FACULTAD DE INGENIERÍA Y TECNOLOGÍA
ESCUELA DE INGENIERÍA
SEDE SANTIAGO

QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM (QAOA) FOR THE MAX-CUT PROBLEM USING BELL AND GHZ STATES

Tesis para optar al título de Ingeniero Civil Informática

Profesor Tutor: Dr. Sergio José Carrasco Novoa
Estudiante(s): Mauricio Nicolás Pérez Romero

Santiago, Chile
2025

HOJA DE CALIFICACIÓN

En _____, el _____ de _____ de _____, los abajo firmantes dejan constancia que el (la) estudiante Mauricio Nicolás Pérez Romero, de la carrera o programa de Ingeniería Civil Informática ha aprobado la tesis para optar al título o grado académico de Ingeniero Civil Informático, con una nota de _____.

_____

Profesor Evaluador

_____

Profesor Evaluador

_____

Profesor Evaluador

Índice

Índice de Tablas

Índice de figuras

Resumen

El objetivo de esta tesis fue resolver el problema del corte máximo de un grafo (Max-Cut problem) a través del algoritmo cuántico variacional *Quantum Approximate Optimization Algorithm* (QAOA), utilizando distintos tipos de estados iniciales del grafo: entrelazando pares de nodos a través de estados de Bell, entrelazando todos los nodos del grafo por medio de un estado Greenberger-Horne-Zeilinger (GHZ), y por último, considerando la inicialización estándar del algoritmo, en que cada nodo del grafo se inicializa como superposición balanceada de los estados 0 y 1. Se estudió el desempeño del algoritmo en grafos de orden $N = 6, 10$ y 16, con número de aristas $K = 2, N/2$ y $N - 1$, considerando grafos balanceados (todas las aristas de igual peso) y grafos con aristas de distinto peso. Cada grafo fue resuelto empleando profundidades del circuito cuántico iguales a 1, 2 y 3. Las soluciones obtenidas fueron comparadas con la solución óptima encontrada a través de una búsqueda exhaustiva. El algoritmo fue implementado en Pennylane, el lenguaje de computación cuántica de la empresa Xanadu.

En circuitos de mínima profundidad [1], en el 44 % de los grafos (7 grafos), los estados de Bell mejoraron el desempeño del algoritmo, reduciendo la distancia entre la solución encontrada y solución óptima en un 38 % en promedio. A medida que se aumentó la profundidad del circuito, el estado de Bell comenzó a ser más deficiente, siendo mejor solo en un 25 % de los casos (4 grafos) con respecto al algoritmo estándar. En dichos casos, sin embargo, el algoritmo con estado inicial de Bell fue más lento en encontrar la solución final.

Por otra parte, el estado GHZ mostró mejoras considerables para circuitos de profundidad igual a 3. Al compararlo consigo mismo, al cambiar de un circuito de profundidad 1 a un circuito de profundidad 3, se redujo en promedio la distancia entre la solución encontrada y la solución óptima en un 55 %. Al cambiar de un circuito de profundidad 2 a uno de profundidad 3, dicha reducción fue de un 57 %, mostrando que en circuitos de profundidad igual a 3 la inicialización en estado GHZ mejora a más de la mitad, con respecto a un circuito de menor profundidad.

La inicialización convencional resultó mejor en un 56 % de los casos, en términos de la distancia entre la solución encontrada y la solución óptima. A partir de esto, se concluyó que la inicialización estándar del algoritmo QAOA en el problema Max-Cut sigue siendo mejor para este tipo de problemas, pero en grafos con aristas de distinto peso se podría considerar inicializar en estados de Bell para grafos de ciertas condiciones específicas.

Abstract

The objective of this thesis was to solve the Max-Cut problem of a graph using the variational quantum algorithm known as the *Quantum Approximate Optimization Algorithm* (QAOA). Different types of initial states for the graph were used: entangling pairs of nodes through Bell states, entangling all the nodes of the graph using a Greenberger-Horne-Zeilinger (GHZ) state, and finally, considering the standard initialization of the algorithm, where each node of the graph is initialized in a balanced superposition of states 0 and 1. The performance of the algorithm was studied on graphs of order $N = 6, 10$, and 16, with the number of edges $K = 2, N/2$, and $N - 1$, considering balanced graphs (all edges having the same weight) and graphs with edges of different weights. Each graph was solved using quantum circuit depths of 1, 2, and 3. The solutions obtained were compared with the optimal solution found through exhaustive search. The algorithm was implemented in Pennylane, the quantum computing framework developed by Xanadu.

In circuits with minimal depth (1), Bell states improved the algorithm's performance in 44 % of the graphs (7 graphs), reducing the gap between the obtained solution and the optimal solution by 38 % on average. As the circuit depth increased, Bell state initialization became less effective, outperforming the standard algorithm in only 25 % of the cases (4 graphs). However, in these cases, the algorithm with Bell state initialization was slower in finding the final solution.

On the other hand, the GHZ state showed significant improvements for circuits with a depth of 3. When compared to itself, changing from a circuit of depth 1 to a circuit of depth 3 reduced the average gap between the obtained solution and the optimal solution by 55 %. When changing from a circuit of depth 2 to one of depth 3, this reduction was 57 %, showing that for circuits with a depth of 3, GHZ state initialization improves by more than half compared to circuits with lower depths.

The conventional initialization proved to be better in 56 % of the cases in terms of reducing the gap between the obtained solution and the optimal solution. Based on this, it was concluded that the standard initialization of the QAOA algorithm in the Max-Cut problem remains the best choice for this type of problem. However, for graphs with edges of different weights, initializing with Bell states could be considered for graphs with specific conditions.

1. Introducción

La computación cuántica representa una de las áreas más prometedoras y desafiantes de la tecnología moderna. Ha experimentado avances significativos gracias a los esfuerzos de investigación y desarrollo de organizaciones como IBM, Google, D-Wave, Xanadu, Rigetti, entre muchos otros.

La computación cuántica está basada en los principios de la mecánica cuántica, como la superposición y el entrelazamiento. A diferencia de la computación clásica, que utiliza bits como unidades básicas de información, los ordenadores cuánticos emplean qubits, que pueden representar simultáneamente los estados 0 y 1. Este comportamiento permite a los computadores cuánticos resolver ciertos problemas de manera más eficiente que los algoritmos clásicos conocidos, con aplicaciones potenciales en comunicaciones, criptografía, optimización, entre otras áreas (Preskill, 2018).

Uno de los lideres es IBM, es de los pioneros en el desarrollo de hardware y software cuántico. Su plataforma IBM Quantum proporciona acceso a computadoras cuánticas a través de la nube, permitiendo que investigadores y desarrolladores experimenten con circuitos cuánticos reales. Tienen un ordenador cuántico basado en superconductores, es decir, qubits construidos utilizando circuitos superconductores, donde estos circuitos deben ser enfriados casi al cero absoluto para mantener el estado cuántico, al estar el circuito en una temperatura muy baja hace que los electrones viajen por el sin resistencia, por lo que son ”superconductores”. Esta tecnología de superconductividad tiene el procesador Eagle lleva hasta 127 qubits, en un plataforma que puede ser utilizada tanto para aficionados como investigadores llamado Quiskit (IBM Quiskit, s.f). Además, IBM está trabajando en arquitecturas de mayor escala, como el procesador Condor, que apunta a superar los 1000 qubits en los próximos años (IBM, 2022).

Otra de las grandes empresas existentes es Google, con su sistema de aprendizaje en su plataforma llamado Cirq, el cual permite a investigadores, ingenieros a probar sus códigos cuánticos al igual que IBM. Recientemente Google, más especificamente Hartmunt Neven el fundador y director de Google Quantum AI, anunció el 9 de diciembre de 2024 un nuevo chip llamado Willow, el cual menciona que mejoró la calidad del qubit en mayores cantidades. Según se indicó, se redujo el ruido producto de la interacción entre qubits y con el medio ambiente. Está reducción del error la denominó “por debajo del umbral” (Neven, 2024).

El lenguaje de programación empleado en esta tesis es Pennylane, implementado por Xanadu, una empresa líder en el campo de la computación cuántica que ha desarrollando hardware cuántico basado en fotones, lo que quiere decir es que, los cálculos de los algoritmos son realizados a través de fotones me-

diante laseres pulsados para emitir particulas de luz y son manipulados con elementos ópticos. Pennylane es una biblioteca de código abierto basada en python para diseñar y simular algoritmos cuánticos o como en este caso algoritmo híbrido cuántico-clásico. Cuenta con una gran comunidad lo que permite tener buen soporte, tutoriales, documentación, además, destacar que los algoritmos diseñados pueden ser probados en computadoras cuánticas reales tanto como las de Xanadu como las de IBM o Google, debido a que también es compatible con el hardware de estas instituciones, aunque esto implica un costo adicional. En esta tesis, no se utilizó un computador cuántico real sino que un simulador proporcionado por la libreria de Pennylane llamado “lightning.qubit”.

Además, en la computación cuántica, se distinguen 2 tipos de computadoras; computadoras basada en compuertas lógicas que implementan algoritmos mediante el uso de circuitos cuánticos (computadora digital), donde una serie de compuertas lógicas manipulan qubits para realizar cálculos. Estas computadoras son utilizadas por IBM y Google, entre otras.

Por otra parte, se tienen las computadoras adiabáticas donde el sistema se inicia en un estado fundamental [1] mientras se modifica gradualmente un Hamiltoniano, hasta llegar a un estado final. Estas computadoras son mayormente conocidas por D-WAVE (Dwave, s.f).

Estos dos tipos de computadoras permite resolver distintos problemas, pero su enfoque principal se encuentra en problemas NP-hard[2] o NP-completo, que resultan ser irresolubles de forma eficiente para computadoras y algoritmos tradicionales o clásicos. Estos problemas pueden ser el conocido problema del ”vendedor viajero”, ”de corte máximo ”, ”de planificación de tareas”, entre muchos otros problemas NP-hard. En esta tesis se abordará el problema del corte máximo o como bien se conoce el problema Max-Cut, utilizando el algoritmo cuántico Quantum Approximate Optimization Algorithm (QAOA) (Bharti et al., 2022).

El QAOA es un algoritmo cuántico de tipo variacional diseñado para resolver problemas de optimización combinatorial. Este algoritmo combina principios de la computación cuántica y clásica, utilizando un circuito cuántico parametrizado para explorar el espacio de soluciones y buscar un estado que maximice una función de costo determinada (Farhi et al., 2014; Bharti et al., 2022).

En esta tesis, se utilizará el QAOA para resolver el problema Max-Cut en grafos con 6, 10 y 16 vértices. El problema Max-Cut consiste en dividir los vértices de un grafo en dos subconjuntos, maximizando la suma de los pesos de las aristas que los conectan. Se implementarán dos enfoques: el QAOA “convencional” y una versión modificada de este algoritmo. En esta última, el circuito se incializa en estados de Bell (entrelazamiento de a pares) o en estados de Greenberger-Horne-Zeilinger (GHZ). Al usar estados de Bell, se

[1] Estado que se encuentra en un mínimo de energía.
[2] problemas que no pueden ser resueltos por algoritmos tradicionales en tiempo polinomial.

entrelazarán aquellos nodos del grafo conectados por aristas de mayor peso (cuando se tiene un grafo en el que no todas las aristas tienen igual peso).

El documento se encuentra organizado del siguiente modo. En la sección 2 se describen el objetivo general y los objetivos específicos, además del alcance, del presente trabajo. En la sección 3 se resume la metodología de investigación seguida durante este proyecto. Luego, en la sección 4, se introducen los conceptos básicos de la mecánica cuántica, necesarios para describir el algoritmo QAOA. En la sección 5 se presenta en detalle el problema del corte máximo, o "problema Max-Cut". Posteriormente, en la sección 6 se introduce el algoritmo QAOA para resolver el problema Max-Cut. Finalmente, en la secciones 7 y 8 se presentan los resultados de la ejecución del algoritmo para la resolución del problema y las correspondientes conclusiones, respectivamente.

2. Objetivos

Se busca evaluar la hipótesis de que el entrelazamiento de nodos conectados por aristas de mayor peso podría mejorar la eficiencia de la búsqueda de la solución óptima en el problema del corte máximo de un grafo con aristas de distintos pesos, determinando si es conveniente comenzar con una inicialización del algoritmo QAOA diferente a la usual, usando estados entrelazados.

2.1. Objetivo General

El objetivo general consiste en resolver el problema Max-Cut, utilizando el algoritmo Quantum Approximate Optimization Algorithm (QAOA) con diferentes estados iniciales del circuito cuántico (estado inicial estándar,estados de Bell y estados GHZ), evaluando el desempeño en términos de la distancia entre la solución encontrada y la solución óptima del problema.

2.2. Objetivos Específicos

1. Evaluar el impacto de la profundidad del circuito cuántico en la calidad de las soluciones obtenidas y el tiempo de convergencia requerido para encontrar una solución en cada forma de inicialización del algoritmo, considerando profundidades $p = 1, 2$ y 3.

2. Analizar el comportamiento de cada inicialización del algoritmo en grafos balanceados (todas las aristas con peso 1) y no balanceados (aristas de distinto peso), en términos de la calidad de las soluciones encontradas y el número de iteraciones requeridas.

3. Determinar en qué casos es conveniente inicializar el algoritmo en un estado entrelazado (Bell o GHZ) a fin de mejorar la eficiencia de la búsqueda de una solución al problema del corte máximo.

2.3. Alcance

Los algoritmos cuánticos desarrollados, fueron investigados y ejecutados utilizando el simulador cuántico “lightning.qubit” de la biblioteca PennyLane. Este simulador funciona en un computador clásico y permite simular un circuito cuántico en algoritmos que involucren un número reducido de qubits. Los algoritmos desarrollados, sin embargo, sí pueden ser ejecutados en computadores cuánticos reales. Esto último requiere de permisos para el uso del hardware cuántico (Xanadu tiene un computador fotónico). En este trabajo, se utilizó el simulador de PennyLane para evaluar el rendimiento del algoritmo QAOA en la resolución del problema Max-Cut de un grafo con distintos tipos de estados iniciales.

3. Metodología de Investigación

En esta tesis se realizaron simulaciones computacionales, para estudiar el desempeño del algoritmo QAOA al resolver el problema del corte máximo de un grafo con distintos estados iniciales. La planificación seguida a lo largo del semestre para la elaboración de la tesis se detalla en el anexo 9.2. A continuación, se describen los pasos de la metodología de investigación, describiendo cada una de las actividades realizadas en cada etapa.

3.1. Etapas de la investigación

1. **Definición del problema y planteamiento del algoritmo:** Formulación del problema de optimización combinatorial cuadrático Max-Cut como un problema de maximización de la energía de un cierto Hamiltoniano de costos. El objetivo fue traducir el problema clásico a un formato que pudiera ser abordado mediante un circuito cuántico y un proceso de optimización clásico. Para esto se realizaron las siguiente actividades.
    - Revisión bibliográfica para entender los fundamentos teóricos del problema Max-Cut y el algoritmo QAOA.
    - Formulación matemática del problema Max-Cut, considerando grafos con pesos balanceados y no balanceados.
    - Planteamiento del ansatz del algoritmo QAOA, esto es, la parametrización del circuito a través de la cual se busca una solución dentro del espacio de Hilbert.
2. **Diseño e Implementación del algoritmo QAOA:**El diseño incluyó configuraciones específicas para los estados iniciales y los tipos de grafos a analizar. En esta etapa se desarrollaron las siguientes actividades.
    - Generación de grafos balanceados y no balanceados (con pesos 1 y 0.3), considerando distinto número de nodos y aristas.
    - Desarrollo del circuito cuántico parametrizado, incorporando la inicialización estándar y las inicializaciones alternativas (Bell y GHZ).
    - Implementación de funciones para variar la profundidad del circuito (número de pasos), probando configuraciones con profundidades 1, 2 y 3.
    - Implementación de 1000 shots (cantidad de veces que el circuito se repite) para mejorar los resultados probabilísticos del sistema cuántico.

- Validación del diseño del circuito mediante pruebas iniciales controladas (grafos con pocos nodos y aristas, verificando que cada algoritmo converja a la solución óptima).

3. **Simulaciones y experimentos:** Esta etapa fue clave para generar los datos necesarios para el análisis. Se realizaron simulaciones del circuito cuántico, explorando cómo las diferentes configuraciones afectaban los resultados. Las actividades de esta etapa fueron las siguientes.

   - Ejecución de simulaciones para cada inicialización (estándar, Bell y GHZ), tipo de grafo (balanceado y no balanceado) y profundidad del circuito (1, 2 y 3).
   - Comparación de los resultados obtenidos con las soluciones óptimas, calculadas mediante una búsqueda exhaustiva en el espacio de soluciones.
   - Registro de métricas de desempeño (precisión de las soluciones, tiempo de ejecución y la eficiencia computacional).

4. **Análisis de resultados:** El análisis de los datos obtenidos permitió extraer conclusiones generales respecto al comportamiento del algoritmo QAOA a través de las actividades que se describen a continuación.

   - Evaluación de las métricas registradas en las simulaciones, identificando los casos en los que cada inicialización tuvo un mejor rendimiento.
   - Comparación del desempeño del QAOA según el tipo de grafo y la profundidad del circuito.
   - Visualización de los resultados mediante gráficos y tablas.
   - Interpretación de los resultados para identificar ventajas y limitaciones de las configuraciones iniciales y del algoritmo en general.

5. **Conclusiones:** En esta última etapa, se integraron las principales observaciones del proceso de análisis de datos. Las actividades de esta etapa se describen a continuación.

   - Se resaltan las ventajas de cada configuración inicial y las condiciones bajo las cuales estas fueron más efectivas.
   - Elaboración de recomendaciones para futuros trabajos.

3.2. Liberia Pennylane y códigos desarrollados

Como se ha mencionado, Pennylane es una biblioteca de código abierto basada en Python. Por esta razón se utilizó un compilador de Python, específicamente Jupyter, en el editor de código fuente Visual Studio Code. Para más detalle, se pueden encontrar los códigos en el siguiente enlace `.https://github.com/NICOSPREAM/Algoritmos-Cuanticos`.

4. Conceptos básicos de Mecánica Cuántica

4.1. Estados Cuánticos

4.1.1. Estados Puros

Un estado cuántico de dimensión finita es un vector $|\psi\rangle$ perteneciente a $\mathbb{C}^n$, en donde $\mathbb{C}$ es el campo de los complejos y n es la dimensión (compleja) del espacio vectorial (es decir, el espacio tiene dimensión real igual a $2n$). El caso más simple es de un qubit, el cual puede verse como un elemento de $\mathbb{C}^2$. Por lo tanto, representamos el estado de un qubit como

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \quad a, b \in \mathbb{C}. \tag{1}$$

Notar que hemos representado un vector $\vec{\psi} \in \mathbb{C}^2$ usando la notación de Dirac $|\psi\rangle$. De esta manera, en general, $|\psi\rangle \in \mathbb{C}^n$ será un vector columna de n componentes complejas. Siguiendo la notación de Dirac, a los vectores los llamaremos "kets". Por otra parte, si transponemos un vector $|\psi\rangle$ y luego lo complejo-conjugamos obtenemos lo que se conoce como "bra", $\langle\psi|$. Por ejemplo, el "bra" correspondiente al "ket" (1) es

$$\langle\psi| = \begin{pmatrix} \bar{a} & \bar{b} \end{pmatrix}, \tag{2}$$

en donde $\bar{a}, \bar{b}$ son los coeficientes a y b complejos-conjugados. Consideremos el caso de un qubit, y los siguientes estados

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \quad , \quad \langle\phi| = \begin{pmatrix} \bar{c} & \bar{d} \end{pmatrix}, \tag{3}$$

en donde $a, b, c, d \in \mathbb{C}$. Es fácil ver que

$$|\psi\rangle\langle\phi| = \begin{pmatrix} a\bar{c} & a\bar{d} \\ b\bar{c} & b\bar{d} \end{pmatrix} \quad , \quad \langle\phi|\psi\rangle = a\bar{c} + b\bar{d}. \tag{4}$$

Esto dice que, en el caso general, $|\psi\rangle\langle\phi|$ es una matriz de $n \times n$ componentes complejas (un operador) y que, en cambio, $\langle\phi|\psi\rangle$ es un escalar.

