librería de Qiskit, que permite la construcción y posterior ejecución de circuitos cuánticos a nivel de puertas.
Por otra parte la función de coste también ha sido utilizada para la ejecución de QA en máquinas de D-Wave y para la ejecución de la función *Qiskit.QAOAAnsatz()*, la cual es una implementación de caja negra de QAOA empleada para su comparación con el código escrito.
El análisis de los resultados ha sido realizado utilizando como datos los proporcionados por las respectivas fuentes de los problemas, además de los resultados de las ejecuciones de QA, QAOA y *Qiskit.QAOAAnsatz()*, así como los circuitos generados por las dos últimas. Las métricas empleadas para esta comparación han sido definidas en el capítulo 5.

Para llevar este proceso a cabo es necesario comprender antes cómo funciona QAOA y cómo se puede formular un problema de optimización con restricciones para que sea resuelto por este algoritmo.

3.1. Traducción de problemas de optimización combinatoria a QUBO

Como ya se ha comentado anteriormente, 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: $f_2(x) = f(x) + P * (A(x) - B(x))^2$, donde

$$P * (A(x) - B(x))^2 \begin{cases} = 0 & \text{si se cumple la restricción} \\ > \max_x f(x) & \text{en otro caso} \end{cases} \tag{3.1}$$

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 haga que el valor de la función de coste sea mayor que cualquier otro resultado en el que no se rompan restricciones. De esta forma toda entrada válida en $f(x)$ tendrá el mismo coste en $f_2(x)$, mientras que toda entrada no válida en $f(x)$ tendrá en $f_2(x)$ un coste mayor que cualquier entrada de $f(x)$.
Si todas las restricciones cumplen que $(A(x) - B(x))^2 >= 1$, entonces el modificador de Lagrange queda sujeto a $P > \max_x f(x)$.

3.2. Circuito de QAOA

El sistema cuántico en QAOA se implementa en un espacio de Hilbert (sección A.1) 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 idea general de QAOA se basa en preparar un estado parametrizado $|\psi(\vec{\beta}, \vec{\gamma})\rangle$ 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 los vectores:

$$\vec{\beta} = [\beta_0, \beta_1, \ldots, \beta_{p-1}], \vec{\gamma} = [\gamma_0, \gamma_1, \ldots, \gamma_{p-1}] \tag{3.2}$$

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

En este estado se diferencian un estado inicial ($|\psi_0\rangle$) y dos operadores, mencionados en la *sección 2.3*, el hamiltoniano del problema y el hamiltoniano de mezcla. Estos se combinan de la siguiente forma:

$$|\psi(\vec{\beta}, \vec{\gamma})\rangle = U(B, \beta_{p-1})U(C, \gamma_{p-1})U(B, \beta_{p-2})U(C, \gamma_{p-2}) \ldots U(B, \beta_0)U(C, \gamma_0) |\psi_0\rangle \tag{3.3}$$

El aumento del número de capas (p) mejora el resultado del algoritmo, pero también aumenta la profundidad del circuito, por lo que lo hace más susceptible de sufrir por el ruido de la ejecución.

3.2.1. Estado inicial

El estado inicial del sistema es la superposición equiprobable, definida como

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes n} = H^{\otimes n} |0\rangle^{\otimes n} \tag{3.4}$$

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

cualquiera en $\{0,1\}^n$ al medir el estado sería en cualquier caso $\frac{1}{2^n}$.

3.2.2. Hamiltoniano de mezcla y del problema

Estos dos hamiltonianos son operadores unitarios definidos como:

$$U(B,\beta) = e^{-i\beta B} \tag{3.5}$$

$$U(C,\gamma) = e^{-i\gamma C} \tag{3.6}$$

Como ha sido explicado en la *sección 2.3*, el operador $U(C,\gamma)$ modifica la fase de los estados, haciendo que los de menor energía aumenten en fase y los de mayor energía disminuyan.
También se ha explicado cómo el operador $U(B,\beta)$ realiza rotaciones de tal forma que aumente la amplitud, y por ende la probabilidad, de los estados con mayor fase.

Operador *B*

B se construye a partir de puertas Pauli-X (σ^x) y únicamente depende de la cantidad de qubits del sistema. Se define de la siguiente forma:

$$B = \sum_{i=0}^{n-1} \sigma_i^x \tag{3.7}$$

$$U(B,\beta) = e^{-i\beta B} = \prod_{j=0}^{n-1} e^{-i\beta\sigma_j^x} = \prod_{j=0}^{n-1} Rx_i(2*\beta) \tag{3.8}$$

Donde $Rx(\alpha)$ es un operador que realiza una rotación de α radianes sobre el eje x de la esfera de Bloch.
También, cuando un operador lineal tiene un subíndice se refiere a que es aplicado a ese qubit, por lo que la expresión $Rx_i(\alpha)$ es una puerta $Rx(\alpha)$ aplicada al qubit i-ésimo. Es equivalente a $I \otimes I \otimes \ldots \otimes R_x(\alpha) \otimes \ldots I \otimes I$.

De esta forma, la variable β del hamiltoniano de mezcla representa un ángulo, y por lo tanto es periódico. Esto es importante porque el aumento indefinido de β puede llevar a disminuir la amplitud de los estados con más fase.

Operador *C*

C debe ser un operador diagonal (*sección A.5*) en la base computacional $|x\rangle$, con los valores de la función objetivo. Esto es equivalente a tener los valores $f(0\ldots0), f(0\ldots1), \ldots, f(1\ldots1)$ en la diagonal de C y el resto de elementos con valor 0. Para un vector $|x\rangle$ de la base computacional se cumple:

$$\langle x| C |x\rangle = f(x) \tag{3.9}$$

Para obtener este operador C se parte de una función de coste en su formato QUBO, con la forma $f(x)$, donde $x = x_0x_1 \dots x_{n-1}, x_i \in \{0, 1\}$.
Un primer paso es convertir la función en formato QUBO a su formato Ising [12]. Una función Ising es equivalente a una tipo QUBO, solo que en lugar de aceptar cadenas del conjunto $\{0, 1\}^n$ acepta cadenas de $\{-1, 1\}^n$ para su entrada. Esto es necesario porque, como se verá más adelante, para convertir una función de coste a un hamiltoniano C se deben usar operadores cuyos autovalores correspondan a los valores que pueden tomar las variables del sistema, en este caso -1 y 1. Para realizar este paso se puede realizar un cambio de variable como el siguiente:

$$\begin{aligned} f(x) &\rightarrow C(z) \\ x_i &\rightarrow \frac{1 - z_i}{2} \end{aligned} \tag{3.10}$$

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 Pauli-Z en el qubit i (σ_i^z).
El motivo por el que para muchos algoritmos cuánticos se utiliza el formato Ising es porque los autovalores de la puerta Pauli-Z son -1 y 1, lo que permite este paso entre variables z y operadores σ^z.
La justificación detallada se encuentra en la *sección D* del apéndice.

Como se verá en aplicaciones concretas de QAOA, al igual que el hamiltoniano $U(B, \beta)$ se expresa mediante operadores $Rx(\alpha)$, la forma de expresar el hamiltoniano $U(C, \gamma)$ es mediante puertas $Rz(\alpha)$, cuya definición es:

$$Rz_i(\alpha) = e^{-i\alpha\sigma_i^z} \tag{3.11}$$

De manera análoga a $Rx(\alpha)$, $Rz(\alpha)$ representa una rotación de α radianes sobre el eje Z de la esfera de Bloch.
De esto se concluye que, al igual que en el caso del hamiltoniano de mezcla, el hamiltoniano del problema también tiene una entrada que representa un ángulo. Por lo tanto si se aumenta lo suficiente, disminuirá la fase de los estados con menor energía, produciendo un resultado contrario al deseado.

3.3. Algoritmo de QAOA

En definitiva, el circuito de QAOA está basado en un estado inicial $|\psi_0\rangle$ y un circuito parametrizado (también llamado circuito ansatz) $U(B, \beta_{p-1})U(C, \gamma_{p-1}) \dots U(B, \beta_0)U(C, \gamma_0)$, cuyo estado resultante se denomina $|\psi(\vec{\beta}, \vec{\gamma})\rangle$.

Como ya ha sido mencionado en la introducción, QAOA es un algoritmo híbrido, por lo que tiene una parte ejecutada en un computador cuántico y otra en un computador clásico. La medición de $|\psi(\vec{\beta},\vec{\gamma})\rangle$ corresponde a la primera, mientras que la variación de $\vec{\beta}$ y $\vec{\gamma}$ es realizada por un optimizador clásico.

Así, la estructura general de QAOA es la siguiente [3]:

1. Inicializar $\vec{\beta}$ y $\vec{\gamma}$ a valores arbitrarios según el número de capas p.
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. Evaluar el estado en la base computacional. Esto, por el funcionamiento del modelo orientado a circuitos, devuelve una estimación de la probabilidad de cada uno de los estados de la base $|x\rangle$.
   2.3. Computar el valor esperado de C: $\langle\psi(\vec{\beta},\vec{\gamma})|C|\psi(\vec{\beta},\vec{\gamma})\rangle$. De forma equivalente, se puede obtener este valor de manera clásica calculando la media ponderada del coste en $f(x)$ de los estados obtenidos según su probabilidad.

   $$\sum_{|x\rangle} f(x) * P(|x\rangle) \tag{3.12}$$

   Donde $f(x)$ es el coste de la función para un valor x obtenido en la ejecución del paso anterior y $P(|x\rangle)$ es la probabilidad estimada de obtener x en dicha ejecución.
   El valor esperado obtenido representa una métrica de cómo de bueno es el resultado de la ejecución del circuito. A menor sea este valor, más cercano está $|\psi(\vec{\beta},\vec{\gamma})\rangle$ al estado fundamental.
   2.4. Hallar nuevos $\vec{\beta_{new}}, \vec{\gamma_{new}}$ usando un algoritmo clásico de optimización y volver al *paso 2.*.
3. Evaluar $|\psi(\vec{\beta_{opt}}, \vec{\gamma_{opt}})\rangle$ para obtener el estado x con mayor probabilidad. Este valor aproximará $\min_x f(x)$.

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

$$\min_{\vec{\beta},\vec{\gamma}} \langle\psi(\vec{\beta},\vec{\gamma})|C|\psi(\vec{\beta},\vec{\gamma})\rangle = \min_x f(x) \tag{3.13}$$

Esto es equivalente a decir que el estado fundamental (sección A.4) del sistema corresponde al valor mínimo de la función de coste.

De esta forma, el optimizador clásico llamará en repetidas iteraciones a una función 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

En este capítulo se describe cómo se han aplicado los algoritmos a los tres problemas de optimización planteados, como se ha indicado en el flujo de trabajo del TFG (fig. 3.1).