Los estados cuánticos, sin embargo, no son "vectores cualquiera" dentro del espacio de Hilbert, sino que

son vectores normalizados (es decir, viven dentro de una esfera). Para el caso del qubit (1), esto significa que

$$|a|^2 + |b|^2 = 1. \tag{5}$$

Por lo tanto, podríamos parametrizar el estado del qubit diciendo que

$$a = e^{i\varphi}\cos(\theta/2) \quad , \quad b = e^{i\phi}\sin(\theta/2), \tag{6}$$

en donde $\theta \in [0, \pi]$ y $\varphi, \phi \in [0, 2\pi]$ son dos fases cualquiera. Notar que, hasta el momento, tenemos 3 parámetros lo cual significaría que el espacio de Hilbert de un qubit tiene dimensión real igual 3 (y no igual a $2n = 4$, $n = 2$), es decir, se pierde una dimensión por la normalización. Además, debemos identificar a todos los vectores que difieren únicamente por la multiplicación de una fase global, es decir, si $|v\rangle = e^{i\theta}|w\rangle$, entonces los vectores $|v\rangle$ y $|w\rangle$ representan al mismo estado, debido a que una multiplicación de una fase global no afecta los resultados medibles del sistema cuántico, por lo que $e^{i\theta}$ puede ser eliminado. Esto reduce un grado de libertad. Por lo tanto, un estado $|\psi\rangle \in \mathbb{C}^n$ vive en un espacio de dimensión $2n - 2$ (se resta un grado de libertad por la normalización y otro por la identificación de la multiplicación por una fase global). Así, un qubit puede parametrizarse eligiendo $a = \cos(\theta/2)$ y $b = \sin(\theta/2)e^{i\phi}$, es decir, el qubit 1 se puede escribir como

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle,$$

en donde θ es el ángulo polar y ϕ el ángulo azimutal de una esfera de radio 1, conocida como esfera de Bloch. Los estados *puros* están localizados en la superficie de la esfera, como se puede observar en la figura 1. Los vectores $|0\rangle$ y $|1\rangle$ son vectores de una base cualquiera de $\mathbb{C}^2$.

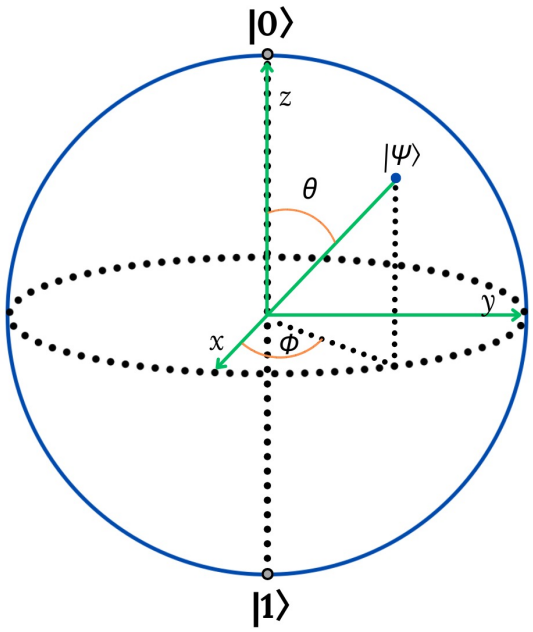


Figura 1: Representación de la esfera de Bloch de un qubit.

Todo lo que se ha señalado es válido para describir estados puros. Además de éstos, existen los estados mezcla, que son representaciones estadísticas de los estados cuánticos (Nielsen y Chuang, 2010).

4.1.2. Estados mezcla

Los estados mezcla se describen mediante matrices densidad ρ, que son operadores Hermíticos, positivos y con traza (suma de los elementos de la diagonal principal de la matriz) igual a 1. Un estado mezcla ρ puede escribirse como

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|,$$

donde p_i son probabilidades, $\sum_i p_i = 1$, y $|\psi_i\rangle$ son estados puros. Los estados mezcla combinan incertidumbres clásicas con incertidumbres cuánticas. En cambio, los estados puros reflejan incertidumbre puramente cuántica. En el caso de un qubit, los estados mezcla corresponden a puntos en el interior de la esfera de Bloch.

4.1.3. Superposición

La superposición es una de las principales propiedades de un sistema cuántico, lo que significa que el sistema puede existir simultáneamente en múltiples estados posibles. Es decir, cualquier combinación lineal (normalizada y con identificación de una fase global) de estados también es un estado posible del sistema cuántico. En computación cuántica, un qubit se escribe como combinación lineal de los estados de la base computacional $|0\rangle$ y $|1\rangle$,

$$|V\rangle = \alpha|0\rangle + \beta|1\rangle.$$

Esto significa que el qubit está en los estados 0 y 1 “al mismo tiempo”, hasta que se realice una medición u observación del qubit. Por otra parte, recordar que $|\alpha|^2 + |\beta|^2 = 1$. Como se verá más adelante, $|a|^2$ es la probabilidad de medir el qubit y obtener el resultado “0”, mientras que $|b|^2$ es la probabilidad de obtener el resultado “1”. Por lo tanto, la condición de normalización nos dice que la suma de probabilidades debe sumar 1. A diferencia de los estados clásicos donde un bit debe ser estrictamente 0 o 1, en la superposición, puede tomar ambas opciones “simultáneamente” (Nielsen y Chuang, 2010).

4.2. Espacio de Hilbert

Formalmente, el espacio $\mathbb{C}^n$ es un espacio de Hilbert de dimensión finita, pues es un espacio vectorial dotado de un producto interno sesqui-lineal. Por lo tanto, a veces escribiremos $\mathbb{C}^n = \mathcal{H}$. Esto es importante para definir conceptos clave como la norma de un vector y la ortogonalidad entre vectores (Nielsen y Chuang, 2010).

4.2.1. Producto interno

El producto interno $\langle v|w\rangle$ es una función que toma dos vectores $|v\rangle$ y $|w\rangle$ y retorna un número complejo. Para que el producto interno sea válido, deben satisfacerse las siguientes tres propiedades:

1. Linealidad en el segundo argumento: $\forall\ \{|\phi_i\rangle \in \mathcal{H}, \gamma_i \in \mathbb{C}\}$ y $|V\rangle \in \mathcal{H}$

$$\langle V| \sum_{i=1}^{N} \gamma_i |\phi_i\rangle = \sum_{i=1}^{N} \gamma_i \langle V|\phi_i\rangle .$$

2. El conjugado del producto interno cambia el orden de los elementos (simetría conjugada o propiedad Hermítica): $\forall\, |V\rangle , |W\rangle \in \mathcal{H}$,

$$\overline{\langle V|W\rangle} = \langle W \mid V\rangle.$$

3. Es positivo, definido: $\forall\, |V\rangle \in \mathcal{H}$,

$$\langle V|V\rangle \geq 0 \quad \text{con igualdad solo si } |V\rangle \text{ es el vector nulo.}$$

El producto interno define una norma (largo) de un vector. Para un vector $|w\rangle$ en el espacio de Hilbert $\mathcal{H}$, la norma se define como la raíz cuadrada de su producto interno:

$$\|w\| = \sqrt{\langle w \mid w\rangle}.$$

4.2.2. Bases Ortonormales (ONB)

Una base de un espacio vectorial V (definido sobre un campo K) es un conjunto de vectores que debe cumplir dos condiciones importantes:

- **Conjunto Generador:** Un conjunto de vectores

$$\mathcal{B} = \{v_1, v_2, \ldots, v_n\} \in V,$$

es un conjunto generador para un espacio vectorial V si cualquier vector en V puede expresarse como una combinación lineal de estos vectores. Es decir, $\forall v \in V$, existen escalares $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \in K$ tales que:

$$v = \sum_{i=1}^{n} \alpha_i v_i.$$

Esto implica que el conjunto $\mathcal{B}$ genera al espacio V. A veces, suele escribirse $V = \text{span}(\mathcal{B})$ para indicar que $\mathcal{B}$ genera al espacio V.

- **Independencia Lineal:** Un conjunto de vectores $\{v_1, v_2, \ldots, v_n\}$ en V es linealmente independiente si ninguno de ellos puede escribirse como una combinación lineal de los demás. Esto significa que, si planteamos la ecuación

$$a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = 0,$$

en donde $\{a_1, a_2, \ldots, a_n\} \in K$, la única solución es aquella en que todos los escalares deben ser cero (solución trivial):

$$a_1 = a_2 = \cdots = a_n = 0.$$

- **Un conjunto ortonormal:** Es un conjunto de vectores $\{v_1, v_2, \ldots, v_n\}$ en un espacio vectorial V que tiene un producto interno y satisface dos condiciones:

  - **Ortogonalidad:** Esto significa que el producto interno entre cualquier par de vectores distintos es cero:

$$\langle v_i \mid v_j \rangle = 0, \quad \forall (i \neq j).$$

  - **Normalización:** Esto significa que el producto interno de un vector consigo mismo tiene norma unitaria, es decir,

$$\langle v_i \mid v_i \rangle = 1, \quad \forall i.$$

Tener un conjunto de vectores ortonormales simplifica los cálculos en el espacio vectorial, ya que cualquier vector en V puede expresarse como una combinación lineal de los vectores de la base, y sus coeficientes son los productos internos del vector con cada uno de los vectores de la base. A partir

de una base cualquiera, se puede generar una base ortonormal usando un procedimiento conocido como proceso de Gram–Schmidt, el cual se detalla en el anexo 9.4.

4.3. Observables

En computación cuántica, un observable es una cantidad física que se puede medir en un sistema cuántico. Los observables describen cantidades físicas medibles, como la energía, el momento o el spin de un electrón (Mermin, 2007).

Los observables se representan como operadores lineales Hermíticos (matrices Hermíticas) que actúan en el espacio de Hilbert. Entonces, si O es un observable, $O\,|\psi\rangle$ (la acción de la matriz sobre el vector) es otro vector en el espacio de Hilbert. El escalar $\langle\psi|O|\psi\rangle$ es el *valor de expectación* del operador O en el estado $|\psi\rangle$.

Un operador O es Hermítico si satisface esta simple condición:

$$O = O^\dagger,$$

donde $O^\dagger$ es la matriz transpuesta conjugada de O. Esto significa que para cualquier par de vectores $|v\rangle$ y $|w\rangle$ en el espacio de Hilbert, se cumple lo siguiente:

$$\langle v|O|w\rangle = \overline{\langle w|O|v\rangle}$$

La Hermiticidad asegura que los valores propios del operador sean números reales.

4.4. Mediciones Cuánticas

Nielsen y Chuang (2010) describe el postulado de la medición cuántica o “colapso de la función de onda”, de la siguiente manera:

> Las mediciones cuánticas se describen mediante una colección de operadores de medición $\{M_m\}$. Estos operadores actúan sobre el espacio de estados del sistema que se está midiendo, y el índice m corresponde a los posibles resultados de la medición.

Las matrices M_m son operadores de medida, asociadas al resultado m de la medición. La matriz $E_m = M_m^\dagger M_m$ es una matriz positiva, que satisface la relación de completitud

$$\sum_m E_m = \mathbb{1}.$$

Las matrices de medición M_m permiten describir el efecto de la medición sobre el sistema y la matriz E_m permite describir las probabilidades de obtener el resultado m. Si el sistema cuántico está en el estado $|\psi\rangle$ antes de la medición, la probabilidad de obtener el resultado m al medir está dada por:

$$p(m) = \langle\psi|E_m|\psi\rangle.$$

De aquí se ve que la relación de completitud significa que las probabilidades suman 1. En efecto,

$$\sum_n p(m) = \sum_n \langle\psi|E_m|\psi\rangle = \langle\psi|\sum_n E_m|\psi\rangle = \langle\psi|\mathbb{1}|\psi\rangle = \langle\psi|\psi\rangle = 1. \tag{7}$$

Como se indica, las matrices de medición (M_m) permiten describir el colapso del estado. En efecto, al obtener el resultado m, el estado del sistema colapsa a:

$$|\psi'\rangle = \frac{M_m|\psi\rangle}{\sqrt{p(m)}}.$$

A este postulado se le conoce como *postulado del colapso*. Por ejemplo, la medición de un qubit en la base computacional $\{|0\rangle, |1\rangle\}$. En este caso, los operadores de medición son matrices proyectivas:

$$M_0 = |0\rangle\langle 0|, \quad M_1 = |1\rangle\langle 1|.$$

Una matriz proyectiva A satisface la condición $A^2 = A$. Por lo tanto, en este caso, $E_0 = M_0^\dagger M_0 = M_0$ y $E_1 = M_1^\dagger M_1 = M_1$. Para un estado $|\psi\rangle = a|0\rangle + b|1\rangle$, las probabilidades de obtener los resultados 0 y 1 son, respectivamente:

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = |a|^2, \tag{8}$$

$$p(1) = \langle\psi|M_1^\dagger M_1|\psi\rangle = \langle\psi|1\rangle\langle 1|\psi\rangle = |b|^2. \tag{9}$$

Si al medir se obtiene el resultado 0, el sistema colapsaría al siguiente estado:

$$|\psi'\rangle = \frac{M_0|\psi\rangle}{\sqrt{p(0)}} = \frac{|0\rangle\langle 0|\psi\rangle}{|a|} = \frac{|0\rangle\, a}{|a|} = |0\rangle.$$

Esto muestra que si al medir se obtiene el resultado 0, el sistema colapsa desde el estado inicial $|\psi\rangle$ al estado $|0\rangle$. Algo análogo ocurre cuando se obtiene el resultado 1, en cuyo caso es fácil ver que el estado colapsaría al estado $|1\rangle$.

4.5. Productos Tensoriales

El producto tensorial es una forma de unir espacios vectoriales para formar espacios más grandes. Esta propiedad es crucial para entender cómo funciona el entrelazamiento y genera una visión un poco más clara acerca de cómo dos o más qubits se encuentran unidos.

Suponga que se tiene un espacio de Hilbert $\mathcal{H}_A$ de un sistema A y un espacio de Hilbert $\mathcal{H}_B$ de un sistema B, el producto tensorial de los espacio se denota como $\mathcal{H}_A \otimes \mathcal{H}_B$ y es un espacio de Hilbert que representa a la combinación del sistema conjunto AB (Nielsen y Chuang, 2010).

Si $|1\rangle \in \mathcal{H}_A$ y $|0\rangle \in \mathcal{H}_B$ son los estados de los subsistemas, el estado conjunto $\in \mathcal{H}_A \otimes \mathcal{H}_B$ se escribe como:

$$|1\rangle \otimes |0\rangle = |10\rangle \,.$$

Por ejemplo, suponga que se trabaja con dos espacios vectoriales V y W, cuyos vectores están definidos como:

$$|v\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \in V, \quad |w\rangle = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \in W.$$

El producto tensorial entre $|v\rangle$ y $|w\rangle$ se calcula como:

$$|v\rangle \otimes |w\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \cdot 3 \\ 1 \cdot 4 \\ 2 \cdot 3 \\ 2 \cdot 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}.$$

Esto muestra que al hacer el producto tensorial entre dos espacios (o de dos vectores) la dimensión del espacio total se incrementa, pues corresponde a la multiplicación de las dimensiones de los subespacios (es decir, en general, la dimensión del espacio total aumenta exponencialmente con el número de subespacios o subsistemas). Por esto, el espacio de Hilbert de n partículas (o n qubits en lenguaje computacional) tiene dimensión 2^n. Por ejemplo, si tenemos un “q-byte” (8 qubits), el espacio de Hilbert de las 8 partículas tiene dimensión $2^8 = 256$.

Además, el producto tensorial debe satisfacer las siguientes propiedades básicas:

1. Para un escalar z y vectores $|v\rangle \in V$ y $|w\rangle \in W$, se cumple la siguiente propiedad:

$$z\big(|v\rangle \otimes |w\rangle\big) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle).$$

Esta propiedad establece que el producto tensorial respeta la multiplicación por escalares, lo que significa que el escalar puede aplicarse a cualquiera de los dos vectores sin alterar el resultado.

Por ejemplo, si se multiplica $|v\rangle$ por un escalar $z = 2$, se obtiene:

$$z\big(\,|v\rangle \otimes |w\rangle\,\big) = 2 \cdot \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 12 \\ 16 \end{pmatrix}.$$

Esto es equivalente a aplicar el escalar z al vector $|v\rangle$ antes del producto tensorial:

$$(z\,|v\rangle) \otimes |w\rangle = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \cdot 3 \\ 2 \cdot 4 \\ 4 \cdot 3 \\ 4 \cdot 4 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 12 \\ 16 \end{pmatrix}.$$

2. Para vectores $|v_1\rangle, |v_2\rangle \in V$ y $|w\rangle \in W$, se cumple:

$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle.$$

El producto tensorial es distributivo respecto a la suma de vectores en el primer espacio V. Esto significa que la suma puede calcularse primero y luego aplicarse el producto tensorial, o viceversa.

Por ejemplo, suponga que $|v_1\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ y $|v_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Si se calcula la suma de estos vectores:

$$|v_1\rangle + |v_2\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

El producto tensorial distribuye sobre esta suma:

$$\big(\,|v_1\rangle + |v_2\rangle\,\big) \otimes |w\rangle = \begin{pmatrix}1\\3\end{pmatrix} \otimes \begin{pmatrix}3\\4\end{pmatrix} = \begin{pmatrix}1\cdot 3\\1\cdot 4\\3\cdot 3\\3\cdot 4\end{pmatrix} = \begin{pmatrix}3\\4\\9\\12\end{pmatrix}.$$

Esto es equivalente a distribuir el producto tensorial:

$$\big(\,|v_1\rangle + |v_2\rangle\,\big) \otimes |w\rangle = \big(\,|v_1\rangle \otimes |w\rangle\,\big) + \big(\,|v_2\rangle \otimes |w\rangle\,\big),$$

donde:

$$|v_1\rangle \otimes |w\rangle = \begin{pmatrix}1\\2\end{pmatrix} \otimes \begin{pmatrix}3\\4\end{pmatrix} = \begin{pmatrix}3\\4\\6\\8\end{pmatrix},$$

y:

$$|v_2\rangle \otimes |w\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}3\\4\end{pmatrix} = \begin{pmatrix}0\\0\\3\\4\end{pmatrix}.$$

Sumando ambos resultados:

$$\begin{pmatrix}3\\4\\6\\8\end{pmatrix} + \begin{pmatrix}0\\0\\3\\4\end{pmatrix} = \begin{pmatrix}3\\4\\9\\12\end{pmatrix}.$$

4.6. Entrelazamiento

El entrelazamiento cuántico es un fenómeno de la mecánica cuántica en el cual dos o más partículas se encuentran en un estado tal que la descripción del estado de una partícula no puede hacerse independientemente del estado de la otra, incluso si las partículas están separadas por grandes distancias. Esto significa que las propiedades de una partícula están fuertemente correlacionadas con la de la otra, de modo que conocer el estado de una nos proporciona automáticamente información sobre el estado de la otra, sin importar la distancia entre ellas.

Matemáticamente, un estado entrelazado de n partículas, $|\Psi\rangle$, es un estado que **no** se puede factorizar como el producto tensorial de estados de cada particula individual. En otras palabras, si el estado de las n

partículas se puede escribir como el producto de los estados individuales de cada partícula,

$$|\Psi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \ldots \otimes |\psi\rangle_n \,, \tag{10}$$

entonces el estado $|\Psi\rangle$ se dice separable o no-entrelazado. Cuando hay dos partículas, un conjunto de 4 estados entrelazados está formado por los estados conocidos como "estados de Bell" (Bell, 1964; Zeilinger, 2002), muy importantes en protocolos de comunicación, criptografía, teleportación cuántica, entre otros. Son los siguientes:

$$|B_{00}\rangle = |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{11}$$

$$|B_{01}\rangle = |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{12}$$

$$|B_{10}\rangle = |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{13}$$

$$|B_{11}\rangle = |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{14}$$

Un estado entrelazado de n partículas es el estado GHZ, o de Greenberger-Horne-Zeillinger , definido como:

$$|GHZ\rangle = \frac{|0\rangle_1 \otimes |0\rangle_2 \otimes \ldots \otimes |0\rangle_n + |1\rangle_1 \otimes |1\rangle_2 \otimes \ldots \otimes |1\rangle_n}{\sqrt{2}} = \frac{|00\ldots0\rangle + |11\ldots1\rangle}{\sqrt{2}}, \tag{15}$$

es decir, corresponde a una superposición de dos estados: en el primero todas las partículas están en el estado $|0\rangle$ y en segundo en el estado $|1\rangle$ (Greenberger et al., 2007).

El entrelazamiento cuántico es crucial para el desarrollo de tecnologías emergentes como la computación cuántica, que aprovecha las correlaciones entre partículas entrelazadas para realizar cálculos mucho más rápidos y eficientes que los que podrían lograrse con la computación clásica.

4.7. Evolución de estados: operadores unitarios

La evolución en el tiempo de un estado cuántico está dictada por la ecuación de Schrödinger,

$$i\hbar \frac{\partial}{\partial t} |\psi\rangle = H |\psi\rangle \,, \tag{16}$$

en donde el operador H es el Hamiltoniano (la energía del sistema) y $\hbar$ es la constante de Planck. La solución a la ecuación de Schrödinger está dada por

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle\,, \tag{17}$$

en donde $|\psi(t)\rangle$ representa el estado en el tiempo t, $|\psi(0)\rangle$ es el estado inicial del sistema cuántico y $U(t)$ es un operador *unitario* generado por el Hamiltoniano del sistema que, al aplicarse sobre estado inicial, genera el estado al tiempo t, $|\psi(t)\rangle$. Cuando el Hamiltoniano no depende del tiempo, el operador $U(t)$ puede escribirse como una exponencial compleja

$$U(t) = \exp\{(-i/\hbar)H \cdot t\}, \tag{18}$$

lo cual muestra que las soluciones son de tipo ondulatorio (ya que una exponencial compleja puede escribirse como suma de senos y cosenos). Un operador U es unitario cuando satisface

$$U^\dagger U = UU^\dagger = \mathbb{1}.$$

Un operador unitario tiene la importante propiedad de conservar la norma de los vectores sobre los que actúa. Si $|\psi\rangle$ es un vector en el espacio de Hilbert, entonces $\|U|\psi\rangle\| = \||\psi\rangle\|$. En la computación cuántica, los operadores unitarios describen las evoluciones de los estados cuánticos a través de las compuertas cuánticas de un circuito, las cuales son todas unitarias (Mermin, 2007).

4.8. Circuitos y compuertas cuánticas

Un circuito cuántico es una secuencia de compuertas lógicas aplicadas sobre un conjunto de qubits. En un circuito, los qubits se representan como “buses” o líneas del circuito, mientras que las compuertas se representan como cajas, con entradas y salidas. Las compuertas representan operaciones unitarias ejecutadas sobre los qubits, lo que significa que modifican su estado de forma unitaria.

Existen compuertas que operan sobre un solo qubit y otras que operan sobre dos qubits. Estas compuertas se utilizan para implementar ciertas operaciones cuánticas, como rotaciones sobre un eje hasta entrelazamiento.

4.8.1. Compuertas de un solo qubit

Las compuertas que actúan sobre un solo qubit realizan transformaciones unitarias que pueden representar mediante matrices de 2×2. Las compuertas más utilizadas son las siguientes (Nielsen y Chuang, 2010):

1 **Compuerta Hadamard** (H):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

2 **Compuerta Pauli-X** (X):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

3 **Compuerta Pauli-Y** (Y):

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

4 **Compuerta Pauli-Z** (Z):

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

5 **Rotaciones**: realiza rotaciones en la esfera de Bloch, en particular, ocupa tres importantes rotaciones gracias a las matrices de Pauli que se vieron anteriormente, estas se definen como: $R_x(\theta), R_y(\theta), R_z(\theta)$ cuyo subíndice indica el eje de rotación. Se definen de la siguiente manera:

- Rotación Eje X

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}.$$

- Rotación Eje Y

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}.$$

- Rotación Eje Z

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}.$$

Las compuertas H, X, Y, Z además de ser unitarias son Hermíticas. Esto significa que $H^2 = X^2 = Y^2 = Z^2 = \mathbb{1}$, de modo que si en un circuito se aplican dos veces de modo consecutivo no alteran el estado inicial.

4.8.2. Compuertas de dos qubits

Compuerta U-controlada: si el qubit de control está en el estado $|0\rangle$, no se realiza ninguna operación sobre el qubit objetivo. En cambio, si el control está en el estado $|1\rangle$, el qubit objetivo se modifica, aplicándole el operador unitario U, es decir, su estado cambia según $|\psi\rangle \rightarrow U|\psi\rangle$. Se puede ver representado en la siguiente figura 2. Uno de los más utilizados y que se ocupan en este trabajo es la Compuerta CNOT (Controlled-NOT) o negación controlada, que aplica la misma lógica. Sin embargo, para denotar la negación controlada en el circuito se realizará como se ve en la figura 3.

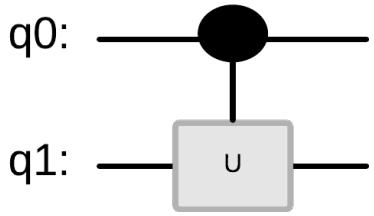


Figura 2: Representación gráfica del funcionamiento del U-controlada. La línea superior es el qubit de control y la línea inferior el qubit objetivo.

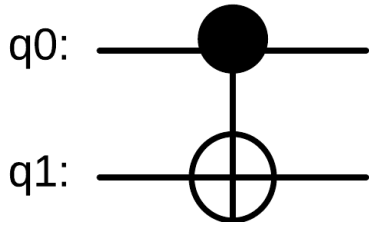


Figura 3: Representación de la compuerta CNOT. La línea superior representa el qubit de control y la línea inferior el qubit objetivo.

En términos matriciales, la compuerta C-NOT se representa en la base computacional a través de una matriz unitaria de 4×4,

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{19}$$

Por ejemplo, si se considera la acción de la compuerta CNOT sobre el estado de dos qubits, en el que el qubit de control está en el estado $|0\rangle_C$ y el qubit objetivo en un estado general $a|0\rangle_T + b|1\rangle_T$, es decir,

sobre el estado producto $|\Psi\rangle = |0\rangle_C (a\,|0\rangle_T + b\,|1\rangle_T)$, se obtiene:

$$\text{CNOT}\,|\Psi\rangle \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} = |\Psi\rangle\,. \tag{20}$$

Esto muestra que la compuerta no realiza ninguna acción sobre el qubit objetivo (pues el control está apagado). En cambio, si se considera la acción de la compuerta sobre el mismo estado, pero ahora con el qubit de control en el estado $|1\rangle_C$, es decir, sobre el estado producto $|\Psi\rangle = |1\rangle_C (a\,|0\rangle_T + b\,|1\rangle_T)$, se obtiene:

$$\text{CNOT}\,|\Psi\rangle \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ b \\ a \end{pmatrix} = |1\rangle_C (b\,|0\rangle_T + a\,|1\rangle_T), \tag{21}$$

lo que muestra que el qubit objetivo fue ahora "negado", es decir, se intercambiaron los coeficientes a y b. La compuerta CNOT es una compuerta que permite entrelazar dos qubits y son difíciles de construir en la práctica. Informalmente, se podría decir que el CNOT es el análogo cuántico al transistor (Nielsen y Chuang, 2010).

5. Problema Max-Cut

En esta tesis se considera el problema Max-Cut, un problema clásico de optimización combinatorial que es NP-Hard o NP-Completo y tiene grandes aplicaciones en diversas áreas como clustering, diseños de VLSI[3] y en física estadística (IBM, 2023).

Sea $G = \{V, E\}$ un grafo formado por vértices pertenecientes al *conjunto de vértices* V, unidos por aristas que pertenecen al *conjunto de aristas* E. El número de nodos (orden) de G está dado por $N = |V|$. El grado de cada nodo corresponde al número de vecinos. En este trabajo, se consideran grafos no-dirigidos y $K-$regulares.

El problema Max-Cut es un problema clásico de optimización en teoría de grafos que consiste en particionar o cortar el grafo en dos subconjuntos de nodos disjuntos S y T, de modo que se maximice el peso

[3]Proceso para crear circuitos integrados con múltiples transistores en un solo chip.

total de las aristas que conectan un vértice en S con un vértice en T. A cada nodo $u \in V$ se le asocia una variable binaria $x_u \in \{1, -1\}$ (Goemans y Williamson, 1995). De este modo, el objetivo consiste en maximizar la función de costos:

$$C(x) = \sum_{(u,v)\in E} \frac{(1 - x_u x_v)}{2} \cdot w_{(u,v)}, \tag{22}$$

donde:

- $(u, v) \in E$ denotan las aristas del grafo G.
- $0 \leq w_{(u,v)} \leq 1$ es el peso de la arista $(u, v) \in E$.
- $x_u, x_v \in \{-1, 1\}$.

Una forma trivial de resolver el problema Max-cut consiste en enumerar todas las posibles soluciones y luego buscar aquella(s) que tiene(n) el mayor número de cortes, donde el criterio de corte viene dado por: Si $x_u \neq x_v$ se realiza el corte, en caso contrario $x_u = x_v$ no se realiza el corte. A este procedimiento se le dirá *búsqueda exhaustiva*. Para un grafo de tamaño N, la lista de posibles soluciones tiene 2^N elementos de manera que el tiempo de ejecución crece exponencialmente con el tamaño del grafo (Bharti et al., 2022).

En la figura 4 se presenta un grafo de orden $N = 5$ y grado $K = 2$, cuyos pesos son todos iguales a 1. Las posibles $2^N = 32$ soluciones del grado y sus correspondientes costos se enumeran en la tabla 1, que resume todos los *bitstrings* posibles.

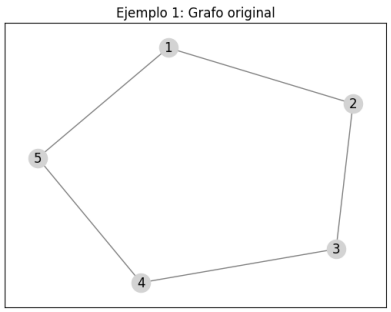


Figura 4: Grafo G no-dirigido y regular, de tamaño $N = 5$ y grado $K = 2$, en donde todas las aristas tienen el mismo peso, $w_{(i,j)} = 1 \; \forall (i, j) \in E$.

x_1	x_2	x_3	x_4	x_5	$C(x_1, x_2, x_3, x_4, x_5)$
1	1	1	1	1	0
1	1	1	1	-1	2
1	1	1	-1	1	2
1	1	1	-1	-1	2
1	1	-1	1	1	2
1	1	-1	1	-1	4
1	1	-1	-1	1	2
1	1	-1	-1	-1	2
1	-1	1	1	1	2
1	-1	1	1	-1	4
1	-1	1	-1	1	4
1	-1	1	-1	-1	4
1	-1	-1	1	1	2
1	-1	-1	1	-1	4
1	-1	-1	-1	1	2
1	-1	-1	-1	-1	2
-1	1	1	1	1	2
-1	1	1	1	-1	2
-1	1	1	-1	1	4
-1	1	1	-1	-1	2
-1	1	-1	1	1	4
-1	1	-1	1	-1	4
-1	1	-1	-1	1	4
-1	1	-1	-1	-1	2
-1	-1	1	1	1	2
-1	-1	1	1	-1	2
-1	-1	1	-1	1	4
-1	-1	1	-1	-1	2
-1	-1	-1	1	1	2
-1	-1	-1	1	-1	2
-1	-1	-1	-1	1	2
-1	-1	-1	-1	-1	0

Tabla 1: Tabla con todas las combinaciones de x_1, x_2, x_3, x_4, x_5, donde sus soluciones son $\{1, 3 \in T\}$, $\{2, 4, 5 \in S\}$.

En la tabla 1 se observa que existen diez soluciones (destacadas en amarillo), correspondientes a los bistrings “11-11-1” , “1-111-1”, “1-11-11”, “1-1-1-1”,“1-1-11-1”,“-111-11”,“-11-111”,“-11-11-1”,“-11-1-11”,“-1-11-11” que definen un total de $C = 4$ cortes. Para cada solución, los nodos 1 y 3 se agrupan en un mismo subgrupo (S) mientras que los nodos 2, 4, 5 en segundo subgrupo (T). Las soluciones se representan gráficamente en la figura 5.

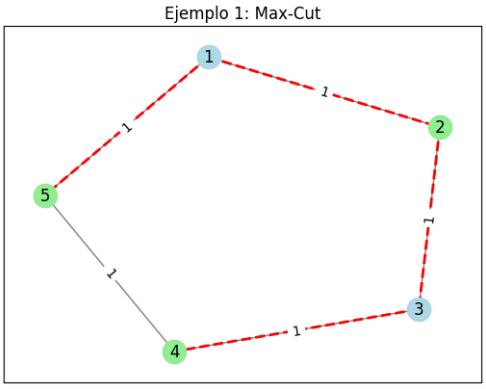


Figura 5: Solución del grafo de orden $N = 5$ y grado $K = 2$.

6. QAOA para Max-Cut

6.1. Historia del algoritmo Quantum Approximate Optimization Algorithm

El Algoritmo QAOA es una innovación reciente en el campo de la computación cuántica, diseñado para abordar problemas de optimización combinatorial de alta complejidad. Su desarrollo se remonta al año 2014, cuando Edward Farhi (2014) y sus colaboradores presentaron por primera vez este algoritmo como una alternativa a los métodos clásicos, que enfrentan dificultades exponenciales en su tiempo de ejecución para grafos complejos. El QAOA combina principios de la mecánica cuántica, como la superposición y el entrelazamiento, con optimización clásica para encontrar soluciones aproximadas eficientes en problemas de gran tamaño.

El diseño del QAOA está inspirado en la computación cuántica adiabática, donde un sistema permanece en el estado fundamental mientras un Hamiltoniano evoluciona lentamente. Sin embargo, QAOA adopta un enfoque no adiabático[4] más adecuado para dispositivos cuánticos ruidosos y de escala intermedia (NISQ, por sus siglas en inglés), permitiendo una implementación más práctica dada las tecnologías actuales (Preskill, 2018). La estructura básica del algoritmo alterna entre la aplicación de una operación unitaria asociada a un Hamiltoniano de costo, que codifica la función objetivo del problema, y una operación unitaria asociada a un Hamiltoniano mezclador, que explora el espacio de soluciones posibles (Farhi et al. ,2014; Hadfield et al., 2019). La función de costos del problema Max-Cut se puede mapear a un Hamiltoniano (tipo modelo de Ising), facilitando su implementación mediante QAOA (Lucas, 2014).

[4]Denominado comúnmente como computación basada en compuertas lógicas.

El QAOA se ha utilizado ampliamente en simuladores y hardware cuántico para resolver problemas como el Max-Cut, asignación de recursos y clustering. Además, investigaciones recientes han explorado su uso en áreas como el aprendizaje automático y la preparación de estados cuánticos (Hadfield et al., 2019). Aunque su capacidad para superar a los algoritmos clásicos aún está en investigación, el QAOA representa un paso clave para realizar un proceso más práctico en la computación cuántica.

6.2. Algoritmo Quantum Approximate Optimization Algorithm (QAOA)

Dada la complejidad del problema, los métodos clásicos para resolverlo tienen limitaciones en términos de tiempo de ejecución y precisión cuando se aplican a grafos de gran tamaño, debido a su complejidad exponencial $O(2^n)$ (n denota el número de nodos grafo). Esto hace que sea muy complicado encontrar una solución cercana a la solución óptima en un tiempo razonable.

En este contexto, se explora el uso del Quantum Approximate Optimization Algorithm, una de las técnicas en la computación cuántica para abordar problemas de optimización combinatorial. QAOA utiliza una combinación de circuitos cuánticos parametrizados, aprovechando el principio fundamental de la superposición cuántica para la exploración de múltiples soluciones simultáneamente, además, de métodos clásicos de optimización para aproximar a la solución óptima del problema Max-Cut.

El objetivo de este trabajo es desarrollar y analizar la implementación de QAOA para resolver el problema de Max-Cut en diferentes instancias de grafos. En esta sección, se presentarán los fundamentos teóricos del algoritmo, detallando la construcción del Hamiltoniano de costo y el Hamiltoniano mezclador, así como el proceso de optimización de los parámetros cuánticos. Posteriormente, se llevará a cabo una serie de experimentos utilizando el simulador cuántico de Pennylane, para evaluar el rendimiento del algoritmo en términos de la calidad de las soluciones y la eficiencia computacional.

6.2.1. Hamiltoniano de costo

En el algoritmo QAOA, la función de costo (22) **se puede mapear a un Hamiltoniano de costo** H_c, que es operador Hermitíco que actúa sobre un espacio de Hilbert. Este espacio contiene los nodos del grafo, que son representados como qubits. Esto hace que pasemos “del mundo clásico al cuántico”. A partir de H_c, se construye un operador unitario de control $U_c = e^{-i\gamma H_c}$, que se implementa a través de compuertas de un solo qubit y de dos qubits (CNOT). El Hamiltoniano de costo es:

$$H_c = \sum_{(u,v)\in E} w_{uv} \frac{1 - Z_u Z_v}{2}, \tag{23}$$

donde

- Z_u y Z_v son los operadores de Pauli-Z que actúan sobre los qubits correspondientes a los nodos u y v del grafo.
- $(u, v) \in E$ es el conjunto de aristas del grafo.
- $0 \leq w_{uv} \leq 1$ representa el peso de la arista entre los nodos u y v.

Este Hamiltoniano se aplica sobre el espacio cuántico donde cada nodo del grafo está representado por un qubit. El operador Z tiene dos auto-valores posibles: $+1$ o -1. Si Z_u y Z_v son iguales, entonces el término $\frac{1-Z_uZ_v}{2}$ vale cero (no hay corte de la arista que une los qubits). En cambio, si Z_u y Z_v son distintos, entonces el término $\frac{1-Z_uZ_v}{2}$ vale 1 (sí hay corte de la arista que une los qubits). Al sumar sobre todas las aristas se obtiene el total de cortes.