En primer lugar, se aplica QAOA al problema presentado en la página web tutorial de Qiskit [10]. En el siguiente problema se reproduce el circuito obtenido en el artículo por Urgelles *et al.* (2022) [8], el cual aumenta la complejidad con respecto al problema previo al añadir restricciones y aumentar el número de qubits. Adicionalmente, se decide aplicar las mismas pruebas sobre otro grafo, para estudiar el funcionamiento del algoritmo al aumentar el número de capas, siguiendo el artículo por Fan *et al.* (2023) [11].

La implementación de QAOA ha sido realizada en código Python en cuadernos de Jupyter y se encuentra disponible en un repositorio de Github habilitado (*sección G* del apéndice).

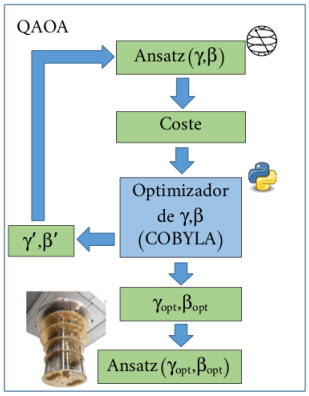


Figura 4.1: Esquema del algoritmo QAOA

En el *código 4.1* se muestra un fragmento ilustrativo de la implementación del algoritmo.

Código 4.1: Fragmento de código de una ejecución de QAOA.

```
1  def compute_expectation(counts: Dict[str, int]) -> float:
2      media = 0
3      for bits, count in counts.items():
4          cost = eval_cost_function(bits)
5          media += cost *count
6
7      return media
8
9  def execute_circuit(theta: List[float]) -> float:
10     qc = generate_qaoa_circuit(theta)
11     counts = sampler.run(qc).result().quasi_dists[0].binary_probabilities()
12     return compute_expectation(counts)
13
14 theta = minimize(execute_circuit, [1.0, 1.0] *num_layers, method="COBYLA")
```

La correspondencia entre el código escrito y el funcionamiento del algoritmo es el siguiente.
Como ya se ha mencionado, para modificar los parámetros β, γ se hace uso de un optimizador clásico llamado *Constrained Optimization BY Linear Approximation* (COBYLA), perteneciente a la biblioteca *scipy* (línea 14 del *código 4.1*).
Este realiza sucesivas llamadas a un método con una lista de floats (que corresponde a los vectores β y γ) como entrada y un float como salida (que corresponde al valor esperado del operador C). Tras cada llamada a este método se modifican los valores de entrada, con el fin de minimizar la salida (coste) obtenida.
En este caso, ese método a minimizar es denominado *execute_circuit()* (línea 9 del *código 4.1*) y tiene tres partes:

- La generación del circuito cuántico a partir del parámetro *theta*, el cual corresponde con los dos vectores β y γ concatenados, de acuerdo con el circuito ansatz del problema, calculado previamente. Esto es realizado por la función *generate_qaoa_circuit()*. Corresponde con el *paso 2.1.* del algoritmo.
- La ejecución del circuito utilizando la librería de Qiskit. Corresponde al *paso 2.2.* del algoritmo.
- La evaluación del resultado obtenido, realizado por la función *compute_expectation(counts: Dict[str, int])* (línea 1 del *código 4.1*), el cual calcula el valor esperado de C de manera clásica, donde *eval_cost_function()* devuelve el coste un valor en la función de coste QUBO. Corresponde al *paso 2.3.* del algoritmo.

Una vez se cumplan los criterios de convergencia seguidos por COBYLA, se devolverá esa entrada *theta*, equivalente a $\vec{\gamma}|\vec{\beta}$, con la que se puede volver a hacer la ejecución cuántica del circuito ansatz resultante de *generate_qaoa_circuit()*, para lograr el resultado del algoritmo.

4.1. Max-cut en grafo de 4 aristas

Este problema, obtenido en la página web tutorial de Qiskit [10], consiste en resolver max-cut para el grafo de la *fig. 4.2*, que consiste en, asignando un valor 0 o 1 a cada nodo maximizar la cantidad de aristas entre nodos de distinto valor.

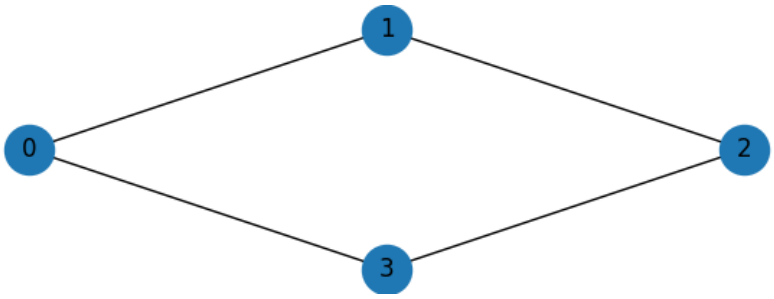


Figura 4.2: Grafo del problema [10]

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 2^4. Estas condiciones hacen que el algoritmo trabaje sobre un circuito pequeño, tanto en profundidad como en número de qubits, lo que reduce el ruido y facilita su implementación.

- **Formulación:**

  Sea $G = (V, E)$ el grafo de la *fig. 4.2*, con $V = [0, 1, 2, 3]$ y $E = [(0, 1), (1, 2), (2, 3), (0, 3)]$ los nodos y aristas de G respectivamente.

  Sea $x_i \in \{0, 1\}$ el color del nodo i.

- **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 multiplicará 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:

$$f(x) = \sum_{(i,j)\in E} (x_i * (x_j - 1) + x_j * (x_i - 1)) \tag{4.2}$$

Así, cada cláusula del sumatorio será -1 para una arista con dos nodos de distinto valor y 0 en otro caso.

Como el problema no tiene restricciones y las variables de entrada son $x_i \in \{0, 1\}$, el problema ya se encuentra en formato QUBO. El paso a formato Ising se realiza de acuerdo con lo descrito en la *sección 3.2.2*. Utilizando el cambio de variable $x_i \to \frac{1-z_i}{2}$ se genera la siguiente función $g(z)$:

$$g(z) = \sum_{(i,j)\in E} (\frac{1-z_i}{2} * (\frac{1-z_j}{2} - 1) + \frac{1-z_j}{2} * (\frac{1-z_i}{2} - 1)) = \sum_{(i,j)\in E} \frac{z_i z_j - 1}{2} \tag{4.3}$$

Como ha sido explicado en la *sección 3.2.2* y demostrado en la *sección D* del apéndice, para obtener el operador C se deben sustituir las variables z_i por puertas σ_i^z. Además se debe tener en cuenta que debido al postulado de medición en mecánica cuántica [13] la fase global es despreciable. Esto significa que dado un operador lineal A y $n \in \mathbb{R}$: $e^{i\gamma n} \cdot e^{i\gamma A} = e^{i\gamma A}$

Y teniendo en cuenta las siguientes definiciones:

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

Se obtiene:

$$\begin{aligned} U(C,\gamma) = \exp(-i*\gamma*C) &= \exp\left(-i*\gamma*\sum_{(i,j)\in E}\frac{\sigma_i^z\otimes\sigma_j^z - 1}{2}\right) = \\ &= \prod_{(i,j)\in E}\exp\left(-i*\gamma*\frac{\sigma_i^z\otimes\sigma_j^z - 1}{2}\right) = \\ &= \prod_{(i,j)\in E}[\exp\left(-i*\frac{\gamma}{2}*\sigma_i^z\otimes\sigma_j^z\right)*\exp\left(i*\frac{\gamma}{2}\right)] = \\ &= \prod_{(i,j)\in E} Rz_i z_j(\gamma) \end{aligned} \tag{4.4}$$

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.

Discusión de la diferencia entre circuitos de distintas implementaciones de QAOA

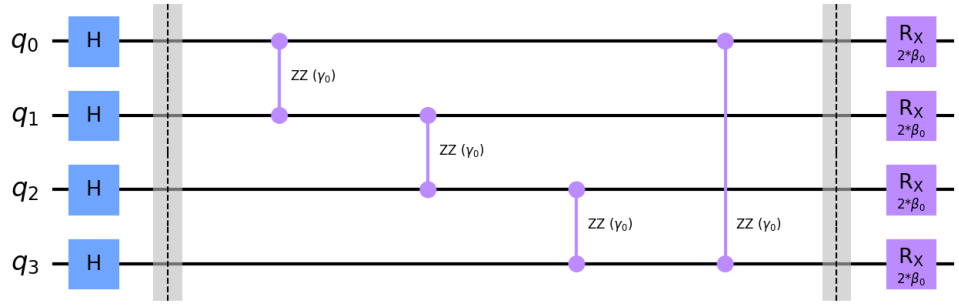


Figura 4.3: Circuito obtenido con la implementación de QAOA del TFG ($p = 1$)

El circuito generado con el desarrollo de este TFG (*fig. 4.3*) y el generado por la función *QAOAAnsatz* de la librería Qiskit (*fig. 4.5*) son equivalentes. Las diferencias son únicamente visuales con res-

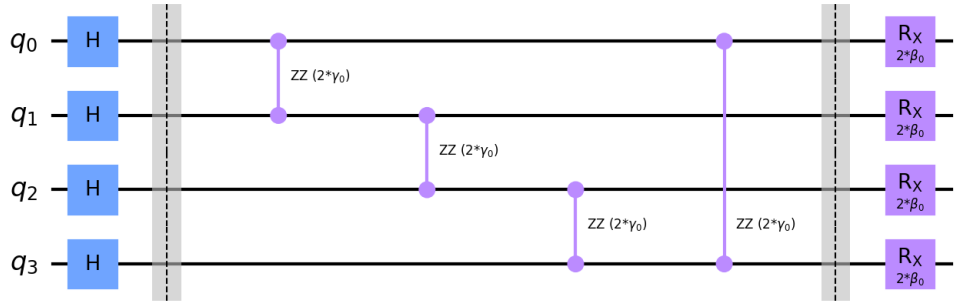


Figura 4.4: Circuito de ejemplo de la página web [10] ($p = 1$)

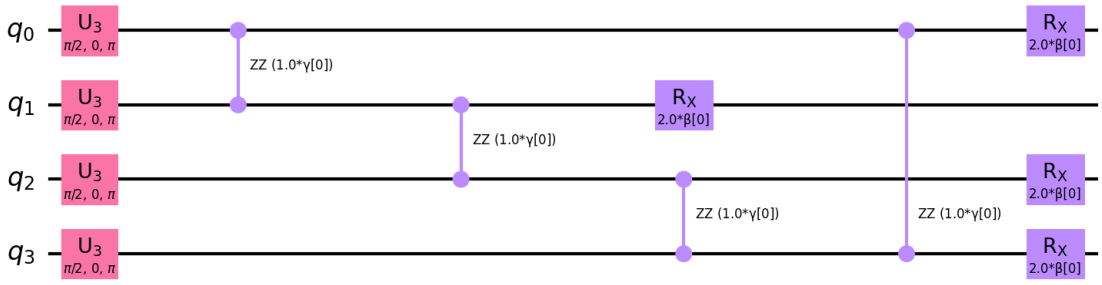


Figura 4.5: Circuito generado por QAOAAnsatz ($p = 1$)

pecto a la disposición de las puertas.