La función de costos (22) es un autovalor de H_c, es decir,

$$H_c|x\rangle = C(x)|x\rangle. \tag{24}$$

Consecuentemente $C(x)$ corresponde al autovalor del Hamiltoniano de costo para el estado $|x\rangle$. Por ejemplo, si se tiene un grafo de 3 nodos, en donde cada nodo está unido a los otros dos restantes por aristas de distinto peso (imagine un triángulo). Si el nodo 1 se deja en un subgrupo (está en el estado $|0\rangle$) y los otros dos nodos se agrupan en otro subgrupo (están en el estado $|1\rangle$), el estado conjunto de los 3 nodos (3 qubits) sería

$$|x\rangle = |0\rangle_1 \otimes |1\rangle_1 \otimes |1\rangle_1 = |011\rangle\,. \tag{25}$$

El Hamiltoniano de costos para este sencillo grafo sería

$$H_c = \frac{(1 - Z_1Z_2)}{2} + \frac{(1 - Z_1Z_3)}{2} + \frac{(1 - Z_2Z_3)}{2}. \tag{26}$$

Al aplicar H_c sobre $|x\rangle$ se obtiene, $H_c\,|x\rangle = 2\,|x\rangle$. Por lo tanto, $C(x) = 2$ (2 cortes) para el bitstring $x = 011$. Así, en general, $C(x)$ representa el costo de la partición representada por el bitstring $|x\rangle$ en el grafo (Zhou et al., 2020).

6.2.2. Función Objetivo del QAOA

El objetivo del algoritmo es maximizar el valor esperado del Hamiltoniano de costo, lo que se logra ajustando los parámetros de un circuito cuántico a través de un proceso iterativo. Específicamente, el algoritmo

busca maximizar la expectativa del Hamiltoniano de costo, definido como:

$$\max_{\gamma,\beta} F_p(\gamma,\beta) = \langle\psi(\gamma,\beta)|H_c|\psi(\gamma,\beta)\rangle, \tag{27}$$

donde $|\psi(\gamma,\beta)\rangle$ es el estado cuántico del sistema a la salida del circuito cuántico (ver sección 6.2.3), que depende de los vectores de parámetros γ y β que se ajustan iterativamente por medio de una optimización clásica de la función objetivo. Este valor esperado mide cuán bien una configuración de qubits $|\psi(\gamma,\beta)\rangle$ se aproxima a la solución óptima del problema Max-Cut.

En este contexto, H_c es el Hamiltoniano de costo que encapsula la función de costo del problema de Max-Cut, y la maximización se realiza iterativamente ajustando los parámetros del circuito cuántico del algoritmo QAOA. El estado final $|\psi(\gamma,\beta)\rangle$ es una aproximación al autoestado del sistema que corresponde al valor máximo de H_c (Kostas, 2023).

Cada capa (layer) del algoritmo QAOA consiste en la aplicación de un unitario de control seguida de la aplicación de un unitario de mezcla. El operador unitario de control está definido como:

$$U_C(\gamma) = e^{-i\gamma H_c},$$

donde:

- H_c es un Hamiltoniano que describe la función objetivo visto en la ecuación 23.
- γ es un parámetro variacional.

Recordar que H_c es un operador Hermítico, haciendo que $U_C(\gamma)$ sea un unitario. El operador se implementa través de 2 compuertas, la compuerta CNOT y una compuerta local σ_z[5] (Farhi et al., 2014).

Por otra parte, el Hamiltoniano de mezcla se define como:

$$H_m = \sum_{i=1}^{N} \sigma_{x,i},$$

donde $\sigma_{x,i}$ es una matriz de de Pauli X que se aplica sobre el qubit i. A partir de este Hamiltoniano, definimos el operador unitario de mezcla

$$U_M(\beta) = e^{-i\beta H_M},$$

en donde β es un parámetro variacional que regula la intensidad de la exploración. El Hamiltoniano de mezcla tiene como objetivo evitar quedar en mínimos locales. Esto genera transiciones entre los estados de $|0\rangle$ y $|1\rangle$ de cada qubit. Además, H_M también es un operador Hermítico, lo que hace que $U(\beta)$ sea

[5]Esta genera una rotación en Pauli Z en la esfera de Bloch

unitario. Estos operadores permiten al algoritmo explorar el espacio de soluciones y encontrar los bitstrings que optimizan la funcion de costo (Academia EITCA, 2024). Esto define el *ansatz* del algoritmo, es decir, la forma en que se explorara el espacio de Hilbert para buscar posibles soluciones.

6.2.3. Circuito Quantum Approximate Algorithm (QAOA)

Este algoritmo utiliza un circuito cuántico parametrizado que alterna entre capas de operadores relacionadas con el problema Max-Cut y utiliza Hamiltoniano de costo, operador de control y operador de mezcla. Ahora se describe detalladamente su estructura y los componentes principales del circuito.

1- Inicialización

El circuito comienza inicializando los estados cuánticos, en este apartado se tienen 3 diferentes tipos de inicializaciones, el de superposición, estado de Bell y el estado de GHZ.

1.1- **Inicialización en superposición**

Esta es la inicialización más común en el algoritmo QAOA, donde se realiza una superposición balanceada de todos los qubits (Kostas et al., 2023), denotada de la siguiente manera:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \,,$$

donde:

- $|s\rangle$ es el estado de superposición
- n es el número de qubits $\in |V|$

Este estado se genera aplicando compuertas Hadamard (H) a cada qubit en el estado inicial ($|0\rangle$), en la figura 6 se observa más gráficamente.

$$H\,|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

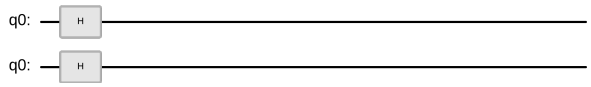


Figura 6: Representación de la inicialización en superposición aplicando una Hadamard a cada qubit.

1.2- **Inicialización en estado de Bell**

Los estados de Bell son un conjunto de cuatro estados cuánticos máximamente entrelazados que forman una base ortonormal en el espacio de Hilbert de dos qubits. Cada uno de estos estados se puede describir mediante una combinación lineal de los estados base $|00\rangle$, $|01\rangle$, $|10\rangle$,

$|11\rangle$ de una manera resumida se pueden escribir de la siguiente manera las inicializaciones:

$$|B_{00}\rangle = |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{28}$$

$$|B_{01}\rangle = |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{29}$$

$$|B_{10}\rangle = |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{30}$$

$$|B_{11}\rangle = |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{31}$$

Pero la más común y recurrente es la ecuación 28 y esa es la que se utilizará en este ejercicio. Entonces, para preparar el estado de Bell $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, el circuito utiliza las siguientes compuertas cuánticas:

1. Aplica una compuerta de Hadamard (H) al primer qubit:

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

2. Aplica una compuerta CNOT al par de qubits, donde el primer qubit actúa como el control y el segundo como el objetivo:

$$\text{CNOT} : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

El circuito correspondiente se ve así:

$$\text{Circuito: } H \longrightarrow \text{CNOT}.$$

Este estado de Bell entrelaza completamente los dos qubits, en la figura 7 se observa el circuito y su interacción explicada anteriormente. El estado de Bell es útil para aplicaciones de comunicaciones cuánticas (Zeilinger, 2002; Bell, 1964).

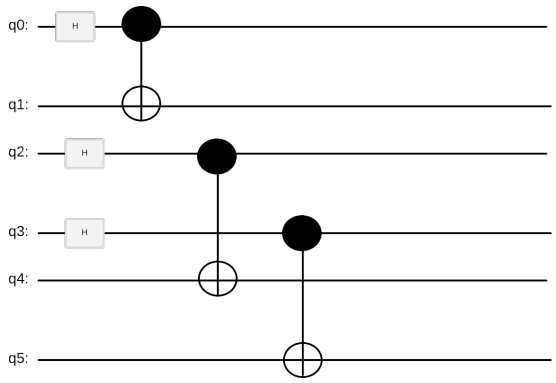


Figura 7: Diagrama del Circuito de Bell con seis qubits. Los qubits entrelazados en este diagrama serían los siguientes $q0 - q1$;$q2 - q4$; $q3 - q5$.

1.3- **Inicialización en estado GHZ**

El estado GHZ es una superposición cuántica de tres qubits o más, en la que los qubits están entrelazados de manera simétrica y correlacionada. A diferencia de los estados de Bell, que involucran dos qubits, el estado GHZ es invariante bajo la permutación de los qubits, lo que significa que no hay una distinción entre ellos. Las mediciones de los qubits en este estado están fuertemente correlacionadas, de tal manera que, al medir uno, los otros dos se determinan automáticamente, lo que demuestra las propiedades no locales de la mecánica cuántica . El el estado GHZ se prepara de la siguiente manera:

- Aplica una compuerta de Hadamard (H) al primer qubit:

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

- Aplica compuertas CNOT en cascada, donde el primer qubit actúa como control para los demás:

$$\text{CNOT} : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \otimes |0\rangle \rightarrow \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle).$$

El circuito se ve así para tres qubits:

$$\text{Circuito: } H \longrightarrow \text{CNOT } (1 \rightarrow 2) \longrightarrow \text{CNOT } (2 \rightarrow 3).$$

Este estado realiza un entrelazamiento entre todos los qubits del sistema y se puede observar más claramente en la siguiente figura 8 (Mermin, 2007; Greenberger et al., 2007).

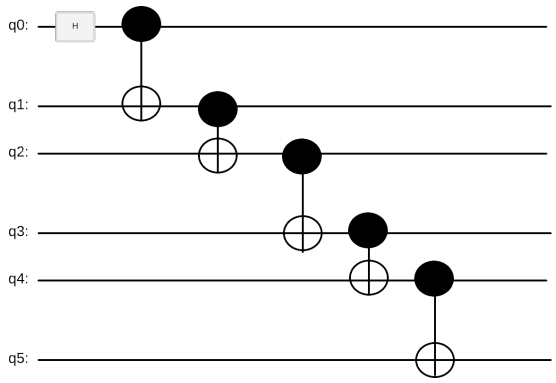


Figura 8: Diagrama del funcionamiento del circuito GHZ con seis qubits. El estado generado en este ejemplo es el siguiente: $\frac{1}{\sqrt{2}}\left(|000000\rangle + |111111\rangle\right)$.

2- Construcción del circuito

El circuito de QAOA se construye alternando entre capas de operadores unitarios asociados con el Hamiltoniano de costo y el Hamiltoniano de mezcla (Kostas et al., 2023).

- **Capa de Costo ($U_C(\gamma)$):**

  La capa de costo se implementa mediante el operador unitario de control

$$U_C(\gamma) = e^{-i\gamma H_C}.$$

  Para el problema Max-Cut, cada término del Hamiltoniano se traduce en compuertas de rotación controladas $R_{Z_i Z_j}$ se puede ver como se aplica en la figura 9 en el circuito cuántico:

$$R_Z(-2\gamma w_{ij}) = e^{i\gamma w_{ij} Z_i Z_j},$$

  aplicadas a cada par de quibit.

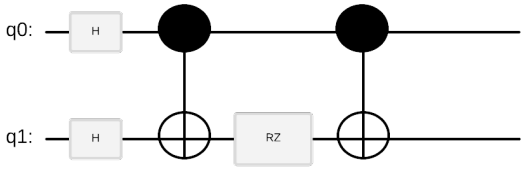


Figura 9: Representación del circuito con 2 qubits aplicando Hadamard y el unitario de control.

- **Capa de Mezcla ($U_M(\beta)$):**

  La capa de mezcla se implementa mediante:

$$U_M(\beta) = e^{-i\beta H_M},$$

Como bien se mencionó anteriormente este realiza rotaciones alrededor del eje x en la esfera de Bloch para cada qubit, en la siguiente figura 10 se observa su implementación en el circuito cuántico.:

$$R_X(2\beta) = e^{-i\beta\sigma_X}.$$

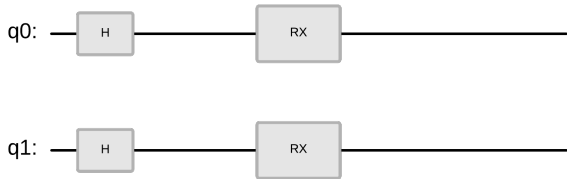


Figura 10: Representación del circuito de dos qubits aplicando el unitario de mezcla.

- **Circuito Completo:**

  El circuito alterna p capas de $U_C(\gamma_k)$ y $U_M(\beta_k)$ como se observa en la figura 11 y en la figura 12 se observa la aplicación del unitario de control y de mezcla respectivamente. por lo tanto el sistema se traduce de la siguiente manera:

$$|\psi_p(\vec{\gamma}, \vec{\beta})\rangle = U_M(\beta_p)U_C(\gamma_p)\dots U_M(\beta_1)U_C(\gamma_1)\,|s\rangle\,,$$

  donde $\vec{\gamma} = (\gamma_1, \dots, \gamma_p)$ y $\vec{\beta} = (\beta_1, \dots, \beta_p)$ son los parámetros variacionales del circuito.

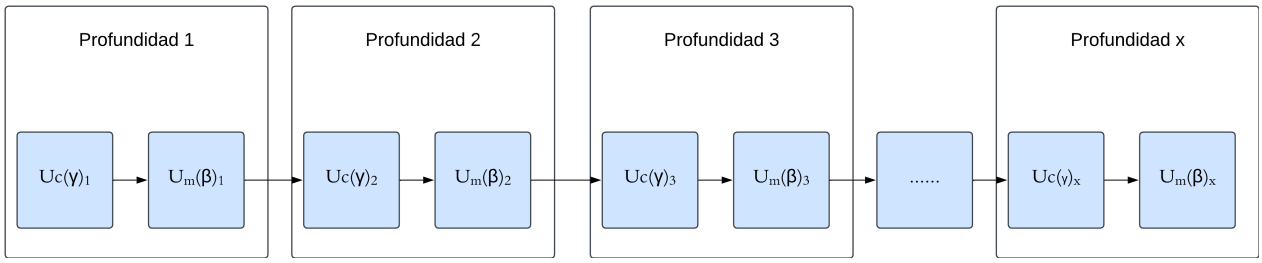


Figura 11: Representación de las profundidades del circuito QAOA.

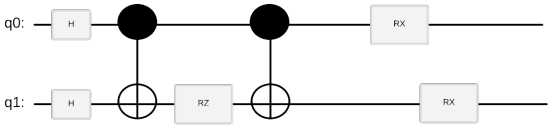


Figura 12: Representación del circuito completo de dos qubits aplicando el unitario de control y de mezcla.

3- Medición y optimización

Después de construir el circuito, se procede a medir en la base computacional y a optimizar los parámetros variacionales para encontrar la solución óptima al problema.

- **Medición:**

  Se mide el valor esperado del Hamiltoniano de costo H_C en el estado $|\psi_p(\vec{\gamma},\vec{\beta})\rangle$:

$$F_p(\vec{\gamma},\vec{\beta}) = \langle\psi_p(\vec{\gamma},\vec{\beta})|\, H_C\, |\psi_p(\vec{\gamma},\vec{\beta})\rangle\,.$$

  Este valor corresponde al costo asociado a los parámetros $\vec{\gamma}$ y $\vec{\beta}$.

- **Optimización Clásica:**

  Los parámetros $\vec{\gamma} = (\gamma_1,\ldots,\gamma_p)$ y $\vec{\beta} = (\beta_1,\ldots,\beta_p)$ se ajustan mediante un optimizador clásico que maximiza la función objetivo $F_p(\vec{\gamma},\vec{\beta})$. Este proceso se realiza de forma iterativa.

- **Solución Final:**

  Al final de la optimización, los valores óptimos de los parámetros $(\vec{\gamma}^*,\vec{\beta}^*)$ definen el estado final:

$$|\psi_p(\vec{\gamma}^*,\vec{\beta}^*)\rangle\,.$$

  Este estado codifica la solución aproximada al problema de optimización original, en la figura 13 se puede observar el procedimiento completo del circuito QAOA.

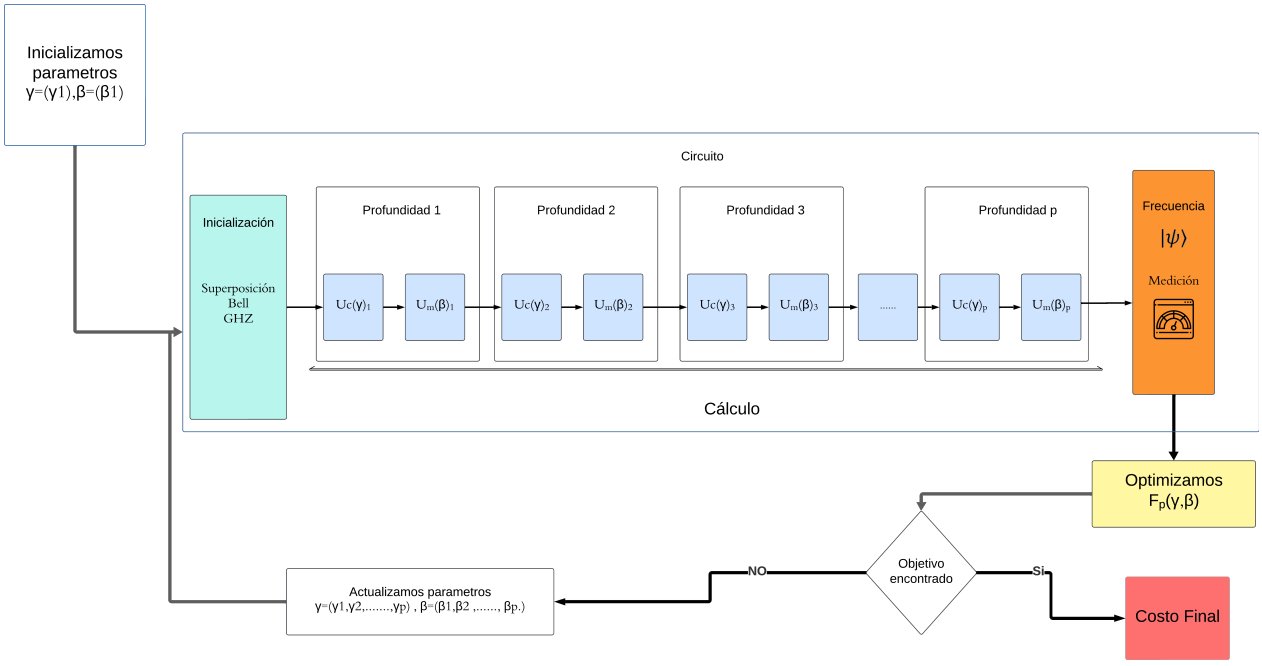


Figura 13: Diagrama del procedimiento QAOA en base al problema Max-Cut

7. Resultados

Se programó el algoritmo cuántico usando el simulador de circuitos cuánticos de Pennylane, que tiene una variedad de simuladores. Por defecto viene integrado el “default.qubit”, sin embargo, se utilizó “lightning.qubit” que es un backend de simulación cuántica para computadores clásicos, lo que lo hace una opción eficiente y optimizada para realizar simulaciones cuánticas rápidas y bajo consumo de recursos sin necesidad de una computadora cuántica real. Esto permitió utilizar una computadora clásica (ver sus especificaciones en anexo 9.1) para hacer las simulaciones. Gracias a esto, se realizaron las siguientes tareas:

1. Se crearon dos versiones para el algoritmo estándar, éstas son:
   - Versión a: Esta versión genera un grafo de manera automática a partir de dos especificaciones: el total de nodos N que va a tener y la cantidad de aristas K conectadas por nodo.
   - Versión b: Esta versión recibe un grafo específico, definido en la versión a.
2. Se creó un código QAOA con inicialización del grafo en estado de Bell, que recibe un grafo generado por la versión a del algoritmo estándar.
3. Se creó un código QAOA con inicialización del grafo en estado de GHZ que recibe un grafo generado por la versión a del algoritmo estándar.