El circuito obtenido de la página web tutorial de Qiskit [10] (*fig. 4.4*) corresponde a desarrollar los mismos cálculos solo que con $f_2(x) = 2 * f(x)$ como función de coste.
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 que el otro. En otras palabras ambas funciones f_2 y f tienen un mismo x que las minimiza.

4.2. Camino más corto en grafo de 4 nodos

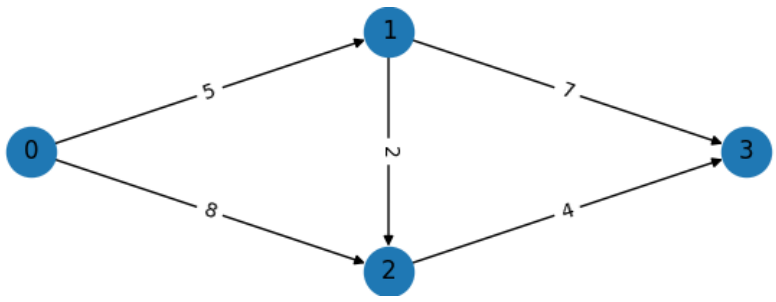


Figura 4.6: Grafo del problema (Urgelles *et al.*, 2022 [8])

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.

El problema, presentado en el artículo por Urgelles *et al.* (2022) [8], consiste en encontrar el camino más corto que conecte los nodos *0* y *3*.

- **Objetivo:**

$$\text{mín}(5X_{01} + 8X_{02} + 2X_{12} + 7X_{13} + 4X_{23}) \tag{4.5}$$

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

- **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 [8] se elige el valor del modificador de Lagrange (*P*) tal que sea estrictamente mayor que el máximo de la función objetivo:

$$P = 1 + \max_x \left(f_{\text{sin restricc}}(X)\right) = 1 + \sum_{(i,j)\in E} w_{ij} = 27 \tag{4.6}$$

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

$$\begin{aligned} f(X) = {} & 5X_{01} + 8X_{02} + 2X_{12} + 7X_{13} + 4X_{23} + \\ & + P(X_{01} + X_{02} - 1)^2 + P(X_{01} - X_{12} - X_{13})^2 + P(X_{02} + X_{12} - X_{23})^2 \end{aligned} \tag{4.7}$$

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. Por lo tanto se emplean 5 qubits.

El paso de la función a formato Ising, explicado en la *sección 3.2.2*, para la conversión de la función de coste al circuito ansatz se realiza teniendo en cuenta que las variables X toman valores 0 y 1. 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.

Estas nuevas variables z van a tomar valores -1 y 1 y cada variable z_k estará asociada al qubit k-ésimo del circuito.

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

$$\begin{aligned} g(z) =& 5\frac{1-z_0}{2} + 8\frac{1-z_1}{2} + 2\frac{1-z_2}{2} + 7\frac{1-z_3}{2} + 4\frac{1-z_4}{2} + \\ &+ P(\frac{1-z_0}{2} + \frac{1-z_1}{2} - 1)^2 + P(\frac{1-z_0}{2} - \frac{1-z_2}{2} - \frac{1-z_3}{2})^2 + \\ &+ P(\frac{1-z_1}{2} + \frac{1-z_2}{2} - \frac{1-z_4}{2})^2 = \\ =& 11z_0 - 17{,}5z_1 - 28z_2 - 17z_3 + 11{,}5z_4 + \\ &+ 13{,}5(z_0z_1 - z_0z_2 - z_0z_3 + z_1z_2 - z_1z_4 + z_2z_3 - z_2z_4) + \\ &+ 80{,}5 \end{aligned} \tag{4.8}$$

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* (demostrado en la *sección D* del apéndice) se obtiene el hamiltoniano $U(C, \gamma)$ a partir de $g(z)$:

$$\begin{aligned} &U(C,\gamma) = \exp(-i\gamma C) = \\ &= Rz_0(11 * 2\gamma) \cdot Rz_1(-17{,}5 * 2\gamma) \cdot Rz_2(-28 * 2\gamma) \cdot Rz_3(-17 * 2\gamma) \cdot Rz_4(11{,}5 * 2\gamma) \cdot \\ &\cdot Rz_0z_1(+13{,}5 * 2\gamma) \cdot Rz_0z_2(-13{,}5 * 2\gamma) \cdot Rz_0z_3(-13{,}5 * 2\gamma) \cdot Rz_1z_2(+13{,}5 * 2\gamma) \cdot \\ &\cdot Rz_1z_4(-13{,}5 * 2\gamma) \cdot Rz_2z_3(+13{,}5 * 2\gamma) \cdot Rz_2z_4(-13{,}5 * 2\gamma) \end{aligned} \tag{4.9}$$

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.

Discusión de la diferencia entre circuitos de distintas implementaciones de QAOA

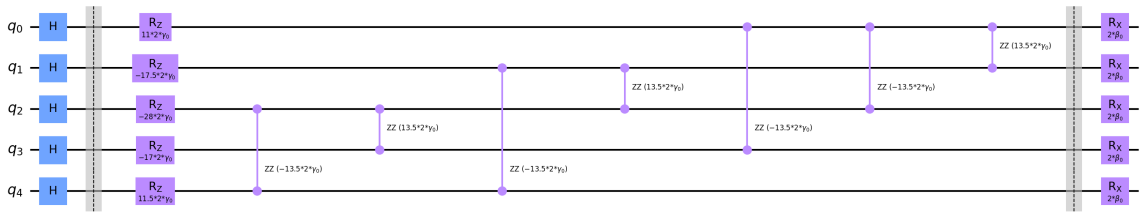


Figura 4.7: Circuito obtenido con la implementación de QAOA del TFG ($p = 1$)

El circuito generado por la implementación propia de QAOA (*fig. 4.7*) es equivalente al generado por la función *QAOA Ansatz* (fig. 4.9). El circuito generado por el artículo (fig. 4.8) sí muestra diferencias con respecto a los otros dos circuitos.

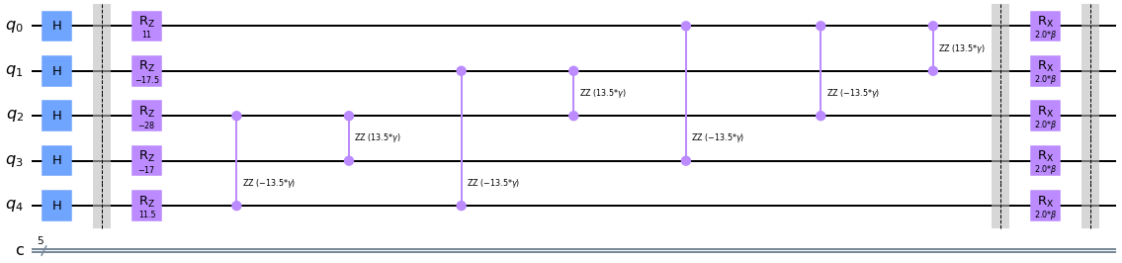


Figura 4.8: Circuito del artículo [8] ($p = 1$)

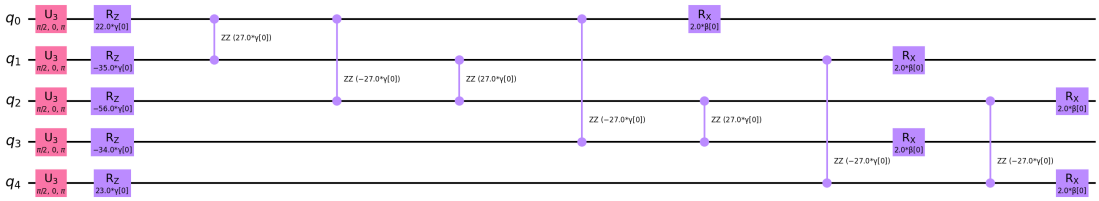


Figura 4.9: Circuito generado por QAOAAnsatz ($p = 1$)

- Los operadores aparecen con la forma $Rzz(n), Rz(n)$ en lugar de $Rzz(n * 2), Rz(n * 2)$ respectivamente. De forma similar al problema anterior, esto no es considerado un error, ya que en el caso del artículo [8] se estará calculando sobre un operador $C' = \frac{C}{2}$, que tiene un mismo estado fundamental pero con la mitad de energía.

- Las puertas Rz no dependen de γ. Esto hace que las rotaciones llevadas a cabo por estas sean constantes lo cual, de acuerdo con el desarrollo matemático mostrado anteriormente y el resultado mostrado por la función de Qiskit que implementa QAOA, es erróneo.

Por esto se concluye que existe una imprecisión en la implementación del artículo y se notifica debidamente a los autores.
Se han realizado pruebas utilizando ambos circuitos para realizar comparaciones y se ha comprobado que la implementación de este TFG coincide con la de la librería Qiskit.

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

Para comprobar el efecto del número de capas del algoritmo QAOA en el problema del camino más corto, se decide ampliar con pruebas sobre otro grafo de distinta dificultad.

Este grafo (*fig. 4.10*) tiene la misma cantidad de nodos que el grafo anterior (*fig. 4.6*) y una arista menos. Esto se traduce en una aplicación de QAOA con menos restricciones pero con los mismos

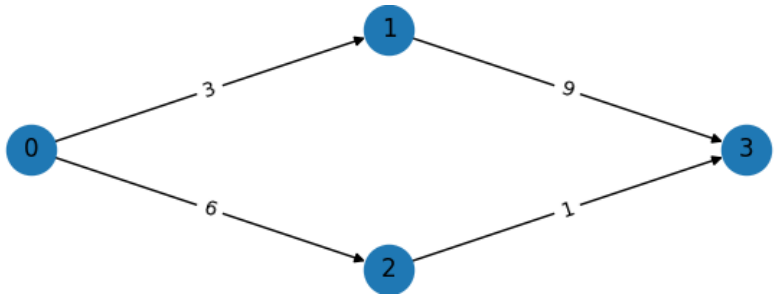


Figura 4.10: Grafo del problema (Fan *et al.*, 2023 [11])

qubits.

Con esta diferenciación se pretende estudiar el comportamiento de estos problemas al aumentar el número de capas.