Se estudiaron grafos de orden $N = 6, 10$ y 16 y se pueden observar en el anexo 9.3. Para los órdenes más bajos ($N = 6$ y $N = 10$), se consideraron los grados $K = 2$ (configuración tipo anillo), $K = N/2$ y $K = N - 1$ (cada nodo conectado con todos los otros nodos restantes). Para el orden $N = 16$ se estudiaron únicamente los casos $K = 2$ y $K = N/2$. En el caso $N = 16$, no se pudo estudiar el caso $K = N - 1$, por limitante de recursos computacionales (el tiempo de ejecución tomaba mucho tiempo alrededor de 1 día y 12 horas por cada inicialización). Por obvias razones, al intentar resolver un problema combinatorial, mientras más complejo sea el grafo, mayor recursos va a tomar. Por esta razón se decidió no realizar el grafo de orden $N = 16$ con $K = N - 1$, además aclarar que no todas las configuraciones de grafo cumplen estrictamente con la cantidad K de aristas, esto debido a que el algoritmo que se desarrollo genera grafos de manera automática para agilizar el proceso y para ahorrar tiempo se decidió dejar los grafos que generaba y realizar todas las pruebas con esos grafos.

A su vez, para cada grafo (N, K) se consideran dos escenarios: **A)** grafo con aristas de igual peso 1 $(w_{ij} = 1)$ y **B)** grafo con aristas de distinto peso. En este segundo escenario, la mitad de las aristas tiene

peso 1 y otra mitad peso distinto de 1 (por defecto, se eligió peso igual a 0.3 para dichas aristas). Consecuentemente, se estudiaron 16 grafos, en el anexo 9.3 se pueden ver los 8 grafos ocupados para el caso de aristas con distinto peso (los 8 grafos restantes son iguales, pero con aristas de igual peso).

Los 16 grafos fueron estudiado con cada uno de los tres algoritmos (estándar, Bell y GHZ), considerando profundidades $p = 1, 2$ y 3. EL desempeño de cada algoritmo fue evaluada en términos de 3 cantidades:

- G=*Gap o diferencia % respecto de la solución óptima*. Mide la diferencia relativa con respecto a la solución óptima encontrada vía búsqueda exhaustiva,

$$G = 100 \times \left| \frac{C_{opt} - C_i}{C_{opt}} \right|,$$

- T=*tiempo de ejecución*. Número de iteraciones hasta converger a la solución final. Se considera que el algoritmo ha convergido la solución final, cuando las variaciones el costo son inferiores al 0.5 %.

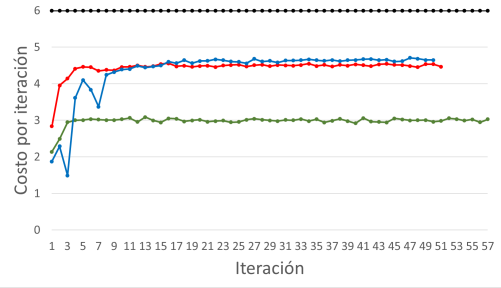
- T_{est} =*tiempo de estabilización*. Número de iteraciones hasta que el costo tenga variaciones inferiores al 5 %. Esto mide cuánto se demora el algoritmo en alcanzar un comportamiento estable, con oscilaciones pequeñas del costo.

Adicionalmente, para garantizar la calidad de los resultados se determinaron una cantidad de "shots", estos representan el número de veces que el circuito cuántico se ejecuta para medir los resultados, debido a que estas mediciones del sistema cuántico son probabilísticas. La configuración de los shots fue de 1000, lo que se traduce que el circuito se repite 1000 veces para estimar con mayor precisión la probabilidad de los estados cuánticos finales.

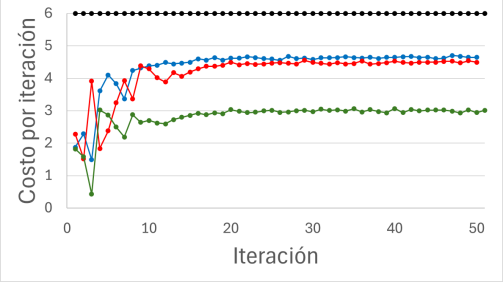
A continuación, se presentan los resultados obtenidos para cada uno de los 16 grafos considerados, en términos de las métricas anteriores (G, T, T_{est}).

7.1. Grafo (N,K)=(6,2)

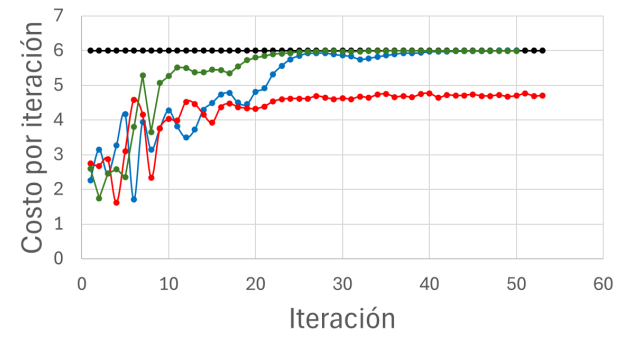
7.1.1. Caso $w = 1$



(a) profundidad 1



(b) profundidad 2



(c) profundidad 3

Figura 14: Gráficos con profundidades 1, 2 y 3 para el grafo = (6,2) con $w = 1$.

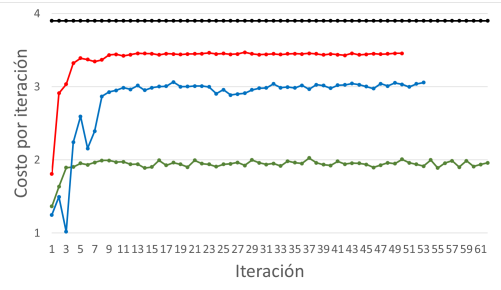
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

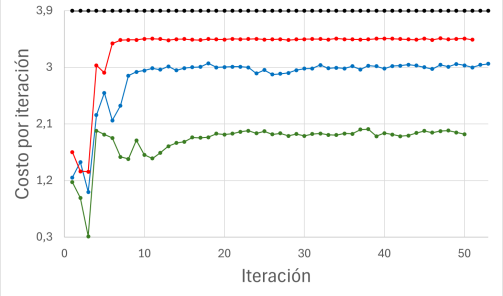
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	8	23,95	52
1	Bell	5	25,55	51
1	GHZ	4	49,53	57
2	Estándar	9	22,5	50
2	Bell	14	25,08	50
2	GHZ	10	49,7	51
3	Estándar	22	0,02	50
3	Bell	17	21,43	52
3	GHZ	10	0,12	50

Tabla 2: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=2, $w_{ij} = 1$.

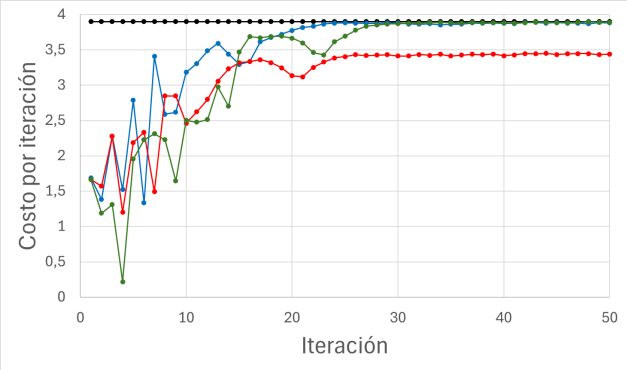
7.1.2. Caso $w \neq 1$



(a) profundidad 1



(b) profundidad 2



(c) profundidad 3

Figura 15: Gráficos con profundidades 1, 2 y 3 el grafo = (6,2) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

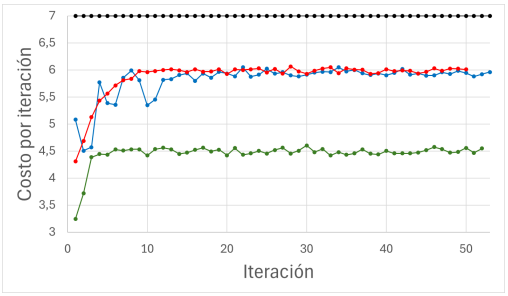
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	9	25,4	52
1	Bell	5	11,39	50
1	GHZ	4	49,77	62
2	Estándar	9	21,61	53
2	Bell	7	11,83	51
2	GHZ	5	50,29	50
3	Estándar	18	0,46	50
3	Bell	15	11,73	50
3	GHZ	16	0,19	50

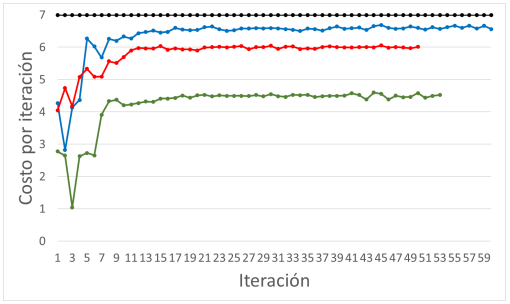
Tabla 3: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=2, $w_{ij} \neq 1$.

7.2. Grafo (N,K)=(6,3)

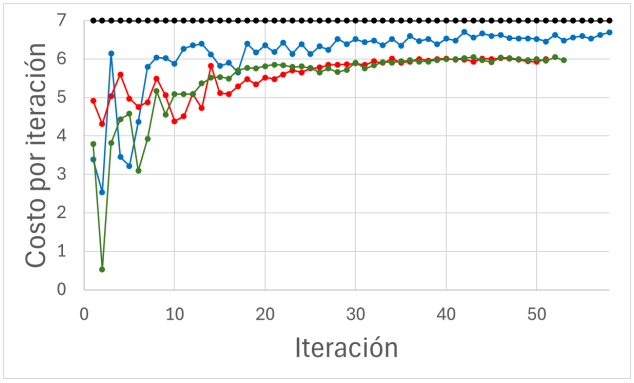
7.2.1. Caso $w = 1$



(a) profundidad 1



(b) profundidad 2



(c) profundidad 3

Figura 16: Gráficos con profundidades 1, 2 y 3 para el grafo = (6,3) con $w = 1$.

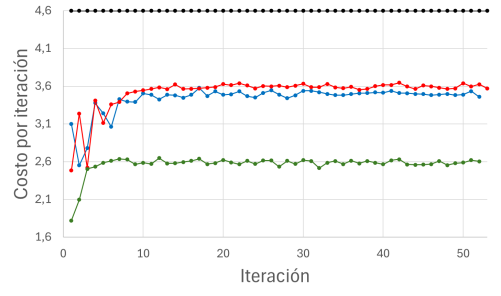
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

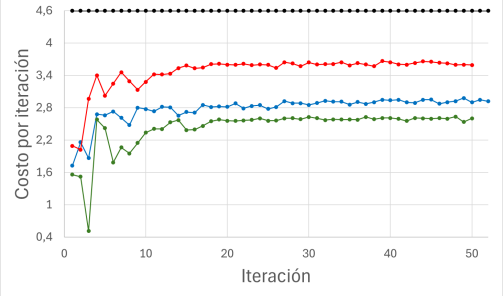
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	13	14,81	53
1	Bell	5	14,12	50
1	GHZ	4	35,03	52
2	Estándar	9	6,26	60
2	Bell	9	14,12	50
2	GHZ	9	35,4	53
3	Estándar	12	4,41	58
3	Bell	16	14,27	51
3	GHZ	11	14,79	53

Tabla 4: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=3, $w_{ij} = 1$.

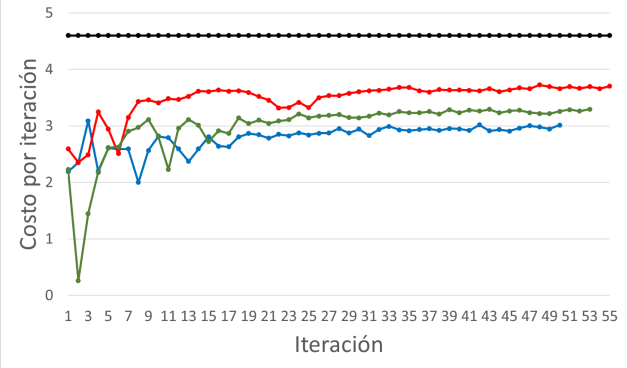
7.2.2. Caso $w \neq 1$



(a) profundidad 1



(b) profundidad 2



(c) profundidad 3

Figura 17: Gráficos con profundidades 1, 2 y 3 para el grafo = (6,3) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

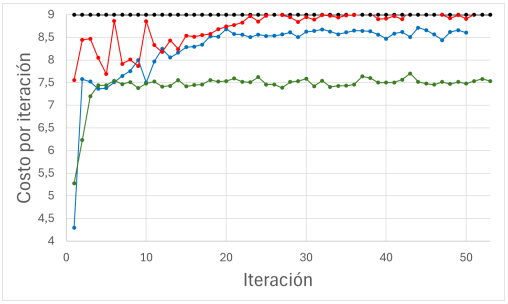
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	8	24,76	52
1	Bell	7	22,37	53
1	GHZ	4	43,37	52
2	Estándar	10	36,45	52
2	Bell	8	21,82	50
2	GHZ	11	43,32	50
3	Estándar	19	34,51	50
3	Bell	9	20,32	55
3	GHZ	19	28,36	53

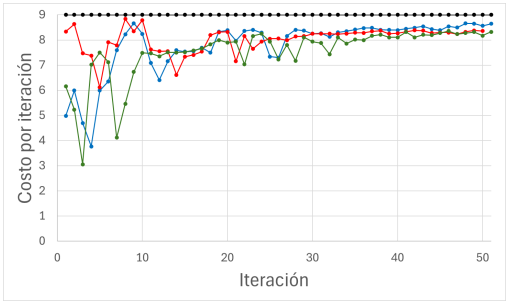
Tabla 5: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=3, $w_{ij} \neq 1$.

7.3. Grafo (N,K)=(6,5)

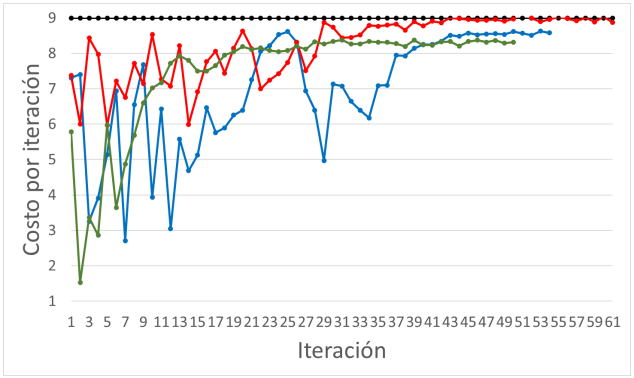
7.3.1. Caso $w = 1$



(a) profundidad 1.



(b) profundidad 2.



(c) Profundidad 3.

Figura 18: Gráficos con profundidades 1, 2 y 3 para el grafo = (6,5) con $w = 1$.

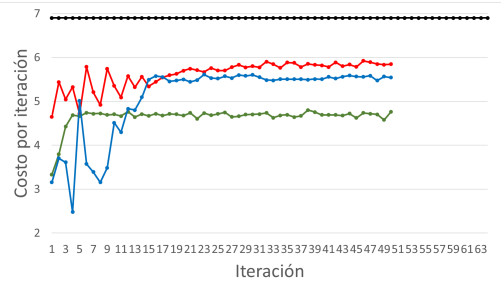
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

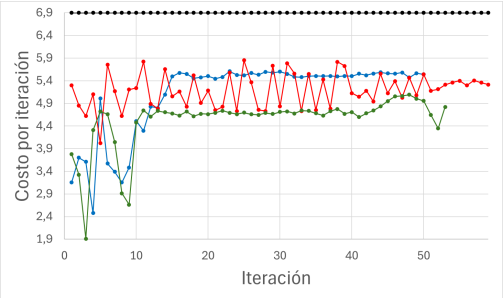
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	12	3,79	50
1	Bell	12	0.03	51
1	GHZ	4	16,27	53
2	Estándar	15	3,92	51
2	Bell	24	7,13	50
2	GHZ	36	7,51	51
3	Estándar	23	4,12	54
3	Bell	30	1,43	61
3	GHZ	19	7,52	50

Tabla 6: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=5, $w_{ij} = 1$.

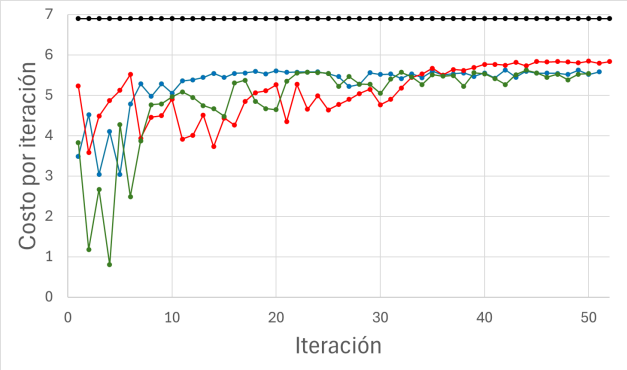
7.3.2. Caso $w \neq 1$



(a) Profundidad 1



(b) Profundidad 2



(c) Profundidad 3

Figura 19: Gráficos con profundidades 1, 2 y 3 para el grafo = (6,5) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

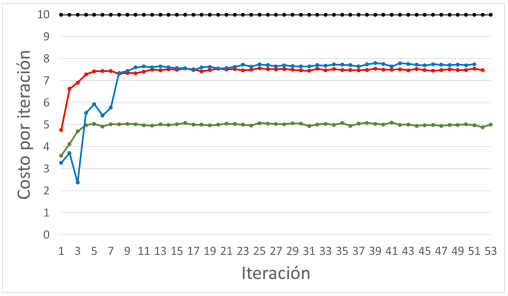
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	11	21,59	64
1	Bell	13	15,28	50
1	GHZ	5	30,96	50
2	Estándar	16	19,69	50
2	Bell	52	22,96	59
2	GHZ	11	30,09	53
3	Estándar	12	19,12	51
3	Bell	26	14,43	52
3	GHZ	9	19,7	50

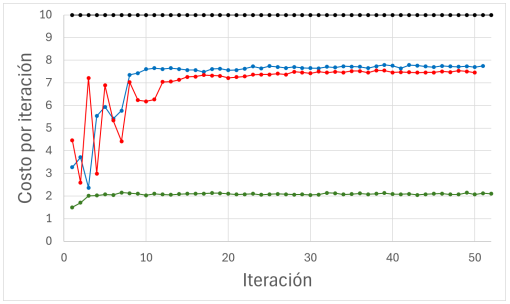
Tabla 7: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=6, K=5, $w_{ij} \neq 1$.

7.4. Grafo (N,K)=(10,2)

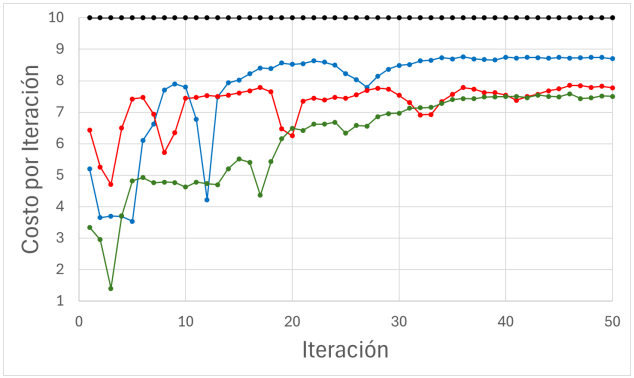
7.4.1. Caso $w = 1$



(a) Profundidad 1



(b) Profundidad 2



(c) profundidad 3

Figura 20: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,2) con $w = 1$.

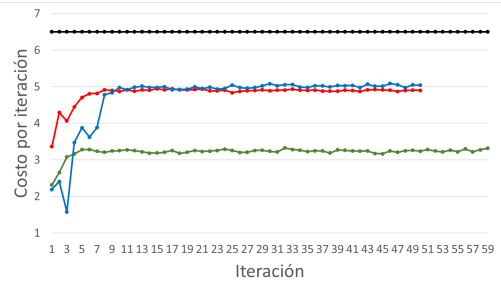
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

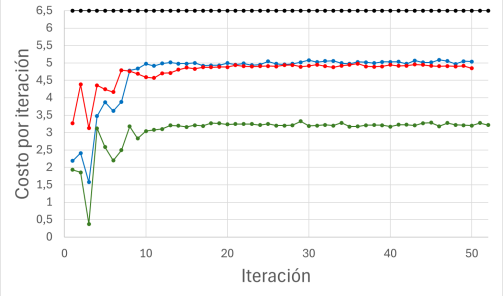
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	7	24,62	53
1	Bell	5	25,19	52
1	GHZ	5	49,91	53
2	Estándar	9	22,5	51
2	Bell	13	25,46	50
2	GHZ	4	79,02	52
3	Estándar	15	12,95	50
3	Bell	22	22,2	50
3	GHZ	21	24,98	50

Tabla 8: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=10, K=2, $w_{ij} = 1$.

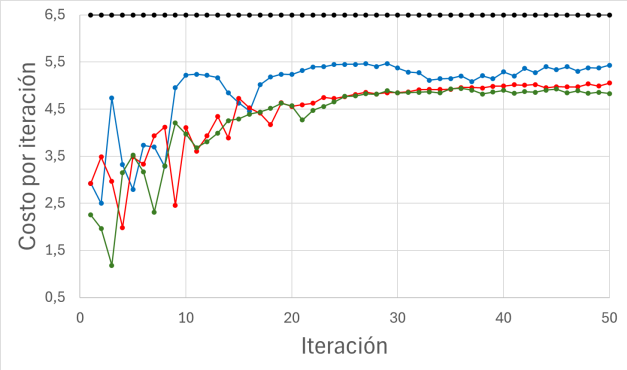
7.4.2. Caso $w \neq 1$



(a) Profundidad 1.



(b) profundidad 2



(c) profundidad 3.

Figura 21: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,2) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

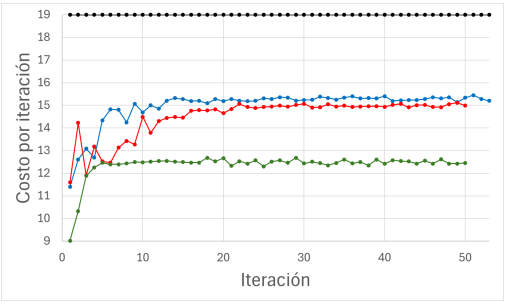
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	7	24,23	51
1	Bell	7	24,6	50
1	GHZ	4	48,95	59
2	Estándar	9	22,46	50
2	Bell	8	25,45	50
2	GHZ	11	50,54	50
3	Estándar	18	16,44	50
3	Bell	20	22,28	50
3	GHZ	12	35,67	50

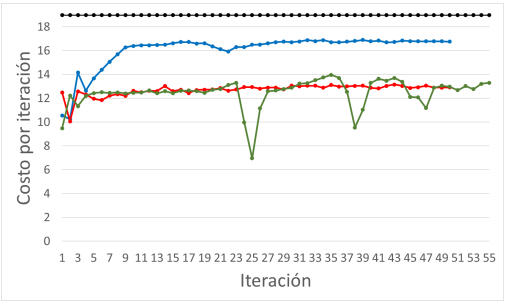
Tabla 9: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=10, K=2, $w_{ij} \neq 1$.

7.5. Grafo (N,K)=(10,5)

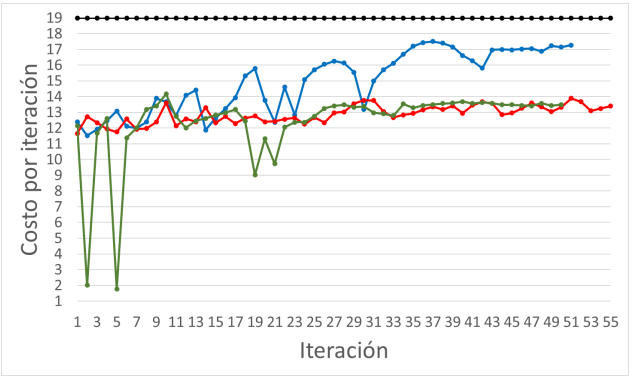
7.5.1. Caso $w = 1$



(a) profundidad 1



(b) Profundidad 2



(c) Profundidad 3

Figura 22: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,5) con $w = 1$.

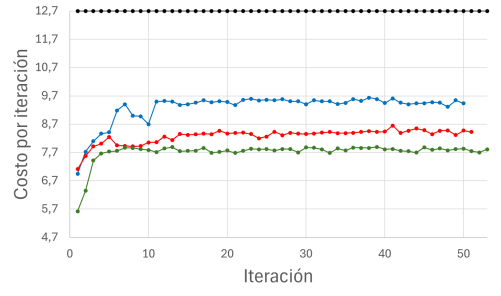
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

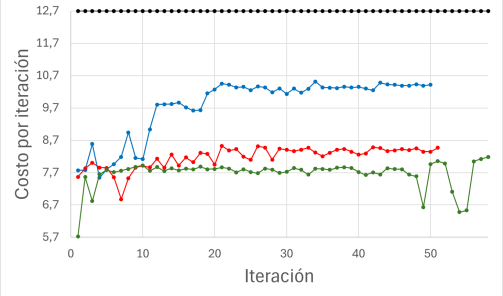
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	7	24,62	53
1	Bell	5	25,19	52
1	GHZ	4	34,43	50
2	Estándar	9	22,5	51
2	Bell	13	25,46	50
2	GHZ	41	29,99	55
3	Estándar	15	12,95	50
3	Bell	21	22,2	50
3	GHZ	23	28,99	50

Tabla 10: Métricas de performance de cada algoritmo para distintas profundidades. Caso N=10, K=5, $w_{ij} = 1$.

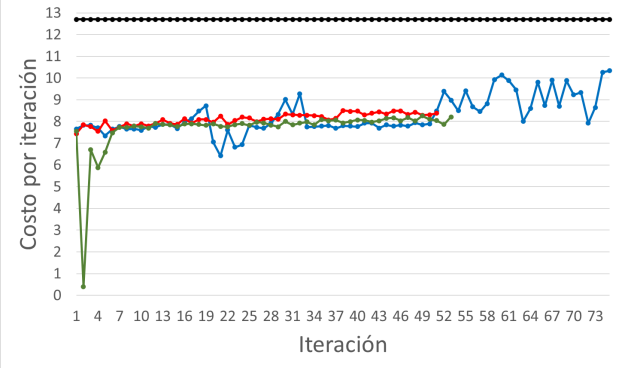
7.5.2. Caso $w \neq 1$



(a) Profundidad 1



(b) Profundidad 2



(c) Profundidad 3

Figura 23: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,5) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

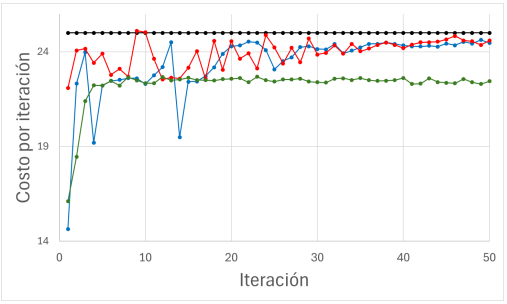
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	7	24,23	51
1	Bell	5	24,6	50
1	GHZ	4	38,47	53
2	Estándar	9	22,46	50
2	Bell	13	25,45	50
2	GHZ	5	35,58	58
3	Estándar	18	16,45	50
3	Bell	21	22,28	50
3	GHZ	7	35,35	53

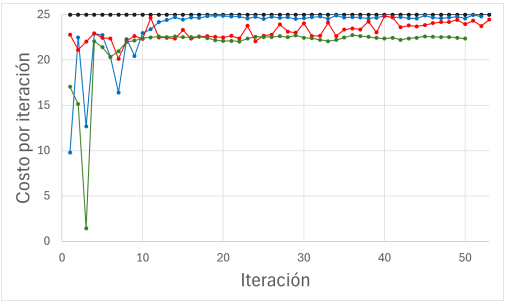
Tabla 11: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=10, K=5, $w_{ij} \neq 1$.

7.6. Grafo (N,K)=(10,9)

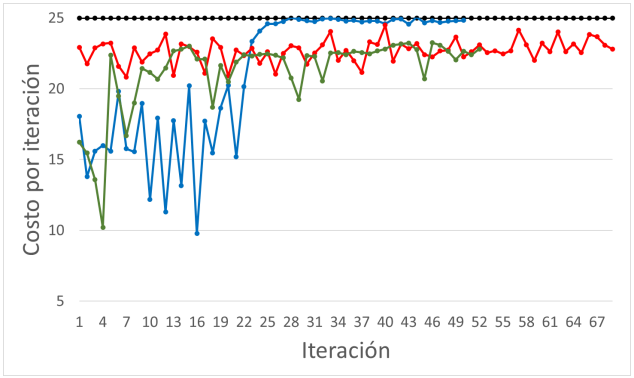
7.6.1. Caso $w = 1$



(a) profundidad 1.



(b) profundidad 2.



(c) profundidad 3.

Figura 24: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,9) con $w = 1$.

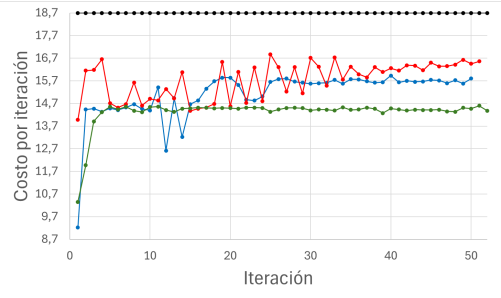
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

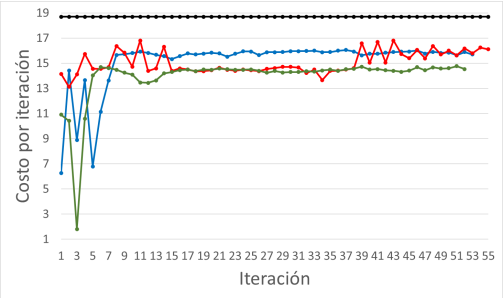
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	16	2,09	50
1	Bell	25	1,62	50
1	GHZ	4	10,2	50
2	Estándar	11	0,82	52
2	Bell	43	2,23	53
2	GHZ	5	10,57	50
3	Estándar	23	0,68	50
3	Bell	42	8,84	69
3	GHZ	10	8,81	52

Tabla 12: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=10, K=9, $w_{ij} = 1$.

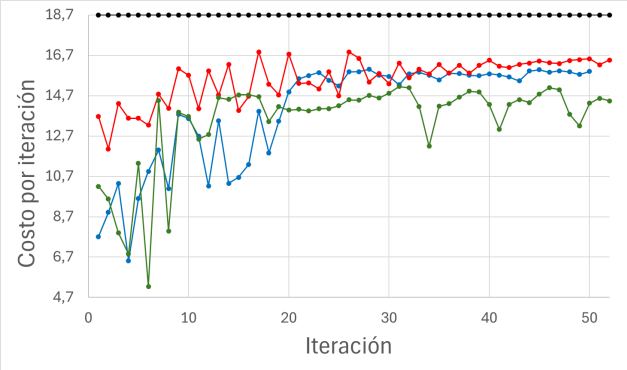
7.6.2. Caso $w \neq 1$



(a) Profundidad 1



(b) Profundidad 2.



(c) profundidad 3.

Figura 25: Gráficos con profundidades 1, 2 y 3 para el grafo = (10,9) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

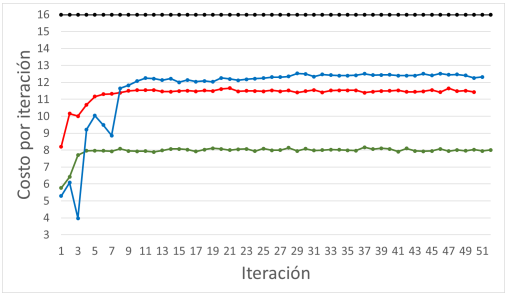
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	16	15,48	50
1	Bell	35	11,43	51
1	GHZ	4	23,1	52
2	Estándar	9	16,08	53
2	Bell	45	13,86	55
2	GHZ	6	22,27	52
3	Estándar	21	14,97	50
3	Bell	32	11,96	52
3	GHZ	14	22,77	52

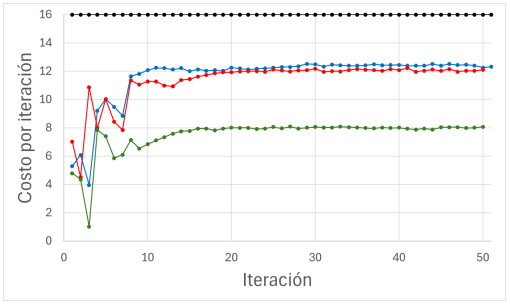
Tabla 13: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=10, K=9, $w_{ij} \neq 1$.

7.7. Grafo (N,K)=(16,2)

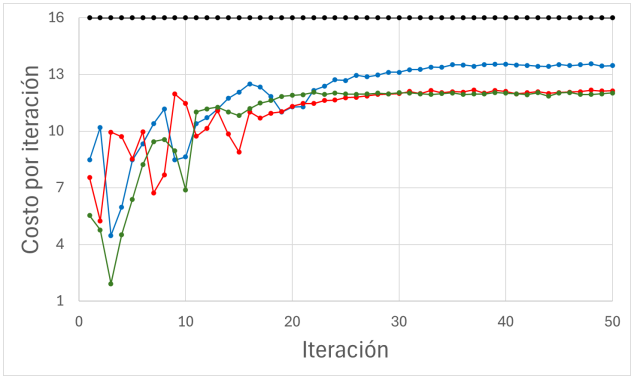
7.7.1. Caso $w = 1$



(a) Profundidad 1.



(b) Profundidad 2.



(c) Profundiad 3.

Figura 26: Gráficos con profundidades 1, 2 y 3 para el grafo = (16,2) con $w = 1$.

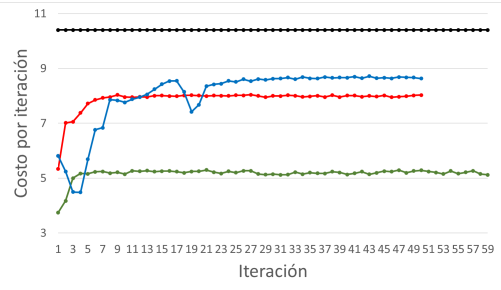
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

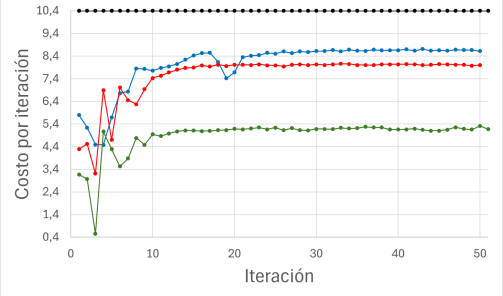
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	10	24,76	50
1	Bell	5	28,58	50
1	GHZ	4	49,93	52
2	Estándar	9	22,98	51
2	Bell	9	24,32	50
2	GHZ	10	49,48	50
3	Estándar	20	15,8	50
3	Bell	17	24,13	50
3	GHZ	12	24,87	50

Tabla 14: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=16, K=2, $w_{ij} = 1$.

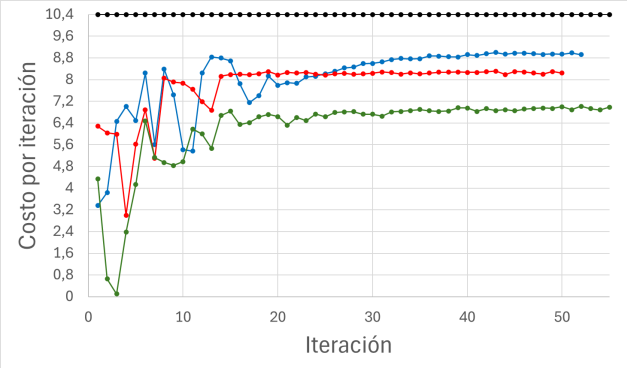
7.7.2. Caso $w \neq 1$



(a) profunidad 1.



(b) Profundidad 2.



(c) profundidad 3.

Figura 27: Gráficos con profundidades 1, 2 y 3 para el grafo = (16,2) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

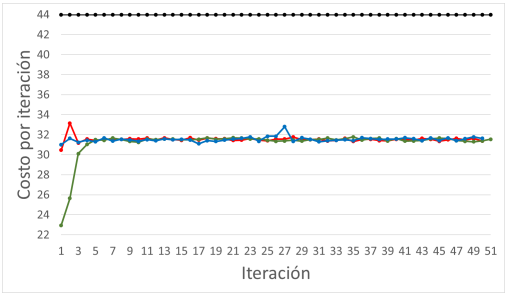
En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	9	25,04	50
1	Bell	2	22,79	50
1	GHZ	4	50,77	59
2	Estándar	9	17,03	50
2	Bell	11	23,03	50
2	GHZ	11	50,25	50
3	Estándar	20	14,21	52
3	Bell	15	20,74	50
3	GHZ	17	31,83	55

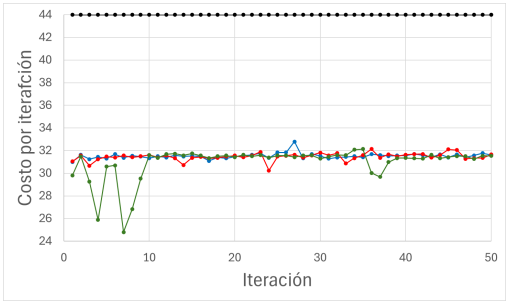
Tabla 15: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=16, K=2, $w_{ij} \neq 1$.

7.8. Grafo (N,K)=(16,8)

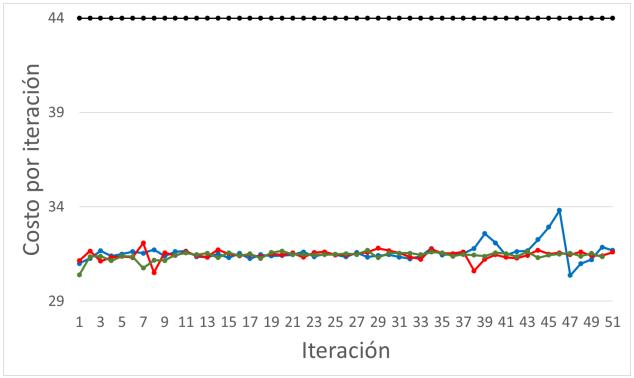
7.8.1. Caso $w = 1$



(a) Profundidad 1.



(b) Profundidad 2.



(c) Profundidad 3.

Figura 28: Gráficos con profundidades 1, 2 y 3 para el grafo = (16,8) con $w = 1$.

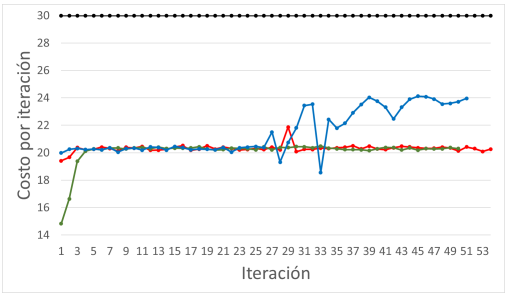
Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

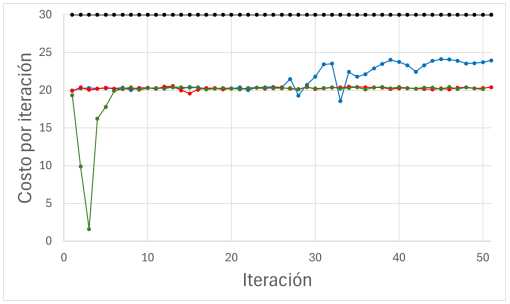
p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	11	28,59	50
1	Bell	4	28,68	50
1	GHZ	4	28,29	51
2	Estándar	2	28,13	50
2	Bell	4	28,05	50
2	GHZ	11	28,31	50
3	Estándar	2	27,97	51
3	Bell	2	28,19	51
3	GHZ	2	28,75	50

Tabla 16: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=16, K=82, $w_{ij} = 1$.