- **Objetivo:**

$$\text{mín}(3X_{01} + 6X_{02} + 9X_{13} + 1X_{23}) \tag{4.10}$$

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

- **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. $X_{13} + X_{23} = 1$: Una restricción extra, similar a la primera. Debe haber exactamente un eje del camino que involucre al nodo final. Fuerza a que el camino finalice en dicho nodo

Con el fin de mantener las similitudes con el problema anterior se elige el modificador de Lagrange (P) utilizando la misma métrica, a saber:

$$P = 1 + \max_x \left(f_{\text{sin restricc}}(x)\right) = 1 + \sum_{(i,j)\in E} w_{ij} = 20 \tag{4.11}$$

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

$$\begin{aligned} f(X) = {} & 3X_{01} + 6X_{02} + 9X_{13} + 1X_{23} + \\ & + P(X_{01} + X_{02} - 1)^2 + P(X_{13} + X_{23} - 1)^2 + P(X_{01} - X_{13})^2 + P(X_{02} - X_{23})^2 \end{aligned} \tag{4.12}$$

El desarrollo para obtener el hamiltoniano del problema, realizado con el mismo procedimiento que

los problemas anteriores, se encuentra en la *sección E* del apéndice
El hamiltoniano del problema (descrito en la *sección 3.2*) partiendo de $f(X)$ es:

$$\begin{aligned} &U(C,\gamma) = \exp(-i\gamma C) = \\ &= Rz_0(-1{,}5 * 2\gamma) \cdot Rz_1(-3 * 2\gamma) \cdot Rz_2(-4{,}5 * 2\gamma) \cdot Rz_3(-0{,}5 * 2\gamma) \cdot \\ &\cdot Rz_0z_1(10 * 2\gamma) \cdot Rz_0z_2(-10 * 2\gamma) \cdot Rz_1z_3(-10 * 2\gamma) \cdot Rz_2z_3(10 * 2\gamma) \end{aligned} \tag{4.13}$$

Con el *hamiltoniano de mezcla* y el vector inicial, definidos en la *sección 3.2* y el **hamiltoniano del problema** obtenido se puede construir el circuito cuántico.

Discusión de la diferencia entre circuitos de distintas implementaciones de QAOA

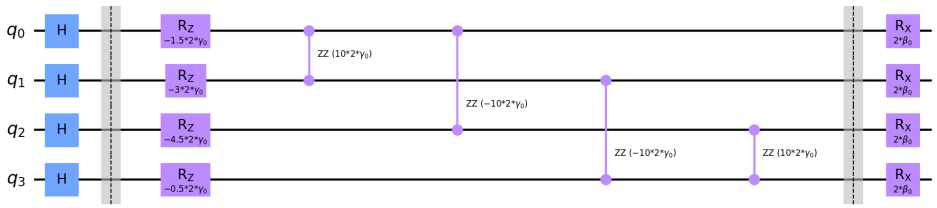


Figura 4.11: Circuito obtenido con la implementación de QAOA del TFG ($p = 1$)

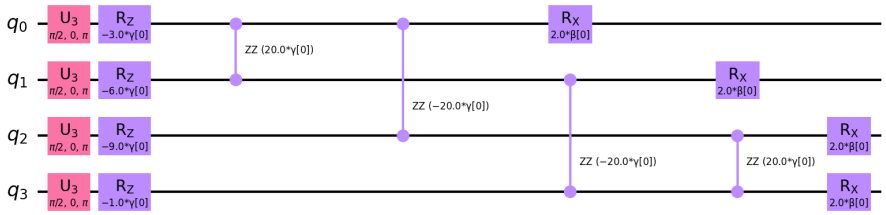


Figura 4.12: Circuito generado por QAOAAnsatz ($p = 1$)

En el artículo por Fan *et al.* (2023) [11], del que se ha obtenido el problema, no se muestra la construcción del circuito de QAOA, por lo que el resultado obtenido en la *fig. 4.11* solo puede ser comparado con el circuito generado por la función *QAOAAnsatz()* (fig. 4.12).
Como los circuitos son equivalentes se asume que la implementación del circuito parametrizado es correcta.

5

Integración, pruebas y resultados

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 que describen el estado del sistema. Así, cuando se habla del resultado de la ejecución de un circuito cuántico, como el mostrado en la *fig. 5.5*, se refiere a probabilidades asociadas a cada estado de la base computacional, que han sido calculadas mediante sucesivas ejecuciones del programa. La cantidad de veces que se ejecuta cada circuito, viene dado por el número de *muestras* (*shots*).

Todas las ejecuciones de Qiskit, salvo en casos señalados, han sido ejecutadas en un simulador en local utilizando los métodos de *Qiskit Runtime* y para las estadísticas necesarias han sido repetidas 1000 veces, que constituye el total de ejecuciones (*TE*).

Para los resultados de QAOA, con el fin de medir la eficacia de distintas ejecuciones, se han utilizado las siguientes métricas:

1. **NA/TE**: Con esta métrica se busca despreciar el ruido presente en cada ejecución. Para ello se realizan *TE* 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.

$$\text{NA/TE} = \frac{NA}{TE} \tag{5.1}$$

   Donde *NA* (número de aciertos) es la cantidad de ejecuciones en las que se ha encontrado el camino óptimo y $TE = 1000$ es el total de ejecuciones realizadas.

   Ejemplo:

   Tomando como ejemplo de salida de QAOA la mostrada en la *fig. 5.5* esto sumaría al cálculo de **NA/TE** $\frac{1}{n}$, ya que el camino óptimo es el obtenido con mayor probabilidad en la ejecución.

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* se indiquen al ejecutar el circuito (por defecto 1024)

Es el caso de gráficas como la de la *fig. 5.5* en la que el eje vertical muestra la probabilidad aproximada de cada valor.

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

$$\frac{1}{TE}\sum_{i=1}^{TE}(MM_i) \tag{5.2}$$

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 *fig. 5.5* la media de muestras del óptimo sería $MM_i = 0{,}59$.

3. **Función gamma:** Se ha utilizado para comprobar la forma general que tiene la función *execute_circuit()*, a minimizar por el optimizador clásico. Esta función pretende representar cómo cambia el valor esperado del algoritmo con la variación del parámetro γ_0 para una versión del circuito correspondiente con $p = 1$. Para esto se ha calculado el resultado del método *execute_circuit()*, con $\beta = 1$ constante para distintos valores de γ. Se ha elegido mantener β constante porque, como ha sido explicado en la definición del hamiltoniano de mezcla (*sección 3.2.2*), este sirve solo para convertir los cambios de fase generados por el hamiltoniano del problema en cambios en la amplitud, pero se ve afectado por la definición del problema específico a resolver.

   Al variar γ y representar la función resultante se puede intuir cómo de posible es encontrar mínimos locales de energía 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.

Estos tres métodos han sido implementados en Python, en los mismos cuadernos de Jupyter donde se encuentran las implementaciones del algoritmo QAOA [14].

A continuación se mostrarán los resultados de ejecución utilizando ambos algoritmos, 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.

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 tutorial de Qiskit [10], 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 ($f(x) = 4$). Un resultado "1010" sería equivalente a dar un valor "1" al nodo 3, "0" al nodo 2 y así sucesivamente.

5.1.1. Resultados con QAOA

Las estadísticas obtenidas son las siguientes:

nº Capas	NA/TE	MM/TE
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 [10]

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.

Se ha representado la función gamma, que muestra el comportamiento de la energía en función del parámetro γ, para poder visualizar posibles mínimos locales que se pudieran encontrar al ejecutar el optimizador clásico:

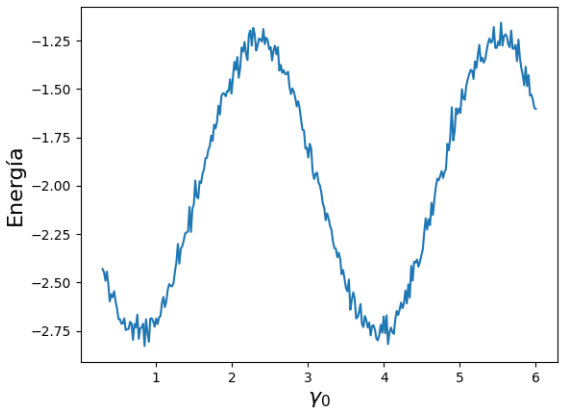


Figura 5.1: Función gamma de *execute_circuit()* (con $\beta = 1{,}0$ y variando γ)

Como se puede ver, la función toma una forma sinusoidal y, como se verá posteriormente, tiene mucho menos ruido que otras ejecuciones. 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	MM/TE
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 *fig. 4.2*

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.

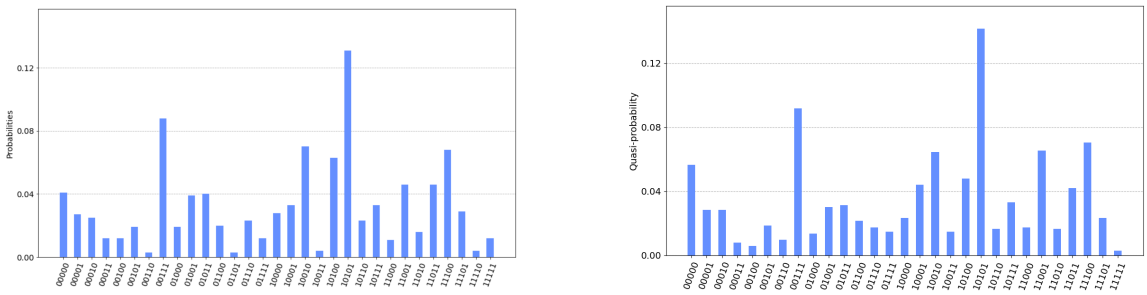
5.2. Camino más corto en grafo de 4 nodos

Como en la sección anterior se ha logrado una implementación correcta de QAOA para un problema simple se investiga 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 y 3 para el grafo de la *fig. 4.6*.

5.2.1. Resultados con QAOA

Solución del artículo de Urgelles *et al.* (2022)

En las siguientes muestras se ha buscado replicar los resultados del artículo [8] del que se ha obtenido el problema. Para ello se ha tomado el circuito parametrizable según los autores, mostrado en la *fig. 4.8*. La prueba de la igualdad entre la implementación propia y la del artículo se encuentra en que, empleando los parámetros $\beta = 0{,}28517317$ y $\gamma = -5{,}05969577$ dados como óptimos, se obtiene un gráfico muy similar al del artículo (*fig 5.2*). El nivel de precisión dado en los parámetros es superior al requerido, ya que con una precisión de 3 o 4 decimales se obtienen los mismos resultados.



(a) Resultado del artículo [8]

(b) Resultado obtenido

Figura 5.2: Comparación entre resultado de QAOA obtenido y resultado del artículo [8]

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	MM/TE
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 [8]

En la *tabla 5.4* se puede observar cómo, en comparación con resultados posteriores de problemas del camino más corto, el caso para $p = 1$ exhibe un comportamiento inusualmente bueno. Esto contrasta con los resultados con valores de capas más altos para los que, al contrario de lo esperado teóricamente, el rendimiento empeora.

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 *fig. 5.2*.

El resultado de la función gamma es el siguiente:

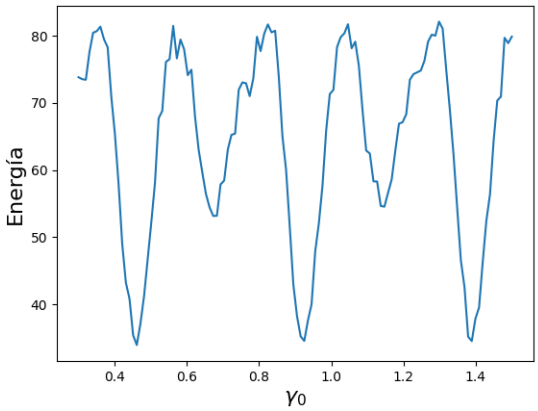


Figura 5.3: Función gamma de *execute_circuit()* (con $\beta = 1{,}0$ y variando γ). Solución del artículo [8]

Se puede ver que existen un gran número de mínimos locales, lo cual dificulta la tarea del optimizador clásico. Haciendo una prueba empleando como parámetros iniciales $(\beta = 1{,}0, \gamma = 0{,}5)$, muy cercanos al mínimo, se obtiene el camino óptimo el 100 % de las ejecuciones, lo que sugiere que el problema se encuentra en que el optimizador clásico no encuentra el mínimo de la función *execute_circuit()*.

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 mejora al aumentar el número de capas, como se esperaba.

Soluciones según el desarrollo seguido en el trabajo

Tomando como punto de partida el circuito de la *fig. 4.7*, correspondiente con el desarrollo correcto de QAOA, se obtienen los siguientes resultados.

Las estadísticas obtenidas en la *tabla 5.5* no dan ningún resultado satisfactorio.
Se aprecia un comportamiento cíclico, similar al que se verá en los resultados del grafo de la *tabla 5.8*. La tendencia observada es que se mejoran las estadísticas hasta las 3 capas y luego disminuyen.

La función gamma resultante se muestra en la *fig. 5.4*. Tiene un mínimo global menos claro con respecto a la función gamma de la versión del circuito del artículo, en la *fig. 5.3*. Además a diferencia

nº Capas	NA/TE	MM/TE
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 con la solución según el desarrollo del trabajo

de dicha gráfica, en esta no se aprecia periodicidad alguna.

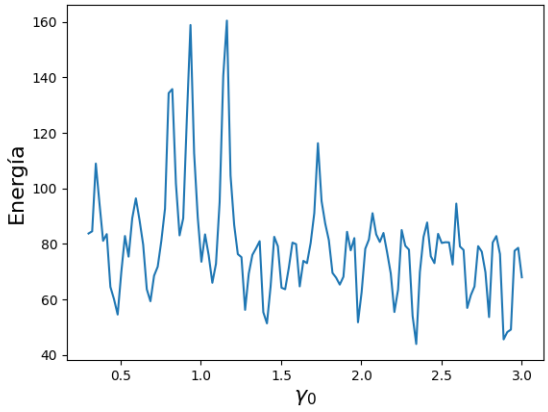


Figura 5.4: Función gamma de *execute_circuit()* (con $\beta = 1{,}0$ y variando γ). Solución según el desarrollo del trabajo.

Se han realizado modificaciones al algoritmo, mostradas y discutidas en la *sección F.1* del apéndice.

5.2.2. Resultados con QAOA en computador cuántico real

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

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

$p = 1$

El código ha sido ejecutado con la imprecisión encontrada en el artículo [8] (*sección 4.2*), tratando así de asemejarse a los resultados que ahí se exponen.

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 COBYLA 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 el mejor valor posible, que corresponde a una energía de 11, que se daría en el caso de obtener la solución correcta (10101) en todas las muestras. El otro valor a remarcar, *nfev*=30, se refiere al número de llamadas a la función *execute_circuit()* realizadas.

Los parámetros obtenidos, ejecutados en un simulador, dan el resultado de la fig. 5.5.

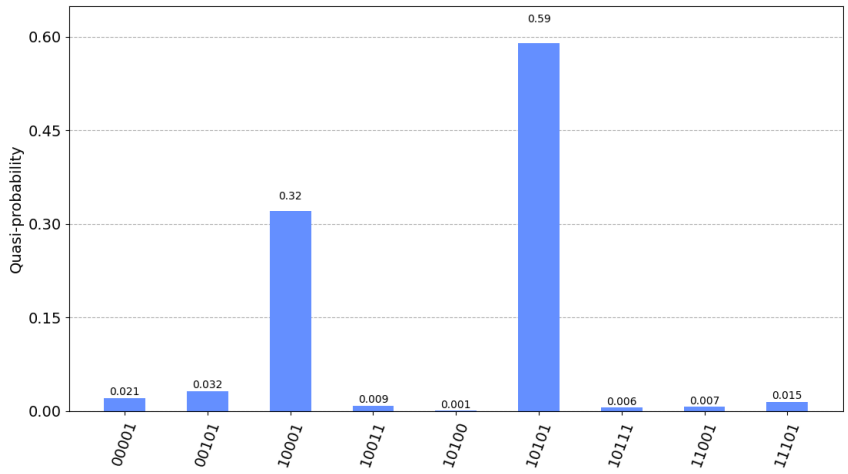


Figura 5.5: Resultado de la ejecución del circuito con $\gamma_{opt}, \beta_{opt}$ en simulador con 1024 muestras

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 obtenidos de la ejecución en *ibm_nairobi* han sido ejecutados en el computador de *ibm_lagos*, dando los resultados de la *fig. 5.6*.

Comparando las *fig. 5.5* y *fig. 5.6* se aprecia la diferencia de ejecutar unos mismos parámetros en un simulador y un computador 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.

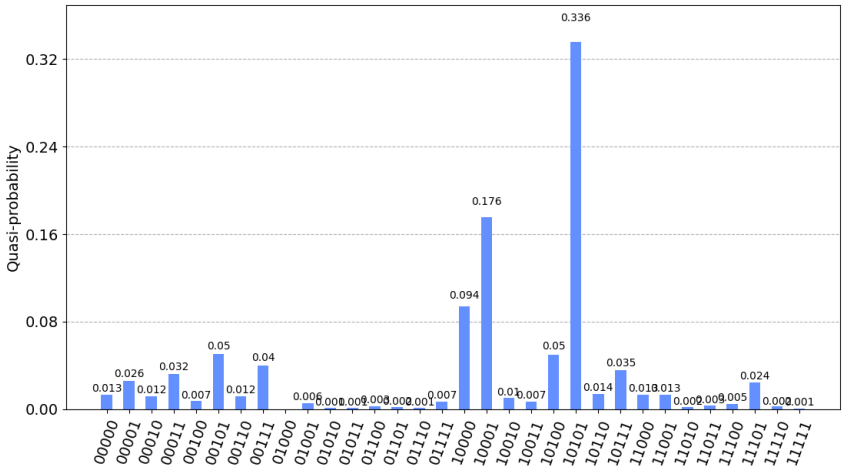


Figura 5.6: Resultado de la ejecución del circuito en *ibm_lagos*

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 5.6*

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

La energía del sistema es equivalente 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 (*tabla 5.4*), 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 QAOA se encuentran ejemplos, como el camino 00111 en la *fig. 5.2*, que ha aparecido en una gran cantidad de muestras pero también tiene una energía elevada.
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

modificador 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 hamiltoniano del problema.

nº Capas	NA/TE	MM/TE
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 5.7: Resultados utilizando *QAOAAnsatz* con el problema de la *fig. 4.6*

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 [8] del que se ha obtenido el problema no son fiables (resultados de la tabla 5.4).

Como ha sido discutido al comentar la función gamma (fig. 5.3) obtenida del circuito del artículo, inicializar β y γ a valores concretos puede llevar a un funcionamiento bueno del algoritmo, aunque de manera artificial. De la misma forma los resultados de un circuito con erratas, como es el dado por el artículo (*tabla 5.4*) pueden ser mejores para capas bajas que los dados por un circuito sin erratas (*tabla 5.5*) de manera completamente arbitraria.

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

Dado los resultados de la sección anterior, se aplica QAOA al problema descrito en el artículo [11], 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 desarrollado en la *sección 4.3*.

5.3.1. Resultados con QAOA

nº Capas	NA/TE	MM/TE
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 5.8: Ejecución de QAOA del camino más corto en grafo del artículo [11]

A diferencia de las ejecuciones del grafo correspondiente a la *tabla 5.4*, aquí se ve un claro incremento de los aciertos (**NA/TE**) con el aumento del número de capas hasta que parece estabilizarse para el 40 %.

Se encuentra también el caso especial de $p = 4$, en el que los aciertos del algoritmo bajan hasta un *0.7 %*.
Además, para $p \geq 2$ se aprecian unos valores de **MM/TE** que se mantienen elevados. Esto quiere decir que en situaciones como $p = 4$, aunque solo se encuentre el camino óptimo un $0{,}7\,\%$ de las ejecuciones, la media de apariciones en los histogramas es inusualmente elevada.

El resultado de la función gamma es el mostrado en la *fig. 5.7*:

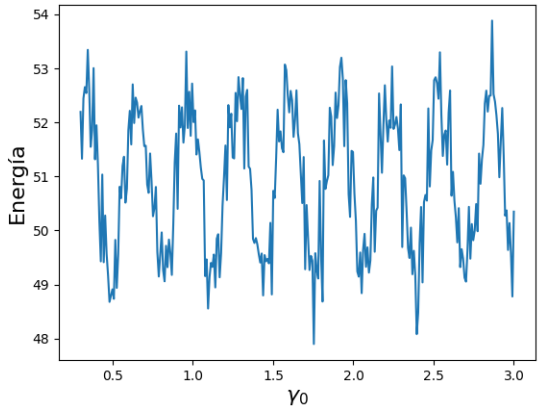


Figura 5.7: Función gamma de *execute_circuit()* (con $\beta = 1{,}0$ y variando γ) del grafo del camino más corto del artículo [11]

A diferencia de las funciones gamma de los problemas de max-cut (*fig. 5.1*) y el camino más corto en grafo de 4 nodos (fig. 5.3), en esta gráfica se ve mucho más ruido, pero también se aprecia cierta

periodicidad a diferencia de la de la fig. 5.3.

En la *sección F.2* del apéndice se muestran otras pruebas realizadas sobre este grafo.