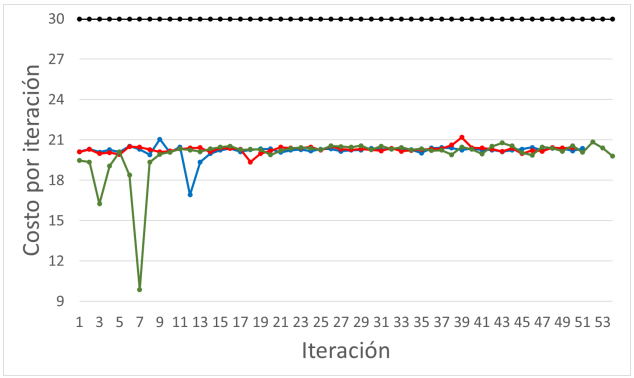
7.8.2. Caso $w \neq 1$



(a) Profundidad 1



(b) profundidad 2



(c) Profundidad 3

Figura 29: Gráficos con profundidades 1, 2 y 3 para el grafo = (16,8) con $w \neq 1$.

Curva roja: algoritmo inicializado en estado de Bell, Curva azul: Algoritmo QAOA estándar, Curva verde: algoritmo inicializado en estado de Ghz.

En términos de las métricas de desempeño, el comportamiento de cada algoritmo se resume en la siguiente tabla:

p	Tipo Algoritmo	T_{est}	G	T
1	Estándar	2	31,14	51
1	Bell	31	32,45	54
1	GHZ	4	32,33	50
2	Estándar	35	20,11	51
2	Bell	4	31,97	51
2	GHZ	7	32,85	50
3	Estándar	14	32,12	51
3	Bell	19	31,95	50
3	GHZ	9	34,04	54

Tabla 17: Métricas de desempeño de cada algoritmo para distintas profundidades. Caso N=16, K=8, $w_{ij} \neq 1$.

7.9. Análisis de Resultados

En ciertos escenarios, el algoritmo inicializado con estados de Bell es superior al algoritmo estándar, en el sentido que reduce el gap entre la solución encontrada y la solución óptima. Para algoritmos de profundidad $p = 1$, los casos en que el algoritmo con estados de Bell tuvo mejor desempeño que el algoritmo estándar fueron los siguientes:

Grafo	Peso	G_{Bell}	G_{est}
$6, 2$	$w \neq 1$	11,39	25,40
$6, 3$	$w \neq 1$	22,37	24,76
$6, 5$	$w = 1$	0,03	3,79
$6, 5$	$w \neq 1$	15,28	21,59
$10, 9$	$w = 1$	11,43	15,28
$10, 9$	$w \neq 1$	11,43	15,28
$16, 2$	$w \neq 1$	22,79	25,04

Tabla 18: Comparación de métricas de desempeño en grafos en donde el algoritmo modificado con estados de Bell es superior al estándar. Se consideran únicamente circuitos de profundidad 1.

En un 44 % que equivale a 7 de los grafos, hubo una mejora en el gap con respecto a la solución óptima, es decir, que el estado de Bell estuvo más cerca de la solución óptima que el estado estándar. Mejorando en promedio el gap en un 38 % .

Al aumentar la profundidad ($p = 2$), los grafos en que el algoritmo con estados de Bell tuvo mejor desempeñó que el algoritmo estándar fueron los siguientes:

Grafo	Peso	G_{Bell}	G_{est}
$6, 2$	$w \neq 1$	11,83	21,69
$6, 3$	$w \neq 1$	21,82	36,45
$6, 5$	$w \neq 1$	15,28	21,59
$10, 9$	$w \neq 1$	13,86	16,08

Tabla 19: Comparación de métricas de desempeño en grafos en donde el algoritmo modificado con estados de Bell es superior al estándar. Se consideran únicamente circuitos de profundidad 2.

Para $p = 2$ hubo 4 grafos (25 % de los casos) en donde el algoritmo de Bell tuvo mejor desempeño que el algoritmo estándar. Para $p = 3$ ocurre algo similar.

Grafo	Peso	G_{Bell}	G_{est}
$6, 3$	$w \neq 1$	20,32	34,51
$6, 5$	$w = 1$	1,43	4,12
$6, 5$	$w \neq 1$	14,43	19,12
$10, 9$	$w \neq 1$	11,96	14,97

Tabla 20: Comparación de métricas de desempeño en grafos en donde el algoritmo modificado con estados de Bell es superior al estándar. Se consideran únicamente circuitos de profundidad 3.

En un 44 % (para $p = 1$) y 25 % (para $p = 2$ y 3) de los casos se observa una reducción de la diferencia

entre la solución encontrada por el algoritmo y la solución óptima. En algunos casos, la reducción es notoria. Por ejemplo, para el grafo (6,5) con pesos iguales el gap se reduce del 4 % al 0.03 %. En general, el beneficio se concentra en grafos con aristas de distintos pesos, aunque no está unicamente restringido a esta situación. En la mayoría de estos grafos los tiempos de estabilización y de convergencia a la solución final son similares entre ambos algoritmos. Sin embargo, hay que destacar que, cuando aumenta la profundidad, el algoritmo de Bell toma mayor tiempo en estabilizarse (el tiempo de estabilización es el doble con respecto al estándar).

En el caso de la inicialización del GHZ, éste presenta resultados no sobresalientes en profundidades $p = 1$ y $p = 2$, es decir, que no presenta ninguna ventaja en comparación con el algoritmo estándar. Sin embargo, este presentó un comportamiento interesante cuando la profundidad es igual a 3, encontrando en ciertos casos la solución óptima, o bien, presentando resultados similares al de Bell o al de la inicialización estándar.

Para $p = 3$, se llego a la solución óptima en los siguientes grafos (configuración tipo anillo):

Grafo	Peso	G_{GHZ}	G_{est}
$6, 2$	$w = 1$	0,12	0,02
$6, 2$	$w \neq 1$	0,19	0,46

Tabla 21: Grafos en que el algoritmo inicializado en estados GHZ alcanzó la solución óptima ($p = 3$).

Estos son los dos casos (13 %) en donde la inicialización de GHZ logra llegar al óptimo, siendo además mejor que el algoritmo con estados Bell. En general, se observa que es un tipo de inicialización que no exhibe grandes beneficios. Por otro lado, el comportamiento en profundidades igual a 3 resultó bastante curioso. El desempeño del algoritmo mejoró de manera sustancial (en 9 grafos) al incrementar la profundidad del circuito, como se exhibe en la tabla 22.

Grafo	Peso	$p = 1$	$p = 2$	$p = 3$
$6, 2$	$w = 1$	49,53	49,70	0,12
$6, 2$	$w \neq 1$	49,77	50,29	0,19
$6, 3$	$w = 1$	35,03	35,40	14,79
$6, 3$	$w \neq 1$	43,37	43,32	28,36
$6, 5$	$w \neq 1$	30,96	30,09	19,70
$10, 2$	$w = 1$	49,91	79,02	24,98
$10, 2$	$w \neq 1$	48,95	50,54	35,67
$16, 2$	$w = 1$	49,93	49,48	24,87
$16, 2$	$w \neq 1$	50,77	50,25	31,83

Tabla 22: Grafos en que el algoritmo inicializado en estados GHZ mejoró su desempeño al incrementar la profunidad del circuito.

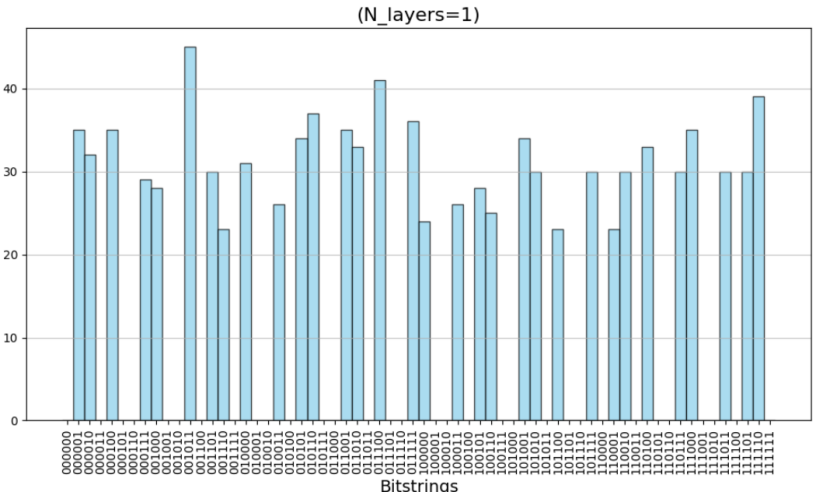
En total, son 9 casos en donde GHZ presenta una mejora considerable en la profundidad igual a 3 con respecto a las demás. Se observa indiferencia si las aristas tienen igual peso o distinto peso. Sin embargo,

resaltan los grafos simples cuyos nodos contienen pocas aristas.

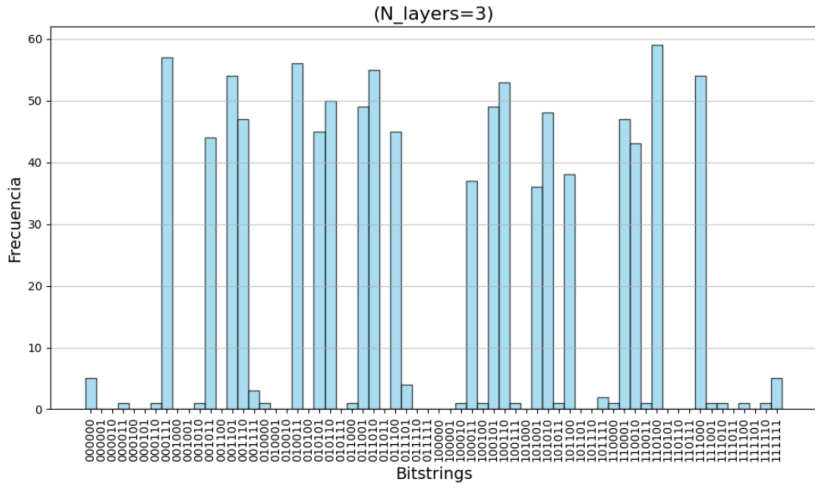
Por lo tanto, al comparar el algoritmo inicializado en estado GHZ consigo mismo para distintas profundidades, se observa que

- El promedio de la reducción del gap de la profundidad 3 con respecto a la profundidad 1 es de un 55 %, y
- El promedio de la reducción del gap de la profundidad 3 con respecto a la profundidad 2 es de un 57 %.

Estos comportamientos de la reducción del gap en ciertos casos se puede ver también en los histogramas que se generaron en el código, donde el eje horizontal son los bistrings (todas las posibles soluciones representados en 0 y 1) y el eje vertical representa la frecuencia con la cual se midió cada bitstring, por lo que el histograma representa todas aquellas soluciones y cuantas veces aparecieron estos resultados al medir el circuito.



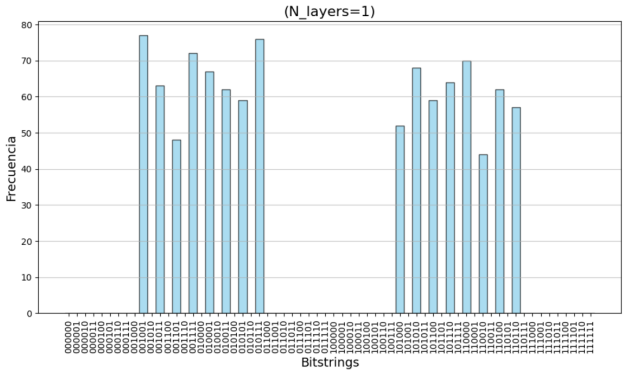
(a) Profundidad 1.



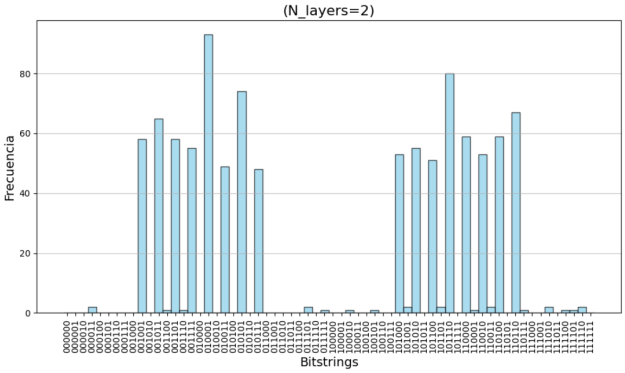
(b) Profundidad 3.

Figura 30: Histogramas de la inicialización GHZ.

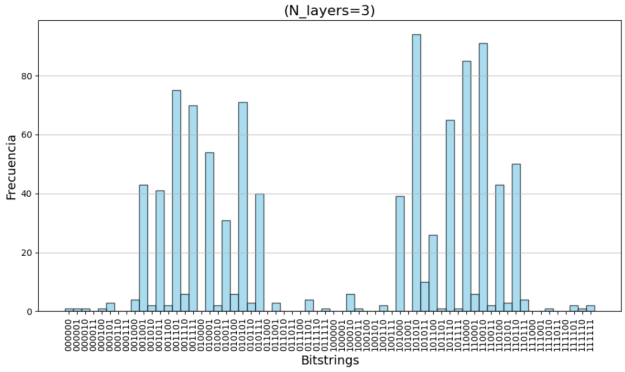
En los mostrados en las figuras 30, cuyo grafo es el (6,2) $w \neq 1$, se observa un comportamiento de bitstring en pares, tanto en profundidad 1 como en profundidad 3. En el histograma 30a no se observa claramente cuál seria el bitstring con la solución óptima. Por otro lado, en el histograma 30b, si bien tampoco se observa claridad en la solución, este tiene menos elementos en pares. Este comportamiento se repite constantemente en la mayoría de los grafos, independientemente si los pesos de las aristas son distintos o son todos iguales. Esto podría explicar porqué la inicialización de GHZ mejora considerablemente en una profundidad igual a 3.



(a) Profundidad 1.



(b) Profundidad 2.



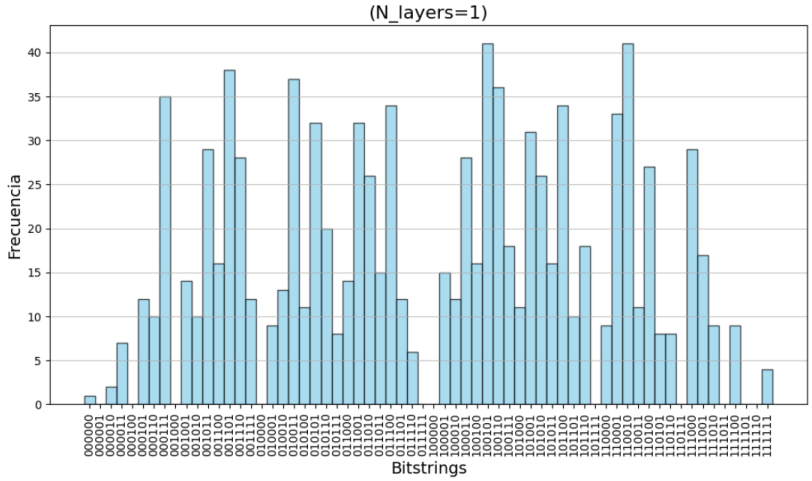
(c) Profundidad 3.

Figura 31: Histogramas de la inicialización Bell.

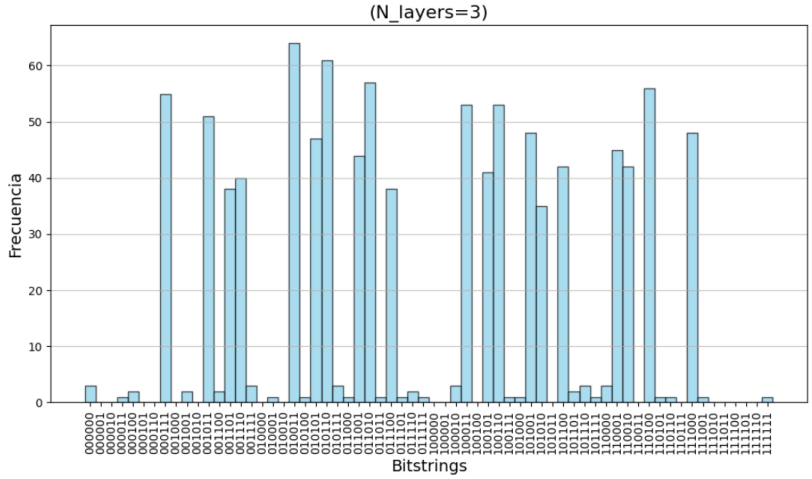
Para la inicialización en estado de Bell también sucede algo curioso y es que al igual que el GHZ este tiene un comportamiento de bitstring en pares, pero en menor medida. Posiblemente, esto puede explicarse debido a que es un comportamiento que refleja el entrelazamiento, sin embargo, para un próximo análisis se investigará en mayor profundidad.

Otra observación importante se resume en la figura 31, que muestra que, para el algoritmo con estados iniciales de Bell, el ruido comienza aumentar a medida que se incrementa la profundidad. Esto sucede tanto para aristas de distinto peso como para aristas de igual peso. Eso explica porqué el estado de Bell es mejor

en profundidades bajas (o, al menos, donde consiguió mejores resultados).



(a) Profundidad 1.



(b) Profundidad 2.

Figura 32: Histogramas de la inicialización Estándar.

Para el caso de la inicialización estándar se observa un comportamiento normal en la figura 32, dejando en claro que a medida que se aumenta la profundidad se genera una mejora. Esto es algo común en la mayoría de los grafos, cuando se utiliza el algoritmo estándar.

8. Conclusiones

En este trabajo se utilizó el algoritmo híbrido cuántico-clásico QAOA. Este cuenta con dos partes, una parte cuántica que se compone de una secuencia de compuertas cuánticas parametrizadas que producen un estado cuántico a partir de un estado inicial. El estado que se genera depende del valor de los parámetros. La otra parte es un proceso de optimización clásico para encontrar el valor de los parámetros que maximiza una cierta función objetivo (la energía de un Hamiltoniano). Las soluciones se buscan en una parte del espacio de Hilbert (definida por el valor que van tomando los parámetros), de forma que se pueden explotar las ventajas cuánticas. La optimización (actualización de los parámetros) se realiza en forma clásica, por ejemplo, siguiendo la dirección del gradiente, Newton-Raphson, etc. Si esta parte fuera cuántica, probablemente se necesitarían muchos más qubits. El algoritmo combina, por lo tanto, aspectos clásicos y cuánticos, que lo hacen atractivo en la era tecnológica actual en que los computadores tienen un número intermedio de qubits (NISQ). Posiblemente, esto explica que estén siendo actualmente ocupados en distintas disciplinas como la química cuántica, comunicaciones, finanzas, logística, etc.

El objetivo principal de esta tesis fue explorar y evaluar cómo el algoritmo variacional Quantum Approximate Optimization Algorithm (QAOA) puede resolver el problema Max-Cut utilizando diferentes configuraciones iniciales: estados estándar, estados de Bell y estados GHZ. Por lo tanto, se buscó responder si el uso de estados entrelazados en la inicialización podría mejorar la eficiencia y precisión del algoritmo en grafos con pesos iguales o desiguales en las aristas.

En términos generales, el código estándar tiene mejor desempeño en más del 56 % de los casos, sin importar si las aristas tienen distinto o igual peso. El estado de Bell exhibe mejor desempeño en muchos casos. Específicamente, *en el subgrupo de grafos con aristas de distinto peso resueltos a través de un circuito de profundidad 1*, el algoritmo con estados de Bell tuvo mejor desempeño que el algoritmo estándar. En cambio, al usar el estado GHZ, el rendimiento fue deficiente en la gran mayoría de casos. Solo en un 13 % de los grafos, el estado GHZ mostró ventajas sobre los estados de Bell y estándar, en términos de las métricas de desempeño utilizadas.

En consecuencia, en un caso general, el algoritmo con inicialización en estado estándar sigue siendo mejor opción al inicializiar el algoritmo QAOA para resolver el problema Max-Cut. Sin embargo, el algoritmo con inicialización en estado de Bell muestra mejores resultados cuando el circuito tiene profundidad 1, en ciertos grafos específicos (7 grafos de un total de 16).

Para responder el objetivo especifico N°1, se sabe que la profundidad del circuito, en términos generales, a medida que aumenta, incrementa el costo de recursos a nivel computacional, pero a su vez mejora la precisión de los resultados. Con una inicialización en estado estándar, los resultados esperados mejoran a medida que se aumenta la profundidad. En cambio con una inicialización en estado de Bell, la precisión de los resultados, en general, disminuye a medida que aumenta la profundidad. Por último, con una inicialización en estado GHZ, al aumentar la profundidad, la precisión de los resultados mejora de forma importante.

En relación al objetivo especifico N°2, se puede señalar que en los grafos, el estado estándar presenta buenos resultados y no existe diferencia en si el grafo se encuentra balanceado o no, mientras que el estado GHZ, independientemente si estos se encuentran balanceados o no, fue el más deficiente en los resultados. Por último, el estado de Bell presenta buenos resultados en grafos no balanceados (en su mayoría). Para los tres tipos de inicialización, la gran parte tiende a converger de manera similar.

Finalmente en respuesta al objetivo especifico N°3, comenzar el circuito en un estado entrelazado generó beneficios, específicamente en estado de Bell, obtuvo beneficios en grafos no balanceados y en profundidades bajas. El estado GHZ exhibió resultados deficientes en general, sin embargo, al aumentar la profundidad a 3, se observó una mejora, independientemente si el grafo es balanceado o no, lo que podría plantearse si en profundidades más altas puede llegar a ser una opción a considerar.

Trabajo a futuro: Para los estados de Bell, se buscará continuar la investigación en grafos con aristas de distintos pesos para grafos de mayor orden (N = 50, 100, 1000) y estudiar el comportamiento del algoritmo en estas situaciones. Principalmente con foco en el caso $p = 1$, que es donde se concentran mejores resultados.

Para los estados GHZ, se requiere continuar la investigación para mayores profundidades, en particular, en grafos con muchos nodos y pocas aristas (en comparación al número de nodos), y estudiar el comportamiento del algoritmo para profundidades mayores, analizando si persiste la mejora observada.

Referencias

[1] Farhi, E., Goldstone, J., y Gutmann, S. (2014). A quantum approximate optimization algorithm. En arXiv [quant-ph]. http://arxiv.org/abs/1411.4028

[2] Preskill, J. (2018). "Quantum Computing in the NISQ era and beyond."Quantum, 2, 79.

[3] Hadfield, S., Wang, Z., O'Gorman, B., Rieffel, E. G., Venturelli, D., y Biswas, R. (2017). From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. En arXiv [quant-ph]. http://arxiv.org/abs/1709.03489

[4] Lucas, A. (2013). Ising formulations of many NP problems. En arXiv [cond-mat.stat-mech]. http://arxiv.org/abs/1302.5843

[5] Goemans, M. X., & Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. https://www.iitgoa.ac.in/~sreejithav/misc/maxcut.pdf

[6] Arufe Rivas, L. (2021). Optimización del algoritmo QAOA para el problema Max-Cut en un computador cuántico [Tesis de maestría, Universidad de Oviedo]. DigiBUO. https://digibuo.uniovi.es/dspace/bitstream/handle/10651/56223/TFM_LisArufeRivas.pdf?sequence=6

[7] Zhou, L., et al. (2020). "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices."Physical Review X, 10(2), 021067.

[8] Academia EITCA. (2024). En el contexto de QAOA, ¿cómo contribuyen el Hamiltoniano de costo y el Hamiltoniano mixto a explorar el espacio de soluciones y cuáles son sus formas típicas para el problema de Max-Cut? *EITCA Academy*. https://encr.pw/tLlfr

[9] Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., Mok, W., Sim, S., Koh, D. E., & Aspuru-Guzik, A. (2022). Noisy intermediate-scale quantum (NISQ) algorithms. *Reviews of Modern Physics, 94*(1), 015004. https://arxiv.org/pdf/2101.08448

[10] Max-Cut y Problema del Vendedor Viajero - Qiskit Optimization 0.6.0. (s. f). Github.io. Recuperado el 11 de diciembre de 2024, de https://qiskit-community.github.io/qiskit-optimization/locale/es_UN/tutorials/06_examples_max_cut_and_tsp.html

[11] Kostas, B., Dean, B., Andrea, C., Chiao-Hui, C., Rui-Hao, L., Komal, P., y Alessandro, S. (2023). A review on Quantum Approximate Optimization Algorithm and its variants. En arXiv [quant-ph]. http://arxiv.org/abs/2306.09198

[12] Nielsen, M. A., y Chuang, I. L. (2010). Quantum Computation and Quantum Information (10th Anniversary Edition). Cambridge University Press.

[13] Greenberger, D. M., Horne, M. A., y Zeilinger, A. (2007). Going beyond Bell's Theorem. En arXiv [quant-ph]. http://arxiv.org/abs/0712.0921

[14] Zeilinger, A. (2002). Bell's theorem, information and quantum physics. 241–254. https://doi.org/10.1007/978-3-662-05032-3_17

[15] Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika, 1*(3), 195–200.https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195

[16] Introduction to Qiskit. (s. f). IBM Quantum Documentation. Recuperado el 18 de diciembre de 2024, de https://docs.quantum.ibm.com/guides

[17] IBM roadmap to quantum-centric supercomputers (Updated 2024). (s. f). Ibm.com. Recuperado el 18 de diciembre de 2024, de https://www.ibm.com/quantum/blog/ibm-quantum-roadmap-2025

[18] Xanadu. (s/f). Xanadu.ai. Recuperado el 15 de enero de 2025, de https://www.xanadu.ai

[19] Quantum programming software — PennyLane. (s/f). Pennylane.Ai. Recuperado el 15 de enero de 2025, de https://pennylane.ai

[20] Neven, H. (2024). Meet Willow, our state-of-the-art quantum chip. Google. https://blog.google/technology/research/google-willow-quantum-chip/

[21] Quantum computing. (s. f). Dwavesys.com. Recuperado el 18 de diciembre de 2024, de https://www.dwavesys.com/learn/quantum-computing/

[22] Mermin, N. D. (2007). Quantum computer science: An introduction. Cambridge University Press.

[23] Perez,(2025).Algoritmos-Cuanticos.https://github.com/NICOSPREAM/Algoritmos-Cuanticos

9. Anexos

9.1. Especificaciones del computador

- Procesador Ryzen 5600
- Memoria RAM de 16gb con 2600 MHZ
- Almacenamiento M.2 de 1TB kingston
- Gráfica GTX 1660 Super de 6 GB

9.2. Planificación

La planificación seguida durante el segundo semestre del 2024, ajustada a la entrega actual, abarca desde agosto del 2024 hasta enero del 2025. Además, durante todo el proceso de la tesis se realizaron reuniones semanales (lunes, 10:30–13:00 hrs) para revisar avances y resultados. A continuación, se describen los resultados parciales obtenidos durante el trabajo de tesis (hitos), así como la distribución del trabajo.

1. Formulación del problema: Se formuló matemáticamente el problema Max-Cut para grafos balanceados y no balanceados. Luego, el problema fue formulado en términos de un Hamiltoniano y se estudió al ansatz del algoritmo QAOA. Este proceso, de tipo teórico, fue realizado bajo la supervisión del profesor.

2. Implementación: Se implementó el algoritmo QAOA en PennyLane, con los distintos tipos de inicialización (estado estándar, Bell y GHZ). Las validaciones fueron realizadas por el alumno usando grafos de pocos nodos y aristas.

3. Análisis de resultados: De manera conjunta se analizaron los resultados obtenidos en cada uno de los grados y circuitos empleados, así como el planteamiento de las conclusiones finales.

4. Redacción y revisión: Redacción del documento de tesis por parte del alumno, bajo direcciones generales del docente.

A continuación, se presenta la carta Gantt seguida durante el semestre. Para una visualización más clara, ingresar al siguiente link
(`https://view.monday.com/8284599769-a4ba00e2ca2f92c5fdcec677b5529849?r=u`)

Figura 33: Carta Gantt seguida durante el semestre 2024.

9.3. Grafos ocupados

Aquí se presentan los grafos con distinto pesos, debido a que los 8 restantes son los mismos pero con peso igual a 1.

1. Grafo (6,2)

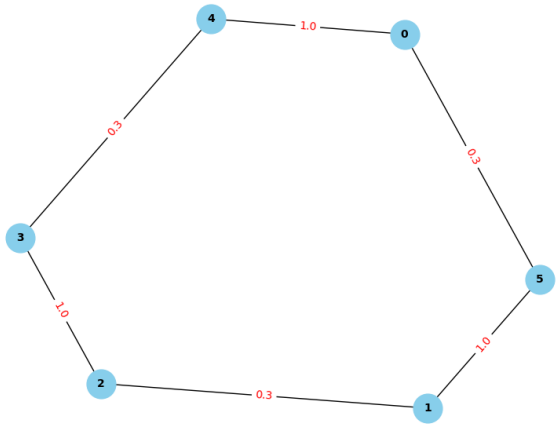


Figura 34: Grafo de orden N=6 y grado K=2.

2. Grafo (6,3)

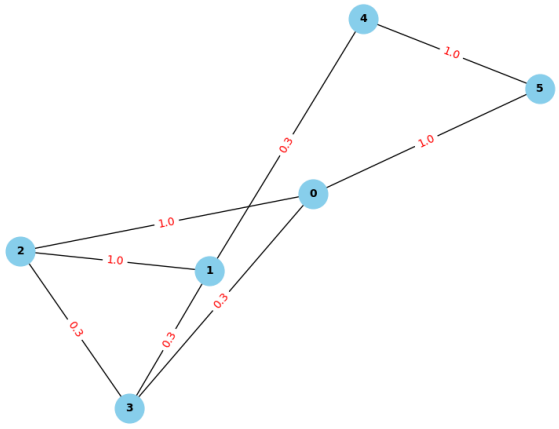


Figura 35: Grafo de orden N=6 y grado K=3.

3. Grafo (6,5)

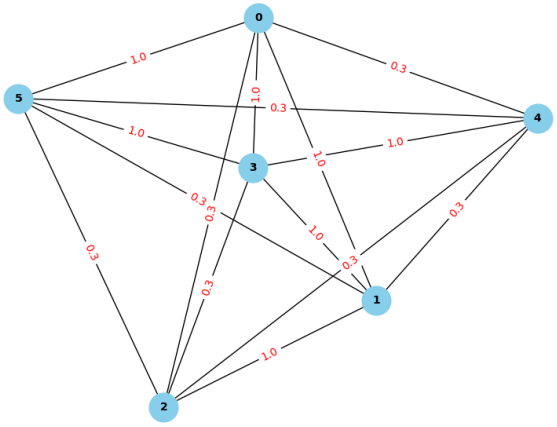


Figura 36: Grafo de orden N=6 y grado K=5.

4. Grafo (10,2)

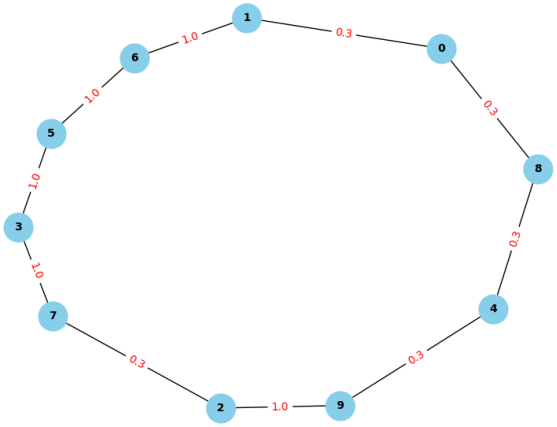


Figura 37: Grafo de orden N=10 y grado K=2.

5. Grafo (10,5)

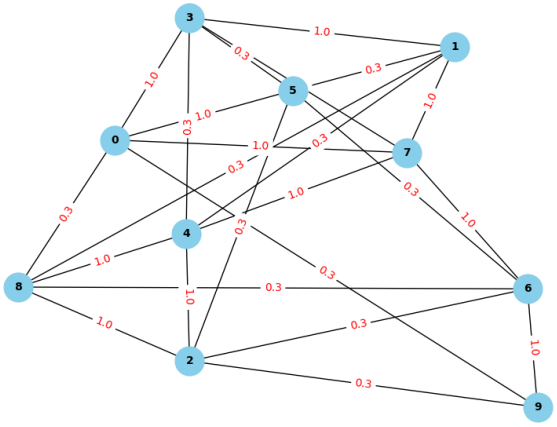


Figura 38: Grafo de orden N=10 y grado K=5.

6. Grafo (10,9)

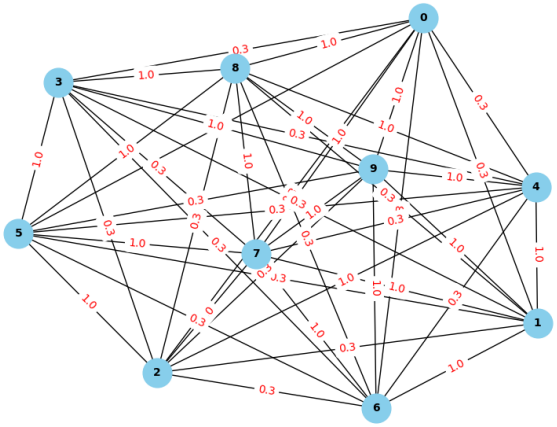


Figura 39: Grafo de orden N=10 y grado K=9.

7. Grafo (16,2)

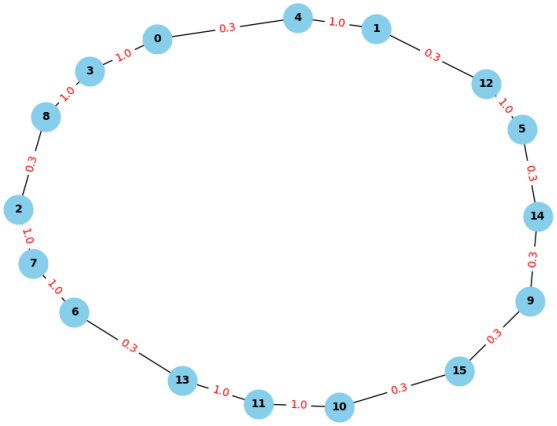


Figura 40: Grafo de orden N=16 y grado K=2.

8. Grafo (16,8)

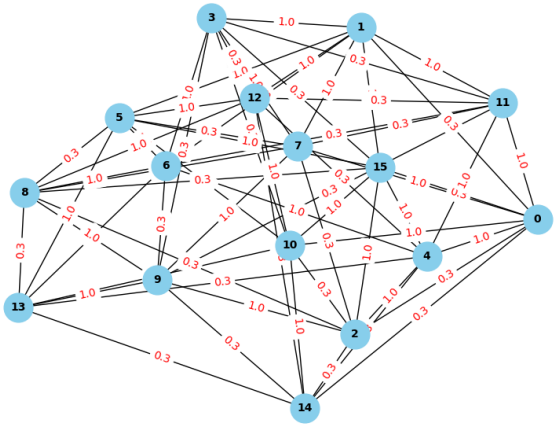


Figura 41: Grafo de orden N=16 y grado K=8.

9.4. Ortonormalización de Gram-Schmidt

El proceso de Gram-Schmidt es un método ampliamente utilizado en álgebra lineal para ortonormalizar un conjunto de vectores dentro de un espacio con producto interno, convirtiendo un conjunto de vectores linealmente independientes en un conjunto ortonormal. Para el procedimiento de Gram-Schmidt se plantea de la siguiente forma:

Inicializar el primer vector

$$v_1 = x_1$$

Construir el segundo vector

$$v_2 = x_2 - \frac{\langle x_2 \mid v_1 \rangle}{\langle v_1 \mid v_1 \rangle} v_1$$

Para todos los vectores en el espacio.

Normalización: Dado que se necesita un conjunto ortonormal, una vez que los vectores son calculados, se normalizan dividiéndolos por su norma, lo cual ya fue explicado previamente.
Ejemplo:

Suponga se tiene los siguientes vectores en $\mathbb{R}^2$:

$$x_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

- Inicializar el primer vector: el primer vector ortogonal es:

$$v_1 = x_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

- Paso 2: Construir el segundo vector: el segundo vector se calcula como:

$$v_2 = x_2 - \frac{\langle x_2 \mid v_1 \rangle}{\langle v_1 \mid v_1 \rangle} v_1,$$

donde:

1. El producto interno $\langle x_2 \mid v_1 \rangle$ es:

$$\langle x_2 \mid v_1 \rangle = \begin{bmatrix} 2 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = (2)(3) + (2)(1) = 6 + 2 = 8.$$

2. La norma al cuadrado de v_1 es:

$$\langle v_1 \mid v_1 \rangle = \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = (3)(3) + (1)(1) = 9 + 1 = 10.$$

Sustituyendo:

$$v_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \frac{8}{10} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \begin{bmatrix} 2{,}4 \\ 0{,}8 \end{bmatrix} = \begin{bmatrix} -0{,}4 \\ 1{,}2 \end{bmatrix}.$$

- Paso 3: Normalización: ahora normalizamos los vectores para obtener un conjunto ortonormal:

  1. Normalizamos v_1:

$$\|v_1\| = \sqrt{3^2 + 1^2} = \sqrt{9+1} = \sqrt{10}, \quad \hat{v}_1 = \frac{v_1}{\|v_1\|} = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

  2. Normalizamos v_2:

$$\|v_2\| = \sqrt{(-0{,}4)^2 + (1{,}2)^2} = \sqrt{0{,}16 + 1{,}44} = \sqrt{1{,}6}, \quad \hat{v}_2 = \frac{v_2}{\|v_2\|} = \frac{1}{\sqrt{1{,}6}} \begin{bmatrix} -0{,}4 \\ 1{,}2 \end{bmatrix}.$$

- Resultado final: el conjunto ortonormal resultante es:

$$\hat{v}_1 = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad \hat{v}_2 = \frac{1}{\sqrt{1{,}6}} \begin{bmatrix} -0{,}4 \\ 1{,}2 \end{bmatrix}.$$

El procedimiento de Gram-Schmidt es muy útil en el contexto de la computación cuántica al garantizar que los vectores sean ortogonales y normalizados, lo que facilita las operaciones, mediciones y cálculos dentro del espacio de Hilbert. Esto permite una mayor claridad y eficiencia en el diseño y análisis de algoritmos cuánticos, como los utilizados en QAOA (Nielsen y Chuang, 2010).