5.3.2. Resultados de D-Wave

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

Tabla 5.9: Resultados de la ejecución en D-Wave del grafo del camino más corto del artículo [11]

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 5.8*).

nº Capas	NA/TE	MM/TE
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 5.10: Resultados utilizando *QAOAAnsatz* con el problema del artículo [11]

6

Conclusiones y trabajo futuro

Conclusiones

Se ha hecho una comparativa entre algoritmos de optimización en dos tecnologías de computación cuántica. De los resultados obtenidos se concluye que la tecnología de *quantum annealing* actualmente presenta una mayor facilidad para la implementación inmediata de los problemas QUBO abordados, además de una mayor tasa de obtención del óptimo.
En D-Wave la implementación de problemas de optimización es más directa. Se puede realizar una ejecución del algoritmo introduciendo la función de coste y restricciones del problema en el formato tipo QUBO, ya que la aplicación del algoritmo es un sistema de caja negra.
Los resultados en D-Wave no muestran incongruencias, ya que todos los resultados inválidos quedan claramente diferenciados en la baja cantidad de muestras en las que se obtienen. Dentro de los caminos válidos, aunque todos resultan con una cantidad de muestras altas, el camino óptimo se diferencia por su menor energía, la cual es equivalente al coste de la función.

En cuanto a la ejecución de algoritmos de optimización en computadores cuánticos basados en circuitos, las principales aportaciones de este trabajo han sido:

- Realizar una explicación completa sobre el funcionamiento e implementación necesarios para resolver un problema de optimización combinatoria cualquiera utilizando QAOA.
- Desarrollar un código en el que se sistematiza la implementación del algoritmo en un circuito cuántico partiendo de una función de coste cualquiera en formato Ising.
- Comparar ejecuciones en computadores reales tanto de D-Wave como de IBM-Q.

De estas aportaciones también se concluye que la comparación entre la implementación de QAOA de este TFG, realizada desde cero, ofrece unos resultados similares a la implementación de QAOA de las librerías de Qiskit, denominada QAOAAnsatz(), por lo que tanto la explicación teórica del algoritmo como el código realizados en este trabajo son correctos.
Tanto por la similitud entre estas dos instancias de QAOA, como por las formas presentadas por las respectivas funciones γ, se concluye que las diferencias encontradas en los problemas del camino más

corto se deben a la naturaleza del problema.

Al realizar el estudio detallado, reproduciendo resultados de artículos publicados, se han encontrado diferencias que han sido discutidas con los autores:

- Para el problema de la *sección 4.1* se concluye que hay una diferencia en los circuitos ansatz de QAOA entre el desarrollo sistemático implementado en este TFG y el resultado mostrado en la página web tutorial de Qiskit [10]. Se ha justificado el motivo por el que el rendimiento del algoritmo no se ve afectado, ya que ambos circuitos parametrizados generan los mismos resultados. El motivo de este cambio llevado a cabo en Qiskit es, muy probablemente, mostrar un circuito simétrico donde todos los operadores tengan una misma rotación, aunque eso se traduzca en una inexactitud matemática porque la energía del sistema no coincida con el valor correspondiente en la función de coste.
- En el grafo resuelto en la *sección 4.2* se han encontrado discrepancias entre la implementación realizada en este trabajo y el circuito mostrado en el artículo [8] del que se ha obtenido el problema. En el circuito mostrado en la fuente del problema, existen operadores que no dependen de ningún parámetro cuando según el desarrollo mostrado en este trabajo y el circuito generado por la función de Qiskit, *QAOAAnsatz()*, sí existe dicha dependencia. Por ello, en el artículo se incurre en una imprecisión, que ha sido debidamente notificada a los autores, y que han confirmado vía correo electrónico.

Trabajo futuro

Este TFG puede suponer un punto de partida para explorar la aplicación de QAOA a otros tipos de problemas cuya resolución eficiente pueda resultar de utilidad.
También se puede estudiar el rendimiento de problemas de optimización no orientados a grafos, para realizar una comparación entre estos y los problemas resueltos en este trabajo.
Como caso representativo de estas dos propuestas, es interesante resolver la versión de optimización del problema de satisfacibilidad booleana, denominado problema de satisfacibilidad máxima, ya que es un caso de problema NP-completo en el que canónicamente se representan otros problemas pertenecientes a ese conjunto.

Para mejorar los resultados obtenidos en QAOA se pueden desarrollar procedimientos post-ejecución, en los que se minimicen los resultados no deseados en la salida del algoritmo, de tal forma que aumenten las muestras asociadas a resultados de baja energía y disminuyan para resultados de alta energía.
Otras modificaciones de QAOA que pueden otorgar mejores resultados son utilizar un hamiltoniano de mezcla diferente al dado por el artículo original del algoritmo o aplicar estrategias diferentes para determinar los parámetros de entrada (γ, β) óptimos en lugar del optimizador clásico *COBYLA*.

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] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," 2000.

[5] A. D. King, S. Suzuki, J. Raymond, A. Zucca, T. Lanting, F. Altomare, A. J. Berkley, S. Ejtemaee, E. Hoskinson, S. Huang, E. Ladizinsky, A. J. R. MacDonald, G. Marsden, T. Oh, G. Poulin-Lamarre, M. Reis, C. Rich, Y. Sato, J. D. Whittaker, J. Yao, R. Harris, D. A. Lidar, H. Nishimori, and M. H. Amin, "Coherent quantum annealing in a programmable 2,000 qubit ising chain," *Nature Physics*, vol. 18, no. 11, p. 1324–1328, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1038/s41567-022-01741-6

[6] IBM, "Theibmquantumdevelopmentroadmap," accessed: 2023-02-11. [Online]. Available: https://www.ibm.com/quantum/roadmap

[7] 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

[8] 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.

[9] M. O. Katanaev, "Adiabatic theorem for finite dimensional quantum mechanical systems," *Russian Physics Journal*, vol. 54, no. 3, p. 342–353, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1007/s11182-011-9620-5

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

[11] Z. Fan, J. Xu, G. Shu, X. Ding, H. Lian, and Z. Shan, "Solving the shortest path problem with qaoa," *SPIN*, vol. 13, 1 2023.

[12] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: http://dx.doi.org/10.3389/fphy.2014.00005

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

[14] O. Julián, "Qaoa," accessed: 2024-05-19. [Online]. Available: https://github.com/orioljulian/QAOA

Apéndices

A

Conceptos básicos

En esta sección se explican ciertos conceptos fundamentales empleados en la computación cuántica, para la comprensión de los algoritmos presentes en este trabajo.
La función de estas explicaciones es dar una guía breve para entender lo explicado a gran escala, pero no se pretende tratar con todas las definiciones formales necesarias para desarrollar QAOA o QA.
Para un aprendizaje en profundidad se recomienda acudir al libro de Nielsen & Chuang (2010) [13], en el que se exponen las bases de la computación cuántica.

A.1. Postulados

En el libro de Nielsen & Chuang (2010) [13] se definen cuatro postulados de la computación cuántica, que consisten, de forma muy simplificada, en:

1. **Espacio de estados y estado de un sistema simple**:
   Este espacio de estados es un espacio de Hilbert complejo de 2 dimensiones. El estado de un sistema, en este caso el qubit/s, viene dado por un vector unitario en el espacio de estados.
   Por ejemplo los qubits $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ y $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ se denominan la base computacional y en función de ella se describe cualquier qubit. Por supuesto se podría emplear cualquier otra base con dos vectores ortonormales, pero se acepta la base computacional como estándar.

2. **Evolución de un sistema**:
   La evolución del sistema viene dada por la ecuación de Schrödinger, que describe cómo evoluciona un qubit/s $|\psi\rangle$ en función de un operador H, denominado el hamiltoniano del sistema.
   $$i\hbar * \frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle \tag{A.1}$$

3. **Medición**:
   El acto de medir un estado cuántico modifica su propio valor.
   Es posible medir en cualquier base, pero típicamente se mide en la base computacional, tal que

el resultado de medir un estado cuántico será $|0\rangle$ o $|1\rangle$ con una probabilidad que será mayor a más cercano sea el estado a dicho vector de la base.

4. **Sistemas compuestos**

   Al unir dos o más qubits entre sí se crea un sistema compuesto, tal que la dimensión del espacio de Hilbert asociado aumenta (para n qubits tiene 2^n dimensiones). La operación utilizada para esto se denomina producto tensorial, y se escribe como "$\otimes$".

A.2. Amplitud

Las amplitudes de un estado definido en una base ortonormal concreta definen cómo se construye dicho estado.
De esta forma, para una base $\{|i\rangle\}$ donde cada vector $|i\rangle$ tiene una amplitud $i \in \mathbb{C}$ asociada para $|\psi\rangle$, se puede definir:

$$|\psi\rangle = \sum_{|i\rangle} i\,|i\rangle \tag{A.2}$$

Cuando las amplitudes i se definen en función de la base computacional ($\{|0\rangle, |1\rangle\}^{\otimes n}$) se cumple que la probabilidad de medir uno de esos estados $|i\rangle$ de la base computacional es:

$$P_{|\psi\rangle}(|i\rangle) = |i|^2 \tag{A.3}$$

Ejemplo: En el estado $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$, al medir en la base computacional $P_{|\psi\rangle}(|0\rangle) = |a|^2$.

A.3. Energía

El hamiltoniano describe el panorama energético de un sistema cuántico. Para un estado $|\psi\rangle$ se puede saber su energía E a partir del hamiltoniano H:

$$E(|\psi\rangle) = \langle\psi|H|\psi\rangle \tag{A.4}$$

Ejemplo: Dado un hamiltoniano $H = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ la energía del estado $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ sería:

$$\langle +|\,H\,|+\rangle = \begin{bmatrix} \sqrt{2}^{-1} & \sqrt{2}^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{2}^{-1} \\ \sqrt{2}^{-1} \end{bmatrix} = 1{,}5 \tag{A.5}$$

A.4. Estado fundamental

El estado fundamental se refiere al estado $|\psi^*\rangle$ de menor energía del sistema.

$$|\psi^*\rangle = \min_{|\psi\rangle \in V} E(|\psi\rangle) \tag{A.6}$$

En QAOA se define un operador C que es diagonal en la base computacional con los valores de la función de coste. Un ejemplo de este operador es el de la *ecuación B.1*.
El operador C tiene como energía el coste de la función clásica asociada.
De esta forma, cuando en QAOA se habla del estado fundamental del operador C se puede hablar de forma equivalente del estado que minimiza la función de coste.

A.5. Operador diagonal

Un operador M es diagonalizable si tiene una representación diagonal con respecto a alguna base ortonormal del espacio de Hilbert V. La representación diagonal de M es:

$$M = \sum_{\lambda} \lambda \, |\lambda\rangle \, \langle\lambda| \tag{A.7}$$

Donde $\lambda \in \mathbb{C}$ son los autovalores de M y $|\lambda\rangle \in V$ sus respectivos autovectores (es decir, $M \, |\lambda\rangle = \lambda \, |\lambda\rangle$).

Si la base de esa representación diagonal es la base computacional, la forma de M será una matriz con 0s y los respectivos autovalores en la diagonal.

Ejemplo: $M = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix} = 4 \, |0\rangle \, \langle 0| + 9 \, |1\rangle \, \langle 1|$

A.6. Fase global

La fase global de un estado $e^{i\theta} \, |\psi\rangle$ es el factor $e^{i\theta}$ y no modifica nunca la probabilidad de medición de los estados de la base computacional. Esto es debido a que, por cómo se define el postulado de la medición, la fase global se cancela.
Es por esto que a efectos prácticos cuando se opera sobre $e^{i\theta} \, |\psi\rangle$ se puede operar de forma equivalente sobre $|\psi\rangle$.

A.7. Fase relativa

Para dos amplitudes $i, j \in \mathbb{C}$ se dice que difieren por una fase relativa si existe $\theta \in \mathbb{R}$ tal que $i = \exp(i\theta) * j$.

Siempre que se mida en la misma base, las variaciones en la fase relativa no modifican la probabilidad de medición.

B

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. Los cálculos han sido plasmados en el script `codigo/phase_and_amplitudes_example.py` del repositorio de código [14], que puede servir de guía también para aprender a realizar operaciones en sistemas cuánticos con numpy.

B.1. Ejemplo de modificación de fases

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

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

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

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 hamiltoniano del problema 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:

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

$$\gamma = 0{,}1 \tag{B.3}$$

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

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:

$$P_\psi(01) = \langle\psi|\, M_{01}^\dagger M_{01}\,|\psi\rangle = \langle\psi|\, M_{01}\,|\psi\rangle = |\frac{1}{2} * e^{-0{,}1i}|^2 = \frac{1}{4} \tag{B.5}$$

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 el libro de Nielsen & Chuang (2010) [13].

B.2. Ejemplo de modificación de probabilidades

Tomando el resultado de la *sección B.1* se puede ver cómo el hamiltoniano de mezcla aumenta la probabilidad de medición de 01.

Como $(\sigma^x \otimes I)^2 = I$ y $(I \otimes \sigma^x)^2 = I$ se puede calcular $U(B,\beta) = e^{-i\beta B} = e^{-i\beta\sigma\otimes I} \cdot e^{-i\beta I\otimes\sigma}$ con la explicación mostrada en la *sección C.1.1* del apéndice.

$$\beta = -0{,}1 \tag{B.6}$$

$$B = \sigma^x \otimes I + I \otimes \sigma^x \tag{B.7}$$

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

$$= \begin{bmatrix} 0{,}99 & -0{,}0993j & -0{,}0993j & -0{,}01 \\ -0{,}0993j & 0{,}99 & -0{,}01 & -0{,}0993j \\ -0{,}0993j & -0{,}01 & 0{,}99 & -0{,}0993j \\ -0{,}01 & -0{,}0993j & -0{,}0993j & 0{,}99 \end{bmatrix} \tag{B.8}$$

$$U(B,\beta) * \frac{1}{2}\begin{bmatrix} e^{-i*0{,}3} \\ e^{-i*0{,}1} \\ e^{-i*0{,}2} \\ e^{-i*0{,}4} \end{bmatrix} = \begin{bmatrix} 0{,}4831 - 0{,}0463j \\ 0{,}5217 + 0{,}0448j \\ 0{,}5142 - 0{,}0047j \\ 0{,}4660 - 0{,}0932j \end{bmatrix} \tag{B.9}$$

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

Calculando la probabilidad de $|01\rangle$ se aprecia un aumento:

$$P_{\psi'}(01) = \langle\psi|\, M_{01}^\dagger M_{01}\,|\psi\rangle = \langle\psi|\, M_{01}\,|\psi\rangle = |0{,}5217 + 0{,}0448i|^2 = 0{,}2742 \tag{B.10}$$

C

Desarrollo de operaciones

C.1. Exponente de una matriz

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

C.1.1. A*A = I

- **Definiciones:**

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

- **Desarrollo:**

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

C.1.2. A es diagonalizable

Dada una función f definida sobre los números complejos se puede definir una función correspondiente sobre A utilizando su representación diagonal(*sección A.5*):

$$f(A) = \sum_{\lambda} f(\lambda) |\lambda\rangle \langle\lambda| \tag{C.3}$$

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

$$\exp(-i\theta A) = \sum_{\lambda} \exp(\lambda) |\lambda\rangle \langle\lambda| \tag{C.4}$$

D

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.

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

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:

$$f(x) = \langle x| C |x\rangle, \forall x \in \{0,1\}^n \tag{D.2}$$

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

$$C|x\rangle = \begin{bmatrix} C(0\ldots00) & 0 & 0 & 0 \\ 0 & C(0\ldots01) & 0 & 0 \\ \vdots & & \ddots & \\ 0 & 0 & 0 & C(1\ldots11) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = f(x)|x\rangle \qquad \forall x \in \{0,1\}^n \tag{D.3}$$

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

$$\sigma^z |0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = +1 * |0\rangle$$
$$\sigma^z |1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -1 * |1\rangle \quad \text{(D.4)}$$

Esto puede ser generalizado como:

$$\sigma^z |x\rangle = (-1)^x |x\rangle, x \in \{0, 1\} \quad \text{(D.5)}$$

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:

  $$\sigma_i^z |x_0 \dots x_{n-1}\rangle = I \otimes \dots \otimes \sigma_i^z \otimes \dots \otimes I |x_0 \dots x_{n-1}\rangle = (-1)^{x_i} |x_0 \dots x_{n-1}\rangle \quad x_i \in \{0, 1\}, i \in 1, \dots n \quad \text{(D.6)}$$

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

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

  $$\sigma_i^z \sigma_j^z |x_0 \dots x_{n-1}\rangle = I \otimes \dots \otimes \sigma_i^z \otimes \sigma_j^z \otimes \dots \otimes I |x_0 \dots x_{n-1}\rangle = (-1)^{x_i}(-1)^{x_j} |x_0 \dots x_{n-1}\rangle \quad x_i \in \{0, 1\}, i \in 1, \dots n \quad \text{(D.7)}$$

- $c * z_i * z_i = c$

  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 *D.6* y *D.7* para sustituir esos términos $(-1)^{x_i}$ por σ_i^z

E

Grafo de camino más corto — de función clásica a hamiltoniano del problema

En esta sección se muestra cómo realiza la conversión de la función de coste en formato QUBO definida en la *sección 4.3*, al hamiltoniano del problema correspondiente.

Para el paso de la función de la versión QUBO a versión Ising se debe realizar un cambio de variable $X_{ij} \rightarrow \frac{1-z_k}{2}$. 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.

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

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

Al igual que para los problemas anteriores (explicado en la *sección 3.2.2* y demostrado en la *sección D* del apéndice), en el operador C del hamiltoniano del problema cada variable z_i corresponderá a un operador Pauli-Z en el qubit i.
El operador C toma por lo tanto el siguiente valor:

$$\begin{aligned} C = & -1{,}5\sigma_0^z - 3\sigma_1^z - 4{,}5\sigma_2^z - 0{,}5\sigma_3^z + \\ & + 10 * (\sigma_0^z\sigma_1^z - \sigma_0^z\sigma_2^z - \sigma_1^z\sigma_3^z + \sigma_2^z\sigma_3^z) + 49{,}5 \end{aligned} \tag{E.2}$$

De forma inmediata se puede construir el hamiltoniano del problema:

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

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

F

Modificaciones de QAOA para el camino más corto

En esta sección del apéndice se muestran distintas modificaciones aplicadas a QAOA sobre algunos de los problemas tratados en el TFG.

F.1. Camino más corto en grafo de 4 nodos

Partiendo de la solución del artículo [8] descrita en la *sección 5.2.1*, se ha realizado una modificación a la formulación del problema con el fin de obtener un mejor resultado.

Esta modificación ha consistido en 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.

$$X_{13} + X_{23} = 1 \tag{F.1}$$

La intención con este caso es que, al haber una restricción más, los caminos que rompan dicha restricción tengan un coste aún más elevado. Así los resultados deberían ser mejores, al filtrar de manera más agresiva ciertos caminos incongruentes.

nº Capas	NA/TE	MM/TE
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 F.1: Resultados de la ejecución de QAOA añadiendo la restricción de la *fórmula F.1*

Los resultados (*tabla F.1*) 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 (*fig. F.1*) toma, en comparación con la original (*fig. 5.3*), unos valores

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

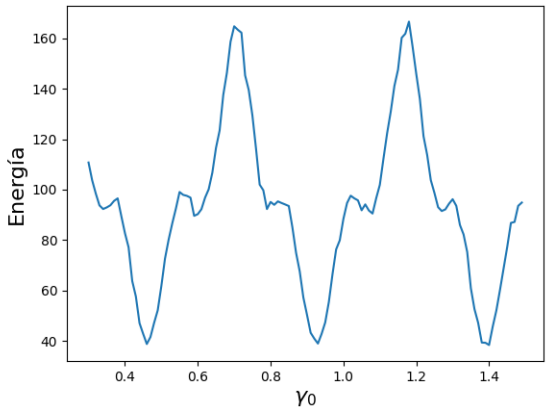


Figura F.1: Función gamma de *execute_circuit()* (con $\beta = 1{,}0$ y variando γ). Con restricción extra

En definitiva, aunque sí se obtienen unos resultados con un mayor índice de la métrica **NA/TE** que los de la *sección 5.2.1*, no es una mejora lo suficientemente pronunciada como para incluir este caso en la sección de resultados.

F.2. Camino más corto para estudiar la variación con el número de capas

F.2.1. Camino más corto omitir variabilidad en puertas Rz

Como primera modificación se ha omitido la variable γ de los operadores Rz, haciendo así que se cometa una imprecisión igual a la del artículo publicado por Urgelles *et al.* (2022) [8].

nº Capas	NA/TE	MM/TE
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 F.2: Ejecución de QAOA del camino más corto en grafo de la *fig. 4.10*

Al igual que en las otras ejecuciones de QAOA en este problema (como la *tabla 5.8*), se observa una mejora al aumentar el número de capas hasta $p = 3$, y después, a diferencia de estas otras, en esta tabla se muestran comportamientos completamente erráticos.

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

Para $p > 3$ se comienzan a observar unos resultados erráticos, siguiendo el mismo comportamiento que en la *tabla 5.5*, la cual corresponde con aplicar el problema anterior a la implementación de QAOA de este trabajo.

F.2.2. Camino más corto aumentando valor del modificador de Lagrange

En el problema original se utiliza $P = 20$.
Este parámetro del algoritmo representa, como ha sido explicado anteriormente, el factor de castigo que se aplica al coste de una cadena de bits de tamaño n que no se encuentre en el espacio de estados del problema.

Con el fin de obtener una diferencia mayor en el coste entre los caminos válidos e inválidos se realizan pruebas sobre el caso mostrado en la *tabla 5.8* cambiando el valor del modificador de Lagrange a P=40.

nº Capas	NA/TE	MM/TE
p = 1	18.8 %	11.89 %
p = 2	45.6 %	19.60 %
p = 3	54.6 %	24.87 %
p = 4	55.0 %	26.17 %
p = 5	39.4 %	20.52 %
p = 6	67.2 %	31.47 %

Tabla F.3: Resultados de la ejecución de QAOA del tercer grafo. P=40

En comparación con la *tabla 5.8*, la *tabla F.3* muestra unos resultados mucho mejores.
Además, son resultados más coherentes con el aumento de p, salvo para el caso $p = 5$, en el que el porcentaje de aciertos disminuye.

G

Repositorio de código

Se ha habilitado un repositorio de Github [14] con todo el contenido de este TFG, incluidos los cuadernos de Jupyter utilizados para realizar las ejecuciones de los algoritmos de este trabajo.

Para ambos Qiskit y D-Wave existen más cuadernos con problemas resueltos y modificaciones de los problemas presentados que los mostrados en el trabajo de fin de grado.

Ejecuciones en D-Wave

Se ha proporcionado un fichero `requirements_dwave.txt` con los paquetes necesarios para utilizar el código de *quantum annealing* en D-Wave.
Para realizar estas ejecuciones además es necesario crear una cuenta en `https://www.dwavesys.com`, para tener un token asociado con el que poder conectarse a los sistemas de D-Wave

El código ejecutado se encuentra en el subdirectorio `codigo/dwave`, separado en cuadernos según el problema a resolver (`shortest-path-dwave.ipynb` y `max-cut.ipynb`).

Ejecuciones en Qiskit

Se ha proporcionado un fichero `requirements_qiskit.txt` con los paquetes necesarios para utilizar el código de QAOA en Qiskit. También es recomendable realizar las ejecuciones en un entorno con Python3 $\leq$ 3.8 (solo ha sido probado para Python 3.8), ya que para versiones superiores surgieron errores empleando las librerías de Qiskit.
Para realizar ejecuciones en computadores reales se debe generar un token de Qiskit en `https://quantum.ibm.com`, pero para cualquier otra ejecución en simulador no es necesario, ya que pueden ser realizadas en la máquina local.

Para las ejecuciones en Qiskit se han utilizado los siguientes cuadernos, todos presentes en el subdirectorio `codigo/`:

- `print_circuits.ipynb`: Utilizado para obtener las imágenes de los circuitos presentadas en este trabajo. Se ha separado de otras secciones porque por defecto un circuito cuántico no muestra los parámetros γ y β, sino coeficientes ya resueltos (es decir, para $\gamma = 1{,}5$ en un operador

$Rz(\gamma * 2)$ mostraría $Rz(3)$).

- `grafo-generico/shortest_path_qaoa.ipynb` y `max-cut-qiskit/max_cut_qaoa.ipynb`: Estos dos cuadernos contienen la implementación de QAOA, generación de estadísticas y ejecuciones en computadores reales.

  Ha sido separado en dos ficheros distintos, según el tipo de problema, para diferenciar las modificaciones en las pruebas y los dibujos de grafos, pero las ejecuciones en circuitos cuánticos y generación de estadísticas son idénticas.

- `grafo-generico/QAOAAnsatz.ipynb` y `max-cut-qiskit/qiskit-QAOAAnsatz.ipynb`: De manera análoga a los ficheros anteriores, contienen las ejecuciones utilizando la implementación de Qiskit de QAOA, separado según el tipo de problema a resolver.

H

Interfaz de Qiskit y D-Wave

En esta sección se muestra una breve documentación sobre la ejecución de programas tanto en Qiskit como en D-Wave.

H.1. Qiskit

Las librerías de Qiskit parecen encontrarse todavía en fases tempranas de su desarrollo, por lo que siguen sufriendo cambios en los métodos para ejecutar circuitos. Por esto es necesario utilizar las mismas versiones de Qiskit, ya que en otro caso podría haber discrepancia en las funciones.

Paquete	Versión
qiskit	0.45.1
qiskit-ibm-runtime	0.17.0
qiskit-ibm-provider	0.8.0
qiskit-aer	0.13.2
qiskit-terra	0.45.1

Tabla H.1: Versiones de Python para Qiskit

Concretamente se emplea el entorno de *Qiskit Runtime*, que es el más utilizado actualmente.

Ejemplo de ejecución en simulador

Código H.1: Ejemplo de ejecución en simulador de Qiskit

```
1  from qiskit import QuantumCircuit
2  from qiskit.primitives import Sampler
3  from qiskit.visualization import plot_histogram
4
5  shots = 1024
6  sampler = Sampler(options={"shots": shots})
7
8  # Construccion de circuito con 2 qubits
```

```
9  qc = QuantumCircuit(2)
10 qc.h(0) # Aplicar puerta Hadamard al qubit 0
11 qc.cnot(0, 1) # Aplicar puerta X con 0 como qubit de control y 1 como objetivo
12 qc.measure_all() # Instrucción para medir
13
14 result = sampler.run(qc).result() # Ejecutar circuito
15
16 probabilities = result.quasi_dists[0] # Resultados obtenidos en la ejecución
17 # {0: 0.51, 3: 0.49}
18
19 quasi_probabilities = probabilities.binary_probabilities()
20 # {00: 0.51, 11: 0.49}
21
22 plot_histogram(quasi_probabilities)
```

Como primer paso un circuito debe ser construido. Una vez especificada la cantidad de qubits (línea 9 del *código H.1*) a incluir se aplican puertas cuánticas explícitamente (líneas 10-12 del *código H.1*).

Para ejecutar en un simulador existen las llamadas primitivas *qiskit.primitives.Sampler* y *qiskit.primitives.Estimator*, que no requieren de la conexión a un computador cuántico real para ser ejecutados:

- *Sampler*: Utilizado para ejecutar el circuito cuántico un número (*shots*) de veces y mostrar los resultados. Esto es equivalente a decir que se calcula $C\,|0\ldots0\rangle$, donde C es equivalente a *qc* en el código.
- *Estimator*: Calcula el valor esperado de un circuito parametrizado (ansatz) y un hamiltoniano (denominado observable). Como ya ha sido explicado en la definición de QAOA, el valor esperado se calcula como:

$$\langle\psi(\theta)|C|\psi(\theta)\rangle \tag{H.1}$$

  Donde $|\psi(\theta)\rangle$ es el estado final del circuito parametrizado y C es el hamiltoniano.

  En el trabajo este objeto no ha sido utilizado, ya que el valor esperado necesario en QAOA (*paso 2.3.* del algoritmo) se puede calcular de manera clásica, a partir de la función de coste.

El circuito del *código H.1* se ejecuta inicializando un objeto *Sampler* con los parámetros necesarios, como el número de muestras (*shots*). La ejecución se realiza en la línea 16 y el resto de instrucciones son para obtener los datos en un diccionario "{bits: probabilidades}" para después obtener su histograma con *qiskit.visualization.plot_histogram*.

Ejemplo de ejecución en ordenador real

Código H.2: Ejemplo de ejecución en computador real de Qiskit

```
1 from qiskit_ibm_runtime import QiskitRuntimeService
```

```
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler, Session, Options

# Guardar token asociado a la cuenta de Qiskit
# Solo se tiene que realizar una vez en cada máquina
QiskitRuntimeService.save_account(channel="ibm_quantum", token="token_ejemplo",
                                  overwrite=True)

QiskitRuntimeService.active_account(QiskitRuntimeService(channel="ibm_quantum"))
service = QiskitRuntimeService(channel="ibm_quantum")

quantum_computer = "ibmq_qasm_simulator" # Nombre del ordenador al que conectarse
backend = service.backend(quantum_computer) # Conexión a dicho ordenador

options = Options() # Objeto opciones para Sampler y Estimator
options.execution.init_qubits = True # Inicializar qubits a 0
options.execution.shots = 512 # Número de muestras

qc = QuantumCircuit(2)
qc.h(0)
qc.cnot(0, 1)
qc.measure_all()

with Session(backend=backend, service=service) as session:
    sampler = Sampler(session=session, options=options)
    quasi_probabilities = sampler.run(qc).result().quasi_dists[0].binary_probabilities()
    quasi_probabilities # Resultado
```

Código H.3: Ejemplo de ejecución en D-Wave

Para poder realizar una conexión con un ordenador cuántico se debe introducir el token de *Qiskit* generado al crear una cuenta. Esto se hace con la función *qiskit_ibm_runtime.QiskitRuntimeService.save_acco* que en Linux genera un fichero `~/.qiskit/qiskit-ibm.json`.

Tanto para la ejecución en un simulador como en un ordenador real se pueden usar unos objetos *Sampler* y *Estimator* equivalentes, solo que para el caso del simulador se obtienen de la librería *qiskit.primitives* y en el caso de conectarse a un computador se obtienen de *qiskit_ibm_runtime*.

Tras conectarse a un ordenador concreto con *qiskit_ibm_runtime.QiskitRuntimeService.backend()* se debe crear un entorno *qiskit_ibm_runtime.Session()*, dentro del cual las ejecuciones se realizan de la forma habitual.

H.2. D-Wave

Los paquetes utilizados son los siguientes:

Paquete	Versión
dwave-ocean-sdk	6.9.0

Tabla H.2: Versiones de Python para D-Wave

```
1  from dimod import BinaryQuadraticModel
2  # Inicializar modelo
3  bqm = BinaryQuadraticModel("BINARY")
4
5  # Función de coste f(x, y) = 2x + 3y
6  peso = {"x": 2, "y": 3}
7  for variable in ["x", "y"]:
8      bqm.add_variable(variable, peso[variable])
9
10 # Añadir restriccion: 3x + 2y -9 = 0
11 bqm.add_linear_equality_constraint([("x", 3), ("y", 2)], constant=-9, lagrange_multiplier=P)
12
13 # Ejecutar programa
14 sampler = EmbeddingComposite(DWaveSampler())
15 sampleset = sampler.sample(bqm, num_reads=1024)
16 print(sampleset)
```

En D-Wave el objeto *dimod.BinaryQuadraticModel* (línea 3 del *código H.3*) describe un modelo que sirve para resolver funciones de coste en versión QBO o Ising con restricciones. En este caso la inicialización a “BINARY” quiere decir que es tipo QBO, en el otro caso sería “SPIN”.

Tanto la introducción de la función de coste como la introducción de restricciones se realizan de formas similares a las líneas 8 y 11 del *código H.3*. El argumento *lagrange_multiplier* corresponde al modificador de Lagrange, utilizado para pasar de modelo QBO a QUBO.

Las sentencias posteriores (líneas 14, 15 del *código H.3*) ejecutan el modelo creado en un ordenador de D-Wave, con una cantidad de muestras *num_reads*.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID