INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN Y DESARROLLO DE TECNOLOGÍA DIGITAL

MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

"BÚSQUEDA NO ESTRUCTURADA MEDIANTE CÓMPUTO CUÁNTICO"

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

PRESENTA
ALFREDO MIGUEL ARTEAGA CRUZ

BAJO LA DIRECCIÓN DE

DR. OSCAR HUMBERTO MONTIEL ROSS
DR. ULISES OROZCO ROSAS

NOVIEMBRE 2022 **TIJUANA, B.C., MÉXICO**

SIP-13
REP 2017

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REGISTRO DE TEMA DE TESIS
Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México, 19 de febrero del 2021

El Colegio de Profesores de Posgrado de Centro de Investigación y Desarrollo de Tecnología Digital en su Sesión
(Unidad Académica)

ordinaria No. 02 celebrada el día 18 del mes de febrero de 2021, conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	Arteaga	Apellido Materno:	Cruz	Nombre (s):	Alfredo Miguel

Número de registro: A 2 0 0 4 0 7

del Programa Académico de Posgrado: MCSD- Maestría en Ciencias en Sistemas Digitales

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

Búsqueda no estructurada mediante cómputo cuántico

Objetivo general del trabajo de tesis:

Desarrollar un algoritmo de cómputo cuántico e implementar en una computadora cuántica para resolver en problema de las N reinas.

2.- Se designa como Directores de Tesis a los profesores:

Director: Dr. Oscar Humberto Montiel Ross 2° Director: Dr. Ulises Orozco Rosas

No aplica: ☐

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

Centro de Investigación y Desarrollo de Tecnología Digital

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director(a) de Tesis

Dr. Oscar Humberto Montiel Ross

2° Director de Tesis

Dr. Ulises Orozco Rosas

Aspirante

Ing. Alfredo Miguel Arteaga Cruz

Presidente del Colegio

Dr. Julio César Rolón Garrido

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN Y DESARROLLO
DE TECNOLOGÍA DIGITAL
DIRECCIÓN

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de Tijuana B.C. siendo las 12:00 horas del día 05 del mes de diciembre del 2022 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado del: Centro de Investigación y Desarrollo de Tecnología Digital

para examinar la tesis titulada: Búsqueda no estructurada mediante cómputo cuántico

del (la) alumno (a):

Apellido Paterno:	ARTEAGA	Apellido Materno:	CRUZ	Nombre (s):	ALFREDO MIGUEL

Número de registro: A 2 0 0 4 0 7

Aspirante del Programa Académico de Posgrado: MAESTRÍA EN CIENCIAS EN SISTEMAS DIGITALES

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 16 % de similitud. **Se adjunta reporte de software utilizado**.

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI** [ ] **NO** [x] **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*
La mayoría de las similitudes reportadas por el software Turnitin se refiere a referencias bibliográficas, textos de documentos oficiales que deberá llevar la tesis, y nombres genéricos a problemas científicos y técnicas existentes.

****Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente, y posterior a la lectura, revisión individual así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR** [x] **SUSPENDER** [ ] **NO APROBAR** [ ] la tesis por **UNANIMIDAD** [x] o **MAYORÍA** [ ] en virtud de los motivos siguientes:
Es un trabajo de investigación con resultados de algoritmos y software realizados por el estudiante. Se cumplieron todos los objetivos y requisitos reglamentarios, obteniendose la publicación de un capitulo de libro idizado en Scopus.

COMISIÓN REVISORA DE TESIS

Dr. Oscar Humberto Montiel Ross
Director de Tesis
Nombre completo y firma

Dr. Moisés Sánchez Adame
Nombre completo y firma

Dr. Juan José Tapia Armenta
Nombre completo y firma

Dr. Ulises Orozco Rosas
2do. Director de tesis externo
Nombre completo y firma

M. en C. Adolfo Esquivel Martínez
Nombre completo y firma

Dr. Julio César Rolón Garrido
Nombre completo y firma
PRESIDENTE DEL COLEGIO DE PROFESORES

SEP
INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y DESARROLLO
DE TECNOLOGÍA DIGITAL
DIRECCIÓN

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN

En la Ciudad de México el día **18** del mes de **Noviembre** del año **2022**, el (la) que suscribe **Alfredo Miguel Arteaga Cruz,** alumno(a) del programa **Maestría en Ciencias en Sistemas Digitales**, con número de registro **A200407**, adscrito(a) a **Centro de Investigación y Desarrollo de Tecnología Digital** manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección de **Dr. Oscar Humberto Montiel Ross** y del **Dr. Ulises Orozco Rosas** y cede los derechos del trabajo intitulado **Búsqueda no estructurada mediante cómputo cuántico**, al Instituto Politécnico Nacional, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo. **marteaga@citedi.mx, amarteagac020995@gmail.com, oross@citedi.mx, ulises.orozco@cetys.mx**. Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.

Alfredo Miguel Arteaga Cruz

Nombre completo y firma autógrafa del (de la) estudiante

Agradecimientos

Al Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI) por brindarme la oportunidad de cursar un posgrado y al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo durante mi estadía en la ciudad de Tijuana durante el posgrado.

Búsqueda no estructurada mediante cómputo cuántico

Resumen

En este trabajo de tesis se presenta un algoritmo híbrido cuántico capaz de resolver el problema de las *N-Reinas* mediante el algoritmo de Grover. Se realizaron comparaciones de funcionamiento con algoritmos clasicos como: *Backtracking*, *Branch and Bound* y Programación Lineal, capaces de resolver el problema para diferente número de reinas.

En *Backtracking* las soluciones se identifican mediante la recursividad, omite las combinaciones que no satisfacen con las restricciones y avanza cuando la posición sea considerada como parte de la solución o retrocede en caso contrario.

En *Branch and Bound* se optimiza el uso del método de la fuerza bruta, identificando soluciones óptimas mediante divisiones en el conjunto de combinaciones, se delimita el espacio de búsqueda eliminando las ramificaciones que no favorecen a la solución.

La Programación Lineal es un método matemático que optimiza máximos o mínimos en una función objetivo, trabaja con restricciones e identifica las combinaciones seguras y delimita el área de búsqueda para colocar al menos una reina por fila, columna o diagonal (positiva o negativa).

El cómputo cuántico permitió procesar la información mediante qubits, identificando una o varias soluciones. Las soluciones se codificaron mediante un registro cuántico en donde los estados representaban las posiciones. Se identifica las combinaciones que cumplieron con las restricciones antes mencionadas.

Las circuitos cuanticos base para esta investigación fueron el oráculo y el algoritmo de Grover. La integración de los elementos antes mencionados permitió desarrollo del algoritmo híbrido cuántico, comprobándose que existe disminución en complejidad computacional y en consecuencia disminución en tiempo de procesamiento.

Palabras Clave: Cómputo cuántico, Algoritmo híbrido cuántico, Implementación cuántica, Algoritmo de Grover, Búsqueda no estructurada.

Unstructured search using quantum computing

Abstract

This thesis work presents a hybrid quantum algorithm capable of solving the *N-Queens* problem using Grover's algorithm. Performance comparisons were made with classical algorithms such as *Backtracking*, *Branch and Bound*, and Linear Programming, which can solve the problem for different numbers of queens.

In *Backtracking*, the solutions are identified by recursion, and it omits the combinations that do not satisfy the constraints and move forward when the position is considered as part of the solution or backward otherwise.

Branch and Bound optimizes the use of the brute force method, identifying optimal solutions through divisions in the set of combinations; the search space is delimited by eliminating the branches that do not favor the solution.

Linear Programming is a mathematical method that optimizes maxima or minima in an objective function, works with constraints, identifies safe combinations, and delimits the search area to place at least one queen per row, column, or diagonal (positive or negative).

Quantum computation allows processing the information using qubits, identifying one or more solutions. Then, the solutions were encoded using a quantum register where the states represented the positions. Finally, the combinations that complied with the restrictions mentioned above were identified.

The base quantum circuits for this research were the oracle and Grover's algorithm. The integration of the elements mentioned above allowed the development of the hybrid quantum algorithm, proving that there is a decrease in computational complexity and, consequently, a decrease in processing time.

Keywords: Quantum computing, Quantum hybrid algorithm, Quantum implementation, Grover's algorithm, Unstructured search.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

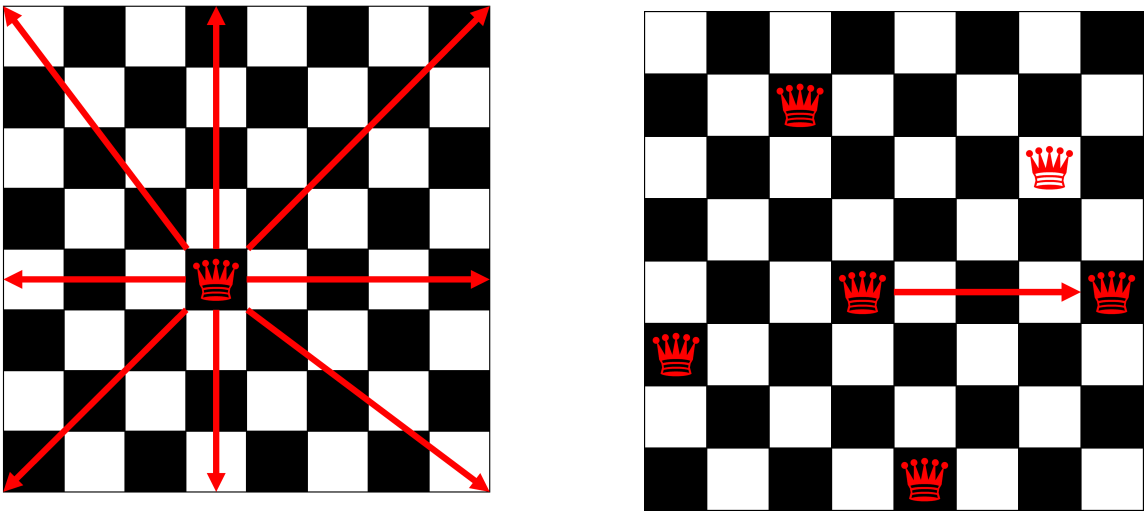
El cómputo cuántico es el procesamiento de información utilizando los principios de la mecánica cuántica [2]. Los dispositivos existentes hasta este momento se encuentran en la era *NISQ (Noisy Intermediate-Scale Quantum)*. Son prototipos que en un futuro serán consideradas computadoras cuánticas [3] y cuentan con la capacidad para demostrar las ventajas que el cómputo cuántico ofrece a pesar de sus limitaciones, como la cantidad limitada de qubits en las computadoras cuánticas, el número limitado de plataformas con acceso a ordenadores cuánticos, limitando el diseño de circuitos y la programación de los mismos, el tamaño maximo del registro cuántico lo que restringe el diseño de circuitos cuánticos que utilicen más qubits que la longitud máxima del registro cuántico. La profundidad, es decir la cantidad de circuitos cuánticos que pueden colocarse en serie sin que se pierda la información.

Esta tecnología tendrá la capacidad de disminuir el tiempo de procesamiento comparado con los algoritmos tradicionales [4–6] antes mencionados facilitando la creación de nuevos métodos para resolver problemas clasificados como NP, NP-Completo y NP-Duro (NP-Hard), caracterizados por su alta complejidad computacional y su elevado tiempo de procesamiento en medida que el número de entradas aumenta.

La solución se obtuvo desarrollando una variante híbrido cuántica que trabaja implementando la ejecución de diferentes algoritmos de cómputo tradicional, aplicando limitaciones establecidas por fila, columna y diagonal en conjunto con el algoritmo cuántico de Grover.

1.1. Antecedentes

El problema de las *8-Reinas* se propuso por Max Bezzel en 1848, el objetivo es colocar 8 reinas en un tablero de ajedrez [7] evitando que se amenacen entre ellas, de acuerdo con las reglas del ajedrez. Sus restricciones son: colocar una reina por fila, columna y diagonal (positiva y negativa) [8] como se muestra en la Figura 1.1. Se considera como solucionado cuando: se identifica una única combinación o todas las que cumplan con las restricciones antes mencionadas [9]. Carl Gauss (1850) [10] identificó un total de 72 combinaciones, sin embargo, en James Glaisher (1874) [7] probó que existen 92. Una vez resuelto se propuso la



(a) Movimientos de la reina sobre el tablero

(b) Posiciones de reinas en conflicto

Figura 1.1: Representación de movimientos y amenazas de reinas

variante de las *N-Reinas* donde N es un número entero positivo que representa las piezas a colocar en un tablero de tamaño $N \times N$.

El problema se ha solucionado mediante técnicas matemáticas y estadísticas, por ejemplo, utilizando algoritmos heurísticos y meta heurísticos, aplicando redes neuronales, mediante lógica difusa, con algoritmos ejecutados en dispositivos cuánticos, entre otros. Sirve para dar solución a problemas cotidianos como: detección de objetos [11], muestreo de espacio de píxeles [12], control de tráfico [13], prevención de bloqueos [14], en sistemas de detección del movimiento y trayectoria por medio de sensores [11], cálculo de la estimación del movimiento en una imagen [15] o pruebas de rendimiento como la evaluación de paralelismo a nivel instrucción y a nivel hilo [16], entre otros. Las pruebas realizadas en diferentes investigaciones permiten identificar un crecimiento y comportamiento exponencial en el tiempo de procesamiento con base en la entrada (número de reinas) definiendo al problema como uno de tiempo polinomial no deterministico (NP-Completo) [13].

En [17] se realiza una aproximación para 4 y 8 reinas para: conocer las combinaciones que satisfacen las restricciones, evaluar los patrones identificados, replicar, implementar y obtener las posiciones seguras para un total de 16 reinas de una manera más rápida. Se identificó una simetría cuando la cantidad de reinas es un número par y una asimetría en caso contrario, como se muestra en la Figura 1.2.

Los algoritmos heurísticos y meta heurísticos que han dado solución a esto son: [18] donde se ejecutó un algoritmo de murciélago basado en algoritmos genéticos para 8, 20, 50, 100 y 500 reinas, [1] mediante el desarrollo de un algoritmo genético basado en la aplicación de un operador de mutación avanzado para 8 y 50 reinas, comparando lo obtenido contra los resultados utilizando algoritmos genéticos sin operador de mutación avanzado, los resultados obtenidos se presentan en la Tabla 1.1 y en [19] usando un algoritmo genético de enjambre.

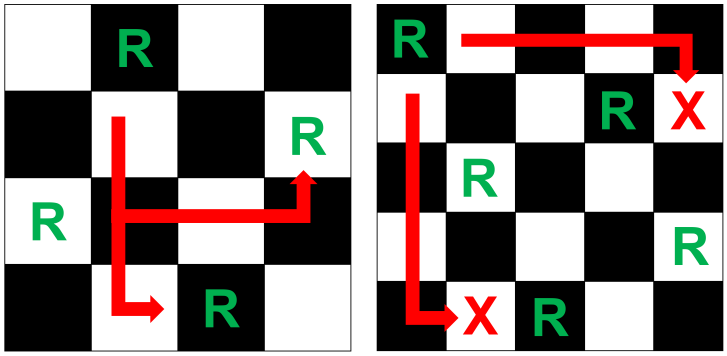


Figura 1.2: Representación de simetría con un número par de reinas (izquierda) y sin simetría con un número impar (derecha)

Tabla 1.1: Tiempo de procesamiento del algoritmo genético para encontrar las posiciones no amenazadas de las reinas, es decir, una solución al problema [1]

Número de reinas	Tiempo de procesamiento
8	0.0307 s
16	0.0972 s
30	0.3420 s
40	0.7885 s
50	1.0443 s
100	14.3688 s

En [20] se presenta una implementación en el cómputo paralelo, utilizando CPU, por sus siglas en inglés, *Central Processing Unit* y GPU, por sus siglas en inglés, *Graphics Processing Unit* para evaluar y solucionar el problema, aplica el algoritmo de recocido simulado para un número menor a 3,000 reinas mientras que en [21] se utiliza un modelo de algoritmo genético maestro-esclavo combinando el algoritmo evolutivo y el de recocido simulado, aplicando una función de aptitud paralela basada en la arquitectura de dispositivos unificados en una GPU, en [22] se aplica una aceleración de algoritmos genéticos con tarjetas de vídeo mediante modificaciones en las operaciones evolutivas ajustándolo a la arquitectura de una GPU.

Los métodos de Programación Lineal [23] encuentran la mejor solución mediante una función objetivo dentro de un conjunto de posibles soluciones definidas por un sistema de ecuaciones y desigualdades lineales. En [24] se establece el tamaño del tablero y el número de reinas a colocar, se identifican y aplican las restricciones correspondientes al igual que la función objetivo encargada de identificar que el número de reinas por fila y columna sea igual a 1 como se muestra en la Figura 1.3.

En [25] se validaron sub-espacios a través de una red neuronal modificada de Hopfield para identificar las combinaciones y soluciones posibles. De igual manera en [26] se aplicó una variación de una red neuronal de Hopfield discreta. En [27] se realiza una aproximación

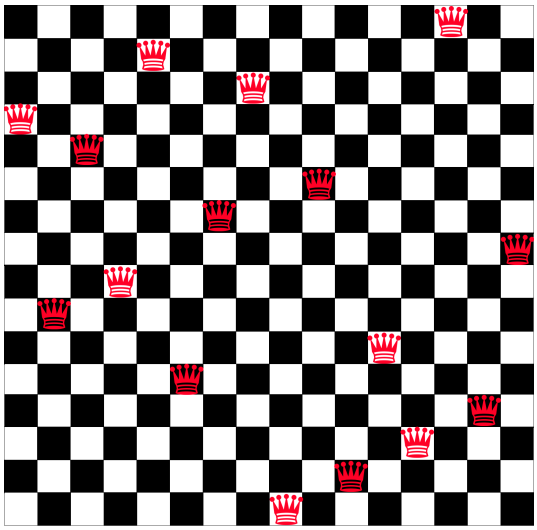


Figura 1.3: Solución para 16 reinas utilizando Programación Lineal

máxima de redes neuronales para encontrar un conjunto de ubicaciones e identificar las combinaciones que cumplan con las restricciones. Trabaja mediante la optimización de funciones objetivo sin restricciones demostrando una eficacia en el rendimiento mediante simulaciones, empleando hasta 500 reinas de modo secuencial comparado con otras redes neuronales existentes. En [28] se utilizó una red neuronal de predicción incorporando el algoritmo de procesamiento analítico de jerarquías (*Analytic Hierarchy Process*) para reducir el tiempo de predicción.

Mediante el uso de algoritmos de fuerza bruta, en [29] se propone la evaluación de un conjunto de posibles soluciones identificando a las que no llevan a una solución satisfactoria, depende completamente de prueba-error y cuenta con un gran nivel de ineficiencia por el número de iteraciones necesarias para eliminar las pseudo-soluciones que no favorecen al obtener el resultado correcto mientras identifica a las que si. La experimentación demostró que funciona eficientemente para un número de reinas menor a 12, sin embargo, para un valor mayor, el número de intentos incrementa exponencialmente, como se muestra en la Tabla 1.2. Se realizó la evaluación desde una reina hasta un máximo de trece en una computadora con un procesador Intel Xeon CPU E5-2609 v2 que trabaja a una frecuencia de 2.50 GHz, una memoria RAM de 32 GB en un sistema operativo Windows 7 enterprise.

En el área de cómputo cuántico, se identificaron las combinaciones que satisfacen las restricciones del problema utilizando el algoritmo de Grover como base [30], otra aproximación lograda en cuántico se presenta en [31], se propone una hibridación del algoritmo que trabaja con operadores de evolución diferencial como alternativa a la mutación en los algoritmos genéticos clásicos, se registró una mejora en los resultados con base en el tiempo de ejecución y evolución de ajuste con optimizaciones del 30 % - 40 % comparados con algoritmos de evolución diferencial en el cómputo tradicional.

Tabla 1.2: Tiempo de procesamiento registrado para las *N-Reinas* utilizando Backtracking

Número de reinas	Tiempo de procesamiento
1	≈ 0 s
4	≈ 0 s
5	≈ 0 s
6	0.0312 s
7	0.1716 s
8	0.7488 s
9	3.5724 s
10	17.8153 s
11	96.8610 s
12	558.0779 s
13	3,599.7074 s

1.2. Planteamiento del problema

Los algoritmos de optimización combinatoria resuelven problemas de tiempo polinomial identificando el mejor resultado dentro de un número finito de soluciones viables, evaluando los espacios de soluciones y disminuyendo su tamaño, permitiendo crear un espacio de búsqueda eficiente y facilitando la obtención de soluciones [32]. Presentan cierta dificultad, puede ser muy pequeña o muy grande y depende de el número de entrada, su procesamiento requiere de poco tiempo, sin embargo, cuando el número de entradas es considerable, requiere de herramientas basadas en algoritmos y optimizaciones para su solución.

En el cómputo tradicional se soluciona aplicando algoritmos convencionales y existen variaciones donde el tiempo de procesamiento crece dependiendo del número de entradas por lo que se busca realizar una implementación híbrido cuántica capaz de resolverlo, utilizando qubits, compuertas y algoritmos cuánticos y disminuir el tiempo de procesamiento.

1.3. Motivación

Los problemas clasificados como NP-Completos no cuentan con un algoritmo eficiente que pueda resolverlos y las computadoras tradicionales requieren de una gran cantidad de tiempo y procesamiento para resolver este tipo de problemas dependiendo de el número de entradas, por lo que surge la necesidad de desarrollar un algoritmo híbrido cuántico que aproveche el potencial que las computadoras cuánticas ofrecen para resolverlos en un tiempo menor al registrado en el computo clásico.

1.4. Hipótesis

Mediante el empleo de cómputo híbrido cuántico a través de la aplicación de las restricciones utilizadas en el cómputo tradicional y la implementación del algoritmo de Grover es posible encontrar soluciones al problema de las N-Reinas con un tiempo de procesamiento menor comparado con los algoritmos tradicionales.

1.4.1. Preguntas de investigación

- P1 : ¿Qué dificultades se identifican en los algoritmos de búsqueda para la solución al problema de las *N-Reinas* en el cómputo tradicional?
- P2 : ¿Cuál es el algoritmo para la solución al problema de las *N-Reinas* que tiene el menor tiempo de procesamiento y por qué?
- P3 : ¿Qué ventajas presenta el algoritmo cuántico de Grover que facilita la solución al problema de las *N-Reinas*?

1.5. Objetivo general

Desarrollar un algoritmo para solucionar el problema de las *N-Reinas* utilizando cómputo híbrido cuántico e identificar las ventajas y desventajas que presenta en comparación con el cómputo tradicional.

1.6. Objetivos específicos

A continuación se presentan

- Entender el estado del arte para identificar las aportaciones que se han realizado hasta el momento para resolver el problema de las *N-Reinas* utilizando cómputo tradicional.
- Analizar los algoritmos representativos para dar solución al problema de las *N-Reinas* utilizando cómputo tradicional.
- Entender el estado del arte para identificar las aportaciones que se han realizado hasta el momento para resolver el problema de las *N-Reinas* utilizando cómputo cuántico.
- Desarrollar la propuesta de solución para resolver el problema de las *N-Reinas* utilizando cómputo híbrido en una computadora cuántica o simulador.
- Implementar la propuesta de solución para demostrar su funcionamiento utilizando un simulador cuántico o una computadora cuántica.

- Interpretar los resultados obtenidos para comparar el desempeño entre el cómputo tradicional y cuántico.

1.7. Aportaciones

En este trabajo de tesis se presenta una alternativa para resolver problemas combinatorios clasificados como NP-Completo, como es el caso con las *N-Reinas*. Esta propuesta se basa en un algoritmo híbrido cuántico utilizado para encontrar combinaciones en un tablero de $N \times N$ que satisfacen a las restricciones por fila, columna y diagonal, identificando las posiciones seguras y no seguras a través del algoritmo de Grover, con base en una búsqueda no estructurada de datos para encontrar una solución dentro de un conjunto de estados, disminuyendo el tiempo de búsqueda y complejidad computacional.

El algoritmo híbrido cuántico reportado en este documento de tesis trabaja bajo la manipulación de un determinado número de qubits a través de compuertas cuánticas, cumpliendo con las restricciones de posicionamiento establecidas y permitiendo la identificación de estados o combinaciones considerados como resultados mediante el algoritmo de Grover, todo esto, utilizando una plataforma con conexión a una computadoras y simuladores cuánticos. Adicional a esto se desarrolló y aceptó el documento *"Evaluation and comparison of brute-force search and constrained optimization algorithms to solve the N-Queens problem"* como capítulo en el libro *"New Perspectives on Hybrid Intelligent System Design based on Fuzzy Logic, Neural Networks and Metaheuristics"* perteneciente a la editorial Springer y se obtuvo una participación en el *"International Seminar on Computational Intelligence"* en el ISCI 2022 bajo el nombre *"N-Queens problem solution using the Grover's algorithm"*.

Capítulo 2

Marco Teórico

En este apartado se presentan los fundamentos matemáticos que describen el problema de las *N-Reinas*, los modelos aplicados para obtener la solución y el funcionamiento de los algoritmos tradicionales (Backtracking, Branch and Bound, Programación Lineal) y cuánticos (Grover), implementados para obtener resultados que permiten comparar los métodos y dar forma a este trabajo de tesis.

2.1. Problema de las *N-Reinas*

Propuesto por Max Bezzel, consta de colocar 8 reinas en un tablero de 8×8, al resolverse su enfoque cambió abriendo paso a una nueva variante del problema. En ella se busca colocar un N número de reinas en un tablero con tamaño $N \times N$ [33]. Para su solución se establecieron condiciones que permiten la distribución de piezas sobre el tablero de ajedrez y se definen como: al menos una reina se coloque en la misma fila, por columna y por diagonal (positiva y negativa), como se muestra en la Figura 2.1. Existen escenarios donde no existe una solución, como es el caso con 2 reinas y donde existen una o más capaces de solucionar el problema.

Para su solución se necesita crear una matriz de tamaño $N \times N$ que simboliza el tablero de ajedrez con casillas enumeradas desde $0, ..., N-1$. La posición se representa como un conjunto de pares ordenados denominados (r, c) y (x, y) donde r y x son la filas y c y y columnas. La distribución de las reinas debe de cumplir con las restricciones establecidas, verificando que no existan amenazas durante la identificación de combinaciones consideradas como soluciones [34], tal que las reinas posicionadas en (r, c) y en (x, y), deben de cumplir con lo siguiente:

- El valor de la fila r debe de ser diferente al de x.
- El valor de la columna c debe de ser diferente al de y.
- El valor de la diagonal positiva $r + x$ debe de ser diferente al de $c + y$.
- El valor de la diagonal negativa $r + y$ debe de ser diferente al de $c + x$.

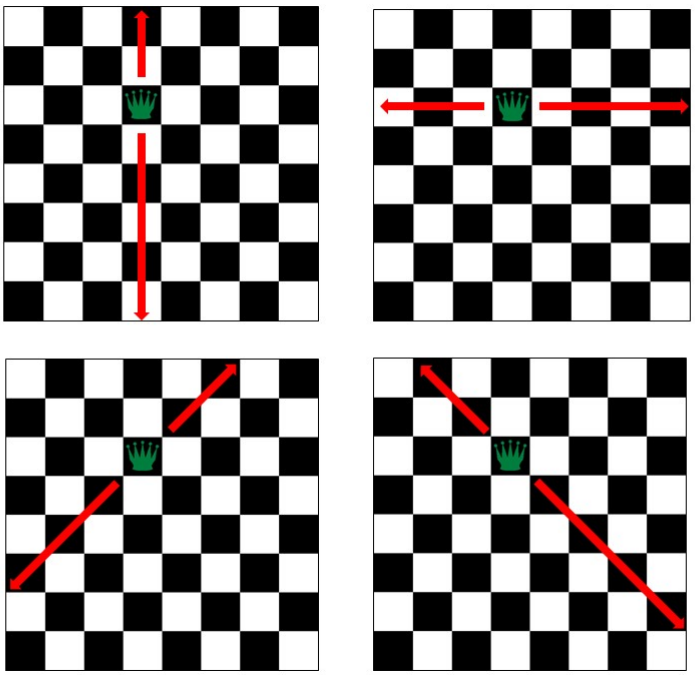


Figura 2.1: Restricciones de las *N-Reinas*

Lo anterior se puede expresar de la siguiente manera:

$$(r,c)(x,y) \triangleq r \neq x \wedge c \neq y \wedge r + x \neq c + y \wedge r + y \neq c + x \tag{2.1}$$

Los resultados presentados en el estado del arte permiten identificar que el tiempo de procesamiento es mínimo cuando el número de reinas N es menor a 12, e incrementa exponencialmente al aumentar el número de piezas, demandando un mayor procesamiento que requerirá de un tiempo polinomial no determinista para poder resolver el problema, es por esto que lo anterior se considera como NP-Completo y su complejidad computacional depende directamente del algoritmo aplicado para su solución. Los algoritmos de fuerza bruta trabajan con una complejidad de orden $O(N^N)$ [35], como el caso de Backtracking con una complejidad $O(N!)$ [36], mientras que la complejidad al utilizar el algoritmo de Grover es de $O(\sqrt{N})$ [37].

2.2. Modelos matemáticos para la solución del problema de las *N-Reinas*

El proceso para la obtención de combinaciones se realiza a través de restricciones, definidas como un número limitado de reinas por fila, columna y diagonal, asegurando que las piezas no se encuentren amenazadas. Existen varios enfoques algorítmicos, como la solución basada en el posicionamiento por filas y columnas [33, 38]. A continuación se explican dos

modelos que trabajan de acuerdo a las implementaciones logradas.

2.2.1. Modelo matemático 1

Considerado como el más generalizado y fácil de implementar, utiliza un modelo de variable binaria X_{ij} donde i representa a las columnas y j a las filas. Cuando $X_{ij} = 1$ significa que la posición es segura, de lo contrario $X_{ij} = 0$.

Las filas cuentan con una condicional que evita que dos o más piezas se encuentren en la misma fila. Si el resultado es diferente a 1, la restricción descartará a la combinación en curso, definido como:

$$\sum_{j=1}^{N} X_{ij} = 1 \qquad \forall i = \{1, ..., N\} \tag{2.2}$$

Las columnas trabajan de manera similar, cuentan con una restricción que evita que dos o más piezas se encuentren en la misma linea mediante la siguiente ecuación:

$$\sum_{i=1}^{N} X_{ij} = 1 \qquad \forall j = \{1, ..., N\} \tag{2.3}$$

Las diagonales incrementan la dificultad, por lo que es necesario identificar que el total de reinas que se encuentren en una misma diagonal (positiva o negativa) sea a lo más una, tal como se muestra en la Figura 2.2.

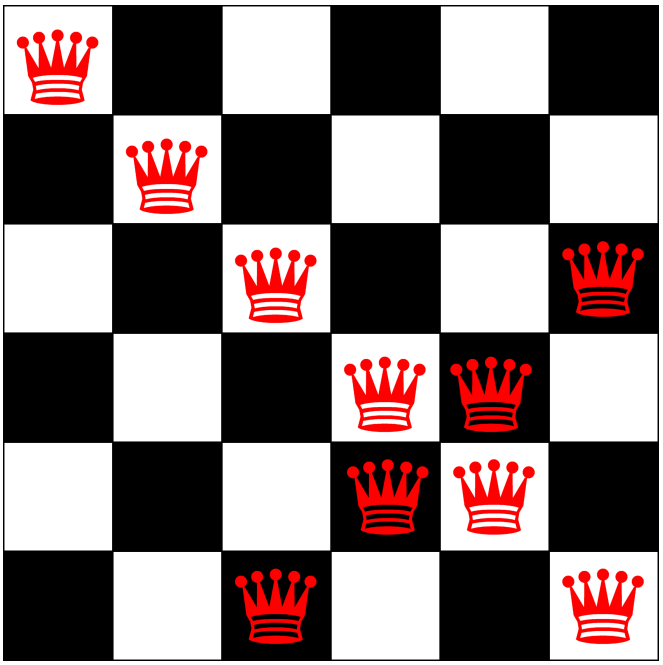


Figura 2.2: Diagonales positivas y negativas

Para verificar la cantidad de reinas por diagonal positiva se aplica la siguiente ecuación:

$$k = i + j \tag{2.4}$$

De esta forma, la suma de todos los valores k de los pares (i, j) deberá de ser 1, asegurando que el total de reinas cumpla con las restricciones y garantizando que esta se encontrará en una posición libre de amenazas. Para considerar diferentes tamaños de tablero y diferente número de reinas se debe de realizar una generalización que contemple estos aspectos, la cual se presenta a continuación.

$$\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij} \leq 1 \qquad \forall k = \{2, ..., 2N\} \tag{2.5}$$

Evaluar las diagonales negativas es posible utilizando una idea similar al anterior, esto es

$$k = i - j \tag{2.6}$$

Obtener el total de piezas se logra mediante la siguiente ecuación:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij} \leq 1 \qquad \forall k = \{-N + 1, ..., N - 1\} \tag{2.7}$$

Con base en lo anterior, el espacio de búsqueda de este modelo es $2^{N\times N}$ para un total de $68,719,476,736$ combinaciones y con un número de reinas de $N = 6$.

2.2.2. Modelo matemático 2

Utiliza la variable X_i para representar a todas las filas de la posición i en el dominio $1, ..., N$. Las restricciones trabajan con las columnas y diagonales. Las filas son evaluadas por defecto con un valor asignado automáticamente, de esta forma no puede haber dos o más reinas. Se verifica observando el número de piezas identificadas. Se define de la siguiente manera:

$$X_i \neq X_j \qquad \forall i \neq j \qquad i \; y \; j \in N \tag{2.8}$$

De forma que:

$$X_i - X_j \neq 0 \qquad \forall i \neq j \qquad i \; y \; j \in N \tag{2.9}$$

Similar al primer modelo, las diagonales positivas y negativas deben de contener a lo más una reina, la forma de las mismas se define como un ángulo de $\pm 45°$. Con base en la definición del modelo donde el valor de la variable indica la columna y fila se determina si dos reinas se encuentran en la misma diagonal. Visualizando la posición de las piezas como puntos en el plano cartesiano se tiene que $R_i(x_i, y_i)$ para la primera reina y $R_j(x_j, y_j)$ para la segunda. Con los valores definidos utilizando la nomenclatura del modelo se tiene $R_1 = (x_1, i)$ y $R_2 = (x_j, j)$; entonces $\Delta x = x_j - x_1$ y $\Delta y = j - i$. La tangente del ángulo se calcula usando $\frac{\Delta y}{\Delta x}$, y conociendo que $tan^{-1}45 = 1$ entonces $\Delta x = \Delta y$. Por lo tanto, dicha restricción se

define como:

$$x_i - x_j \neq i - j \qquad \forall i \neq j \qquad i\ y\ j \in N \tag{2.10}$$

2.3. Algoritmos aplicados al cómputo tradicional para la solución del problema de las *N-Reinas*

Existen diferentes formas de solucionar estos problemas y se han desarrollado diferentes mejoras con la finalidad de disminuir el tiempo de procesamiento necesario para obtener cada una de las combinaciones posibles con diferente número de entradas. El objetivo de los algoritmos que se presentan a continuación es el de conocer la forma en como trabajan, tal es el caso de "Backtracking", "Branch and Bound" y Programación Lineal.

2.3.1. "Backtracking"

Es un método recursivo, tiene como finalidad encontrar soluciones viables capaces de resolver problemas de optimización combinatoria mediante la fuerza bruta. Trabaja con la ejecución de tareas de forma secuencial, evaluando paso a paso cada dato. Es clasificado como un algoritmo de búsqueda exhaustiva y todas las soluciones viables son consideradas y evaluadas hasta encontrar una solución óptima, de lo contrario, la posible solución se descarta, regresa a un estado anterior y comienza con una nueva evaluación. Esto se realizará un determinado número de veces hasta encontrar todas las combinaciones posibles [39]. Un árbol de combinaciones es una manera práctica y sencilla de representar al algoritmo, como se muestra en la Figura 2.3 donde se realiza una evaluación de todos los datos para descartar todos aquellos que no conducen a una respuesta satisfactoria, por lo que el número de iteraciones y validaciones incrementa a medida de que el tamaño del problema aumente [40].

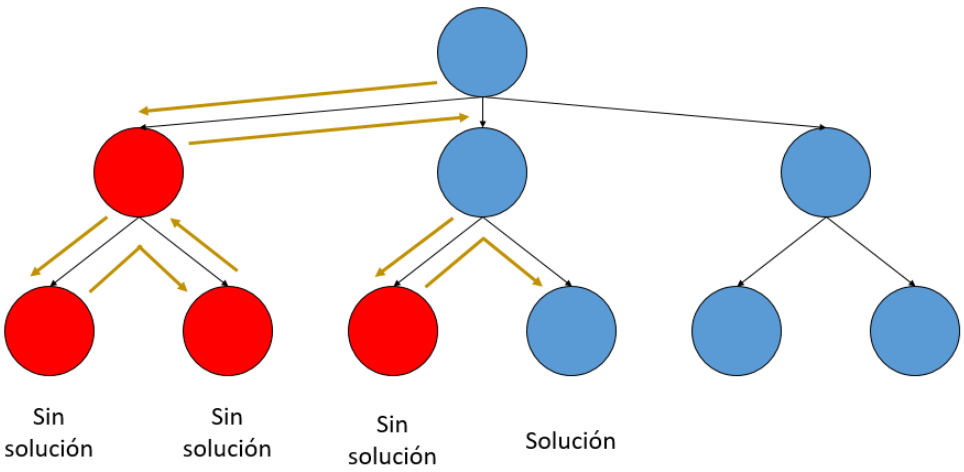


Figura 2.3: Representación gráfica de funcionamiento del algoritmo "Backtracking".

Este método se ha implementado en la solución de las *N-Reinas* considerando todas las posiciones de las piezas como soluciones parciales y evaluando cada combinación con las restricciones establecidas, identificando las casillas seguras y el total de combinaciones que cumplen con las limitaciones por fila, columna y diagonal. El algoritmo se encarga de trabajar con cada pieza colocada y cuando identifique una posición no segura regresará a un estado anterior para continuar por un nuevo camino, asegurando la integridad de cada pieza sobre el tablero de ajedrez [41].

Algoritmo 1: Backtracking

```
Entrada: N ← Número de reinas
Salida: X[i, j] ← Posición segura
Crear Tablero ← Tamaño N × N
si Reinas colocadas = N entonces
    Regresar True
en otro caso
    Asignar j ← [0, 1, ..., N]
    si [i,j] es segura entonces
        Asignar X[i, j] ← 1
        Aumentar j = j + 1
        Almacenar Tablero ← [i, j]∀X[i, j] = 1
        Regresar True
    en otro caso
        Disminuir j = j − 1
        Aumentar i = i + 1
    fin
    si [i + n,j] no es seguro entonces
        Regresar False
        Fin del algoritmo
    fin
fin
```

El Algoritmo 1 muestra su funcionamiento y comienza con la creación de la posición inicial que se realiza de manera ordenada, inicia con una posición inicial $[i, j] = [0, 0]$ en el tablero, se coloca una reina y, partiendo de esta ubicación, el algoritmo comenzará a evaluar las casillas adyacentes. Tras validar las posiciones se identifican los espacios seguros e inseguros pertenecientes a la primera posición, verificando que únicamente se encuentre una reina en la fila, columna y diagonales actuales. Al realizar este paso, el algoritmo cambiará de columna modificando la posición tal que $j = j + 1$, paso seguido se validan las restricciones con respecto a la posición de la reina anterior y en caso de encontrarse amenazada alternará entre filas realizando pasos hacia atrás. Estas validaciones se realizarán hasta que todas las reinas se hayan colocado en el tablero.

2.3.2. "Branch and Bound"

Este método cuenta con optimizaciones para realizar la búsqueda de casillas seguras en un menor tiempo. Resuelve problemas utilizando una optimización global con variables enteras, objetivos y constantes lineales. Cuenta con una estructura similar al "Backtracking",

reduciendo la cantidad de iteraciones y ciclos necesarios para encontrar una solución mediante la eliminación de ramificaciones [42] y aplicando dos condiciones:

1. Eliminar la ramificación siempre y cuando las soluciones obtenidas no sean factibles comparadas con las soluciones previamente encontradas.

2. Eliminar la ramificación cuando la solución no cumpla con los requerimientos para que sea considerada como solución.

La representación gráfica del algoritmo se muestra en la Figura 2.4. Se ha empleado para resolver el problema de las *N-Reinas* encontrando una solución óptima mediante diferentes combinaciones [43]. Se puede representar como un árbol de soluciones donde cada rama puede llevar a una posible solución y para confirmarlo se realizará una evaluación, si esta contiene un resultado favorable se considerará como parte de la solución, en caso contrario se omitirá y no se tomarán en cuenta las ramificaciones subsecuentes a partir del dato evaluado, dicha acción se le conoce como "poda" y con esto se logra una optimización sobre el tiempo de ejecución del problema y disminuye el tiempo de procesamiento necesario para encontrar las soluciones al problema de las *N-Reinas*. Al igual que el método anterior, se realiza un *Backtrack* o paso hacia atrás que se ejecutará únicamente cuando la siguiente fila ya no cuenta con opciones para evaluar, por lo que esa rama se poda para evitar continuar con una ejecución de tareas que se conoce no va a entregar ningún resultado [44].

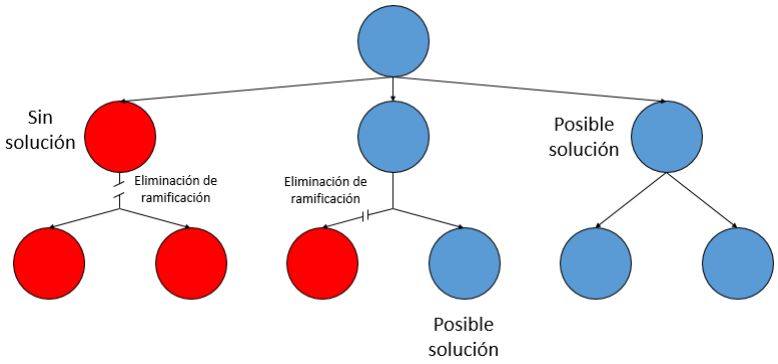


Figura 2.4: Representación gráfica de funcionamiento del algoritmo "Branch and Bound"

Para su solución se establece el número de piezas con las que se trabajará (N) y el espacio de trabajo de un tamaño $N \times N$. Se inicializa colocando una reina en el espacio $[i, j] = [0, 0]$ validando los espacios adyacentes y marcándolos como posiciones no seguras, en este punto se están omitiendo una gran cantidad de combinaciones que no llevarán a ninguna solución. Después se identificarán las posiciones seguras y se colocará la siguiente reina, evaluando las casillas de las filas, columnas y diagonales vecinos disminuyendo el espacio de búsqueda sustancialmente. Se validan las nuevas posiciones seguras y en caso de no contar con ellas, se retrocede un paso y se alterna con otra posición segura hasta colocar todas las piezas, tal como se explica en el Algoritmo 2.

Algoritmo 2: Branch and Bound

```
Entrada: N ← Número de reinas
Salida: X[i, j] ← Posición segura
Crear Tablero ← Tamaño N × N
si Reinas colocadas = N entonces
    Regresar True
en otro caso
    Asignar j ← [0, 1, ..., N]
    Recorrer filas i ← [0, 1, ..., N]
    si [i,j] es segura entonces
        Asignar X[i, j] ← 1
        Cambiar Columna a j = j + 1
        Regresar True
    en otro caso
        Omitir la posición [i, j]
        Marcar posiciones i[0, 1, ..., N] ← 0
        Cambiar a columna j = j + 1
    fin
    si i[0, 1, ... , N] no son seguras entonces
        Regresar False
        Fin del algoritmo
    fin
fin
```

2.3.3. Programación Lineal

Trabaja mediante optimizaciones para encontrar una solución óptima, está definida por un conjunto de ecuaciones y desigualdades lineales [23] y se compone de dos elementos, la función objetivo que trabaja con una declaración cuantificada y las restriccione que delimitan los valores de las variables en un modelo matemático [45]. Resolver el problema de las *N-Reinas* necesita de un método matemático para generar un modelo lineal, generando soluciones para maximizar o minimizar un proceso y tomar decisiones que componen a la solución, su función objetivo se describe como la sumatoria de reinas, definidas como X y localizadas en una coordenada (i, j) de la siguiente manera:

$$R = \sum_{i=1}^{N} \sum_{j=1}^{N} X[i, j] \tag{2.11}$$

Mientras que sus restricciones se definen como las filas, columnas y diagonales donde las reinas no pueden ser colocadas y se expresan de la siguiente manera:
La restricción 1: La sumatoria de las reinas por columna debe de ser igual a 1.

$$N \sum_{j=1}^{N} X[i, j] \leq 1 \tag{2.12}$$

La restricción 2: El número de piezas por columna debe de ser igual a 1.

$$N \sum_{i=1}^{N} X[i,j] \leq 1 \tag{2.13}$$

La restricción 3: Al menos una reina debe de posicionarse por diagonal positiva.

$$(N-2) \sum_{i,j \in 1 \to N, i-j=k}^{N} X[i,j] \leq 1 \tag{2.14}$$

La restricción 4: Similar a la restricción 3, el número de reinas por diagonal negativa debe de ser de 1 o menor.

$$(2N-1) \sum_{i,j \in 1 \to N, i+j=k}^{N} X[i,j] \leq 1 \tag{2.15}$$

El Algoritmo 3 muestra los parámetros iniciales como el número de reinas a evaluar, la matriz que representa al tablero de ajedrez y los pasos a seguir para encontrar las combinaciones y soluciones.

Algoritmo 3: Programación Lineal

```
Entrada: N ← Número de reinas
Salida: X[i,j] ← Posición segura
Crear Tablero ← Tamaño N × N
si Reinas colocadas = N entonces
    Regresar True
en otro caso
    Colocar la reina X[i,j]
    Ejecutar OR-Tools de Google Librería de especializada de Prog. Lineal
    si [i,j] es segura entonces
        Asignar X[i,j] ← 1
        Evaluar posición
        Regresar True
    en otro caso
        Evaluar nueva posición en [i, j+1]
    fin
    si Reinas colocadas = N entonces
        Regresar False
        Fin del algoritmo
    fin
fin
```

2.4. Cómputo cuántico

Es una rama de la informática que se encuentra en una etapa de desarrollo, trabaja bajo los principios de la física cuántica y con el uso de funciones probabilísticas [46]. Su función es la de procesar información representada por estados cuánticos aprovechando los fenómenos como la "superposición" y el "entrelazamiento".

Los dispositivos cuánticos abordan ciertos tipos de problemas más rápido que cualquier computadora clásica, involucran un inmenso número de variables, resultados potenciales, simulaciones y optimizaciones. Se han identificado diversas aplicaciones como el cálculo y solución de problemas computacionales "NP-Difícil" y "NP-Completo" [47]. Los algoritmos cuánticos desarrollados hasta este momento favorecen en la solución de problemas difíciles, como la resolución de problemas de optimización utilizando herramientas de simulación cuántica con optimizadores cuánticos, construcción de bloques en un ordenador cuántico con el propósito de resolver problemas Hamiltonianos [48], entre otros. Estos dispositivos se caracterizan por utilizar "qubits" (*Quantum bits*) como unidad de información y funcionan con partículas subatómicas [49] modificando sus valores mediante el uso de compuertas al momento de realizar operaciones.

La información se representa en qubits y trabajan con dos estados, $|0\rangle$ y $|1\rangle$. La diferencia que existe entre estos y los bits clásicos es la cantidad de estados con los que se puede trabajar, en el cómputo clásico se realiza con un estado, mientras que en cuántico es con múltiples ($|0\rangle$ o $|1\rangle$) a través de una combinación lineal entre los estados previamente mencionados que permite el paralelismo cuántico y una evaluación simultanea de los valores 0 y 1, llamado superposición y el resultado se describe como un nuevo estado [50, 51]. La representación básica de un estado con un qubit se define como un vector de columna de dos dimensiones, este contiene toda la información necesaria utilizada para poder describir al sistema cuántico y se representa de la siguiente manera:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.16}$$

Donde α y β son números complejos, $|\psi\rangle$ es el qubit resultante y $|0\rangle$ y $|1\rangle$ son los estados de qubit que serán modificados en el espacio de Hilbert.

La esfera de Bloch [52] es una herramienta que facilita la visualización de los estados cuánticos; consta de vectores que representan la posición de un qubit según el estado en donde se encuentre y permite la representación de la evolución del estado mediante rotaciones en la misma esfera. Un ejemplo se muestra en en la Ecuación 2.16:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.17}$$

Al representar el estado de la Ecuación 2.17 se obtiene:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \text{ entonces } \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \therefore |\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.18}$$

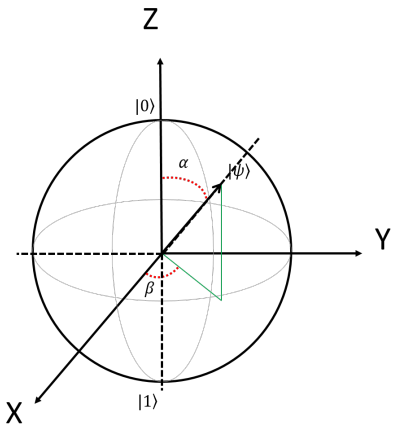


Figura 2.5: Representación de esfera de Bloch

2.5. Compuertas cuánticas

Son operaciones unitarias encargadas de modificar los valores de los qubits y de realizar operaciones entre ellos. Condicionan los valores del estado del qubit objetivo y el resultado se representa como estado de un qubit controlado. Algunas compuertas básicas y su representación matricial se muestran a continuación.

$\boxed{I}$	Compuerta de identidad	$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$\boxed{X}$	Compuerta de Pauli X	$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
$\boxed{Y}$	Compuerta de Pauli Y	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
$\boxed{Z}$	Compuerta de Pauli Z	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
$\boxed{H}$	Compuerta Hadamard	$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

2.5.1. Compuertas cuánticas de un qubit

Su función es la de modificar el valor de los qubits y poder realizar operaciones entre ellos a través del entrelazamiento y la superposición cuántica, como se muestra a continuación [53].

Compuerta *X*

Su función es la de invertir el estado en el que se encuentre el qubit de la siguiente manera:

$$\alpha\,|0\rangle + \beta\,|1\rangle \;\text{—}\boxed{X}\text{—}\; \alpha\,|1\rangle + \beta\,|0\rangle \tag{2.19}$$

La representación matemática es:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.20}$$

El qubit realiza un movimiento en el eje X con una rotación de π radianes, se puede representar como se muestra en la Figura 2.6 mediante la esfera de Bloch.

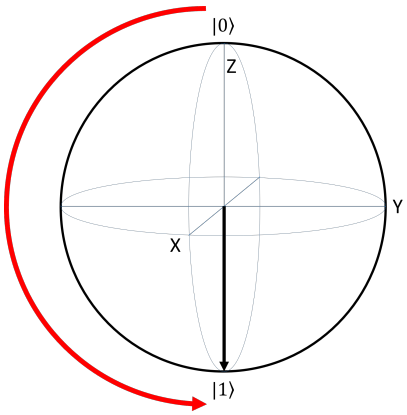


Figura 2.6: Movimiento de la esfera de Bloch en el eje X

Compuerta *Y*

Modifica el estado del qubit alrededor del eje Y realizando un cambio en el estado $|0\rangle$ a $|1\rangle$. Su representación y funcionamiento es el siguiente.

$$|0\rangle \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{—}\boxed{Y}\text{—} |0\rangle \begin{bmatrix} 0 \\ -i \end{bmatrix} \tag{2.21}$$

La representación matemática es:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.22}$$

El qubit realiza un movimiento en el eje Y y se representa como se muestra en la Figura 2.7.

Compuerta *Z*

Modifica la posición del qubit sobre el eje Z mediante una rotación de un valor de π radianes.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.23}$$

Al aplicar la compuerta Z se obtiene el siguiente resultado.

$$\alpha\,|0\rangle + \beta\,|1\rangle \text{—}\boxed{Z}\text{—} \alpha\,|0\rangle - \beta\,|1\rangle \tag{2.24}$$

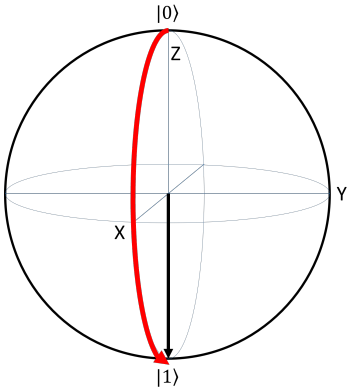


Figura 2.7: Movimiento de la esfera de Bloch en el eje Y

La representación matemática es la siguiente:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.25}$$

El movimiento realizado en el eje Z utilizando la esfera de Bloch se muestra en la Figura 2.8.

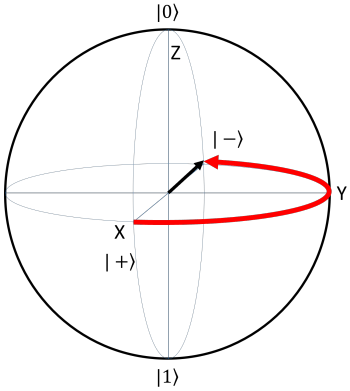


Figura 2.8: Movimiento de la esfera de Bloch en el eje Z

Compuerta Hadamard

Cambia el estado inicial del qubit de $|0\rangle$ o $|1\rangle$ a una superposición de los mismos, realiza una rotación de valor π sobre el eje de las X y una rotación de $\frac{\pi}{2}$ sobre el eje de las Y.

$$\alpha\,|0\rangle \;\text{—}\boxed{H}\text{—}\; \alpha\,|0\rangle \tag{2.26}$$

La representación matemática es la siguiente:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.27}$$

Se encarga de establecer los valores de los estados de $|0\rangle$ y $|1\rangle$ a $|+\rangle$ y $|-\rangle$ respectivamente.

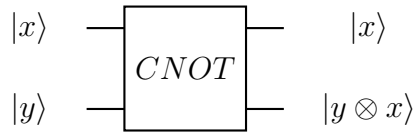
$$\begin{aligned} |0\rangle \text{ —}\boxed{H}\text{— } \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \\ |1\rangle \text{ —}\boxed{H}\text{— } \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{2.28}$$

2.5.2. Compuerta de dos qubits *CNOT*

Es una compuerta controlada y opera con dos o más qubits. Está compuesta por un qubit control cuyo valor será invariante durante la aplicación y un qubit objetivo que niega el valor cuando el qubit control tiene un valor igual a 1, en caso contrario el valor es igual a 0. Se puede representar de la siguiente manera.

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle \end{aligned} \tag{2.29}$$

Su representación en un circuito cuántico es la siguiente.



(2.30)

2.5.3. Entrelazamiento cuántico

También llamado Estado de Bell, es una propiedad permite la unión de dos qubits y los cambios de estados aplicados en uno modificarán al segundo de forma inmediata [51]. Incrementa la velocidad de procesamiento en las computadoras cuánticas y el número de operaciones que se pueden realizar con diferentes qubits al mismo tiempo [54]. Se representa usando dos o más qubits, tal como se muestra a continuación.

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.31}$$

Tiene la capacidad de no descomponerse en factores independientes y la medición resultante puede ser una de las siguientes:

- Medición $1 = 0$ con la probabilidad $p = \frac{1}{2}$ y el estado final sería $|\Psi'\rangle = |00\rangle$
- Medición $2 = 1$ con la probabilidad $p = \frac{1}{2}$ y el estado final sería $|\Psi'\rangle = |11\rangle$.

Los estados de Bell representativos se encuentran en la base computacional Z, se generan a partir del entrelazamiento de dos qubits, se denotan como $|\beta_{xy}\rangle$ y son los siguientes.

$$\begin{aligned} |\beta_{00}\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |\beta_{01}\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\beta_{10}\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ |\beta_{11}\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned} \tag{2.32}$$

Y es posible representarlos usando la siguiente formula.

$$|\beta_{xy}\rangle = \frac{1}{\sqrt{2}}(|0y\rangle + (-1)^x |1\bar{y}\rangle \tag{2.33}$$

Esta propiedad es clave para poder resolver el problema de las *N-Reinas*, permite trabajar con múltiples qubits al mismo tiempo y aplicando algoritmo de Grover facilita la identificación de estados de múltiples qubits entrelazados, al momento de extrapolar sus valores y mediciones se obtienen combinaciones que favorecen y cumplen con las restricciones de posicionamiento, obteniendo una solución satisfactoria al problema.

2.5.4. Algoritmos cuánticos

Realizan una codificación cuántica de datos a qubits, procesan una serie de operaciones mediante diferentes compuertas cuánticas modificando sus valores y evalúan el resultado obtenido durante su proceso. Tienen como finalidad identificar soluciones a problemas complejos que no se pueden resolver en una computadora clásica. Algunos de los algoritmos base dentro del cómputo cuántico son el de Deutsch - Jozsa, el de Simon, de Shor y de Grover [55]. El presente problema utilizará el de Grover como base para identificar las posiciones amenazadas y seguras en en donde es posible colocar las piezas de ajedrez.

2.5.5. Oráculo

Es un elemento compuesto por un conjunto de compuertas cuánticas que identifican una solución o estado cuántico dentro de un conjunto finito de datos. Genera una polarización a un valor y lo verifica con base en el resultado o estado deseado. [56, 57]. El oráculo es parte importante en el desarrollo del algoritmo híbrido cuántico, identificará el estado que cumpla con las restricciones validará que cumpla con las restricciones establecidas.

2.5.6. Algoritmo de Grover

Su función es realizar búsquedas en una base de datos con N número de entradas hasta encontrar un valor determinado. Su complejidad computacional se define como $O(\sqrt{N})$. Se propuso por Lov Grover (1996) y presenta una aceleración cuadrática en el procesamiento de datos con una aceleración exponencial en comparación con sus equivalentes en el cómputo tradicional [58].

Este algoritmo trabaja con rotaciones en el espacio de Hilbert con relación al estado $|s\rangle$ y con un conjunto de elementos en la base de datos $|w'\rangle$ donde N es el número de elementos y M es el número de soluciones [59]:

$$|s\rangle = \sqrt{\frac{N-M}{N}}\,|s'\rangle + \sqrt{\frac{M}{N}}\,|w\rangle \tag{2.34}$$

Se compone de dos operadores de reflexión, U_{ora} que es el resultado obtenido por el oráculo y U_s que representa el operador de difusión de Grover. Tiene como finalidad mejorar la amplitud de $|w\rangle$ mediante una reflexión con valor $U_{ora} = I - 2\,|w\rangle\,\langle w|$ y modificando el valor de la solución de manera que $U_s = 2\,|s\rangle\,\langle s| - I$. La amplitud máxima para las soluciones se obtiene al realizar repetidas iteraciones en el algoritmo, de forma que $Iteraciones \leq \frac{\pi}{4}\sqrt{\frac{N}{M}}$ [60].

Su funcionamiento se basa en la creación de una superposición de todos los estados bases utilizando la compuerta Hadamard, tal como se muestra en la Figura 2.9 y define mediante la siguiente ecuación:

$$|s\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle \tag{2.35}$$

Se genera un crecimiento en la amplitud de todos los estados, como se observa en la Figura 2.10. Y tras realizar la superposición de los estados, se realiza un incremento en la

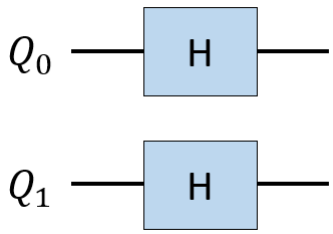


Figura 2.9: Inicialización y superposición de qubits

amplitud en el estado solución mientras que los demás presentan un decremento tal como se muestra en la Figura 2.10.

Al realizar el incremento se aplica la función oráculo, identificando la posible solución y aplicando un cambio de fase, tal como se muestra en la Figura 2.11.

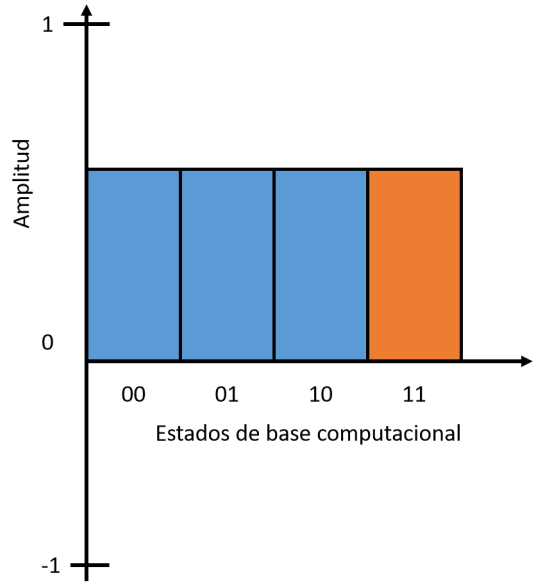


Figura 2.10: Representación gráfica del funcionamiento de la inicialización, superposición y modificación de amplitudes

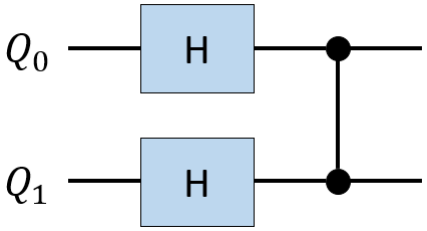


Figura 2.11: Circuito esquemático de aplicación de oráculo

El oráculo realiza un cambio en el valor de la amplitud de positivo a negativo, ejerce un cambio en su trayectoria e identifica la solución mediante la modificación del signo de la amplitud [61] con base en la siguiente ecuación.

$$U_f \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle = \begin{cases} \left|x\right\rangle & \text{si } f(x) \text{ es } 0 \\ -\left|x\right\rangle & \text{si } f(x) \text{ es } 1 \end{cases} \tag{2.36}$$

Por ejemplo, identificar el estado $|11\rangle$ se realiza de la siguiente manera :

$$U_\omega \left|s\right\rangle = U_\omega \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle) \tag{2.37}$$

Por lo que la matriz creada a partir de los valores de los estados cuenta con los siguientes datos:

$$U_\omega = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.38}$$

Su representación en circuito se muestra en la Figura 2.12, y se define la entrada y su modificación a la salida después de aplicarse.

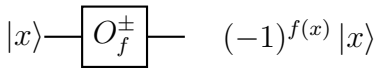


Figura 2.12: Representación en bloque de un oráculo

La aplicación de la función se muestra en la Figura 2.13, donde la posible solución cambia de plano mientras que los estados que no pertenecen a la solución se mantienen en su misma posición.

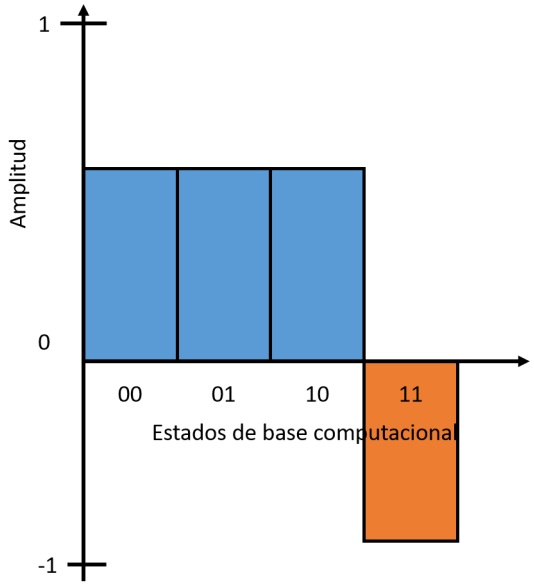


Figura 2.13: Representación de cambio de amplitud utilizando un oráculo

Al identificar la solución y modificar su amplitud se aplica un difusor, que ejerce una reflexión al valor de la solución y amplifica el resultado mientras que los demás estados lo disminuyen [62], como se muestra en la Figura 2.14.

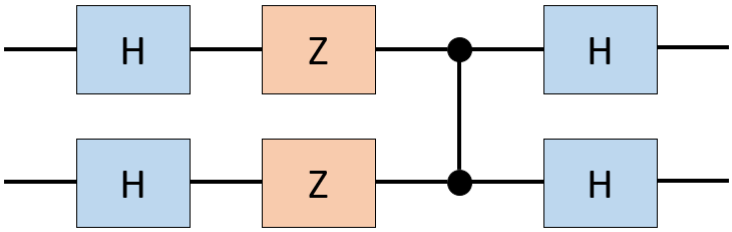


Figura 2.14: Circuito cuántico del difusor

Su funcionamiento se muestra en la Figura 2.15, donde el estado solución presenta una mayor amplitud y los demás una menor, como se muestra en la Figura 2.10.

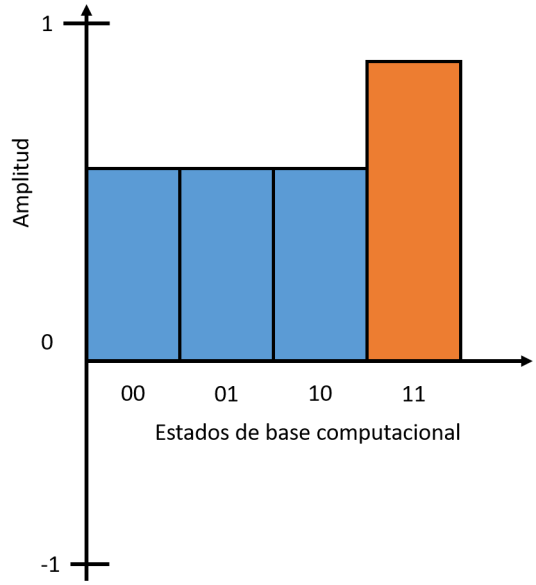


Figura 2.15: Funcionamiento de difusor sobre circuito cuántico

Este trabaja mediante la reflexión del estado para establecer una fase negativa al estado ortogonal de $|s\rangle$ de la siguiente manera:

$$U_s = 2\,|s\rangle\,\langle s| - 1 \tag{2.39}$$

Por lo que es necesario transformar el estado $|s\rangle \rightarrow |0\rangle$ a través de la aplicación de la compuerta Hadamard a cada qubit

$$H^{\otimes N}\,|s\rangle = |0\rangle \tag{2.40}$$

Generando el cambio de fase al estado ortogonal del estado $|0\rangle$ obteniendo lo siguiente:

$$U_0\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle - |11\rangle) \tag{2.41}$$

Así los estados cambian su signo a negativo excepto el estado $|00\rangle$. Finalmente se realiza una ultima transformación $|0\rangle \rightarrow |s\rangle$ donde

$$H^{\otimes N}U_0H^{\otimes N} = U_s \tag{2.42}$$

Para obtener una solución más acertada es necesario realizar la ejecución de este algoritmo [63] reiteradas veces. La representación gráfica del circuito de Grover se muestra en la Figura 2.16.

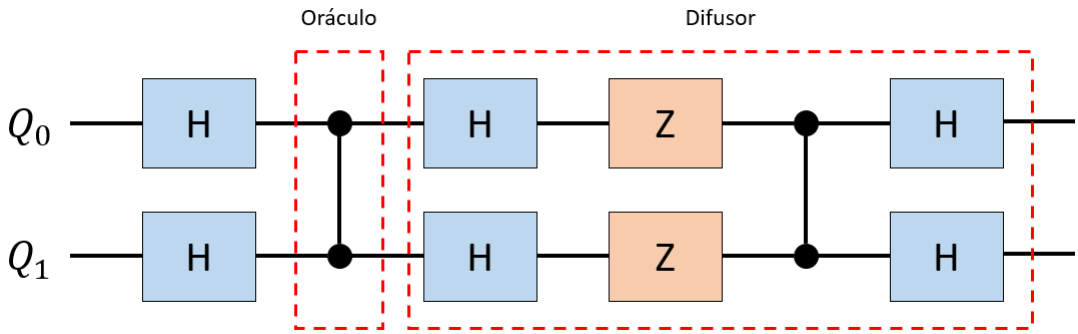


Figura 2.16: Circuito cuántico de Grover

2.5.7. Computadora y simulador cuántico IBMQ

IBM trabaja con cómputo cuántico y tiene una plataforma llamada "IBM Quantum Experience", cuenta con procesadores cuánticos reales y simuladores. Permite ejecutar programas y circuitos cuánticos, algunas de las herramientas [64] con las que cuenta son para la creación, simulación y ejecución de circuitos cuánticos, ofrece un almacenamiento en la nube para alojar los circuitos creados, y posee documentación para su uso. Algunos servicios y herramientas son:

- Simulación:
  Espacio de trabajo para construir los circuitos utilizando el lenguaje de programación Python o mediante la manipulación de bloques. Permite identificar posibles errores antes de exportar el programa a una computadora cuántica real mediante el uso de simuladores.

- Ejecución:
  Apartado para ejecutar los circuitos o programas cuánticos previamente depurados, trabaja de manera remota gracias a su implementación en la nube y cuenta con una cola de ejecución para su ejecución. El tiempo de ejecución puede variar dependiendo de la demanda de los mismos y el tamaño del circuito/algoritmo.

- Creación:
  Es el área de trabajo para la elaboración y desarrollo de nuevos programas mediante código o utilizando el creador de circuitos por bloques.

- Almacenamiento:
  Espacio designado en la nube utilizado para almacenar los programas, circuitos elaborados y resultados obtenidos durante las ejecuciones.

- Ayuda:
  Apartado para consultar la documentación y manuales de uso de cada componente con el que cuenta la plataforma. También cuenta con una área de preguntas frecuentes

y facilita su uso para personas que no conocen el entorno pero cuentan con las bases para elaborar circuitos cuánticos.

IBM cuenta con tres computadoras cuánticas, una de ellas cuenta con 5 qubits y las dos restantes con 16. Además cuenta con un simulador cuántico con capacidad de 32 qubits y permite realizar ejecuciones para identificar errores antes de pasar a una computadora cuántica real. Cuenta con dos dispositivos de 20 qubits, sin embargo solamente pueden ser utilizados por sus socios [65]. Esto puede variar con el paso del tiempo ya que la plataforma se encuentra en constante actualización y desarrollo, incrementando el número de qubits en sus procesadores y creando nuevas compuertas para facilitar su uso y aplicación.

Capítulo 3

Aplicación del algoritmo de Grover para resolver el problema de las *N-Reinas*

El algoritmo híbrido cuántico propuesto busca las posiciones seguras para colocar un N número de piezas en un tablero de tamaño $N \times N$ [66] con base en la siguiente ecuación:

$$R_{ij} = \begin{cases} 1 & \text{si existe una reina en la posición } (i,j) \\ 0 & \text{si no} \end{cases} \in \{1,0\} \tag{3.1}$$

El uso del cómputo cuántico facilita la obtención de las posiciones seguras para colocar a las piezas en el problema de las *N-Reinas*. Los puntos a considerar para lograrlo son: las restricciones establecidas para cada situación, la implementación del algoritmo de Grover y la identificación de las posiciones no amenazadas. Las restricciones con las que se trabajan son similares a las utilizadas en los algoritmos de cómputo tradicional, sin embargo, se considera que la implementación del cómputo híbrido cuántico permite la ejecución de múltiples qubits representando varias posiciones de manera simultánea, es por esto que el algoritmo de Grover identificará que los estados cumplan con las restricciones establecidas [67], las cuales se presentan a continuación:

- Restricción por fila: La suma de los estados que representan a las reinas y se encuentran posicionados de manera horizontal debe de ser de al menos 1, representado de la siguiente manera.

$$\sum_{j,i=0}^{N-1} R_{i+Nj} = 1 \tag{3.2}$$

- Restricción por columna: Similar a la restricción por fila, realiza una sumatoria de todos los estados que colindan de manera vertical y validando que el total sea igual a

1 expresado de la siguiente manera.

$$\sum_{i,j=0}^{N-1} R_{i+Nj} = 1 \tag{3.3}$$

- Restricción por diagonales: Trabaja con dos variables definidas como diagonal positiva y negativa. Realiza una sumatoria para cada instancia y evalúa el número total de piezas por cada diagonal de la siguiente manera:

  - Diagonal positiva:

$$\sum_{j=0}^{(N-1)-i} R_{i+(N+1)j} = 1 \tag{3.4}$$

  - Diagonal negativa:

$$\sum_{j=0}^{i} R_{i+(N-1)j} = 1 \tag{3.5}$$

  Las restricciones facilitan la tarea de encontrar las posiciones seguras y la combinación encontrada se evalúa con el oráculo mediante la siguiente ecuación:

$$U_\omega = 2\left|\omega\right\rangle\left\langle\omega\right| - 1 \tag{3.6}$$

Así, todo estado con valor $U_\omega = 1$ representa las posiciones no seguras mientras que $U_\omega = 0$ representa las seguras. Todas las identificadas como válidas se almacenan en el registro cuántico

$$\left|\varphi\right\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left|s\right\rangle \tag{3.7}$$

Al aplicar las restricciones se generará una gran cantidad de combinaciones y los registros cuánticos almacenarán las combinaciones para su filtrado aplicando el algoritmo de Grover [30] descrito como el total de estados encontrados dividido por $\sqrt{N}$. Como se muestra en la siguiente ecuación:

$$\left|\psi\right\rangle = \frac{\left|0\right\rangle + \left|1\right\rangle + \ldots + \left|V_1\right\rangle + \ldots + \left|V_m\right\rangle + \ldots + \left|N-1\right\rangle}{\sqrt{N}} \tag{3.8}$$

El oráculo modifica del valor a $U_f = (-1)^{f(i)}$, mientras que el difusor realiza el cambio de fase como se muestra en la Ecuación 3.6.

3.1. Diagrama de flujo del algoritmo de Grover para resolver el problema de las N-Reinas

Durante el desarrollo del algoritmo híbrido cuántico, se realizó una implementación basada en la integración del algoritmo de Grover y las restricciones de posicionamiento de las reinas para identificar las zonas seguras y no seguras y colocar las piezas que cumplan con las limitaciones establecidas, además se presentan diagramas que muestran el funcionamiento y procesamiento de datos que se lleva a cabo para determinar las soluciones y combinaciones que cumplen con lo antes descrito.

En la Figura 3.1 se identifica el diagrama central del funcionamiento del algoritmo híbrido cuántico. Inicia el posicionamiento de una reina en el tablero desde la posición $(0, 0)$ (fila 0 y columna 0) y a partir de esto se identifican todas las posiciones seguras por fila, columna y diagonal. En esta paso se evalúan todas las posiciones seguras por fila, columna y diagonal, tal como se muestra en la Figura 3.2.

Al asignar un qubit en cada posición segura se determina, con base en las restricciones utilizadas en los algoritmos de cómputo tradicional, los diferentes estados cuánticos considerados como pseudo-soluciones, después se ejecuta el algoritmo de Grover para su evaluación, como se muestra en los diagramas de las Figuras 3.3 y 3.4.

Y se identifican las soluciones que cumplan con las restricciones, almacenando los estados cuanticos en un registro y extrapolando el resultado representado en un tablero, como se muestra en las Figuras 3.5 y 3.6 respectivamente, logrando así resolver el problema de las *N-Reinas* utilizando un algoritmo híbrido-cuántico.

3.2. Pseudocódigo del algoritmo de Grover para resolver el problema de las N-Reinas

El Algoritmo 4 muestra el pseudocódigo que da solución solución al problema de las *N-Reinas*. Explica la identificación de la posición inicial, las modificaciones que se realizan al momento de que las restricciones por fila, columna y diagonal no se cumplen, la asignación de qubits a cada posición segura registrada a partir del posicionamiento inicial de una reina y la implementación del algoritmo de Grover, parte fundamental en la evaluación de los estados obtenidos derivados de la aplicación de restricciones, al igual que la extrapolación y representación de resultados en un tablero de ajedrez con dimensiones previamente establecidas.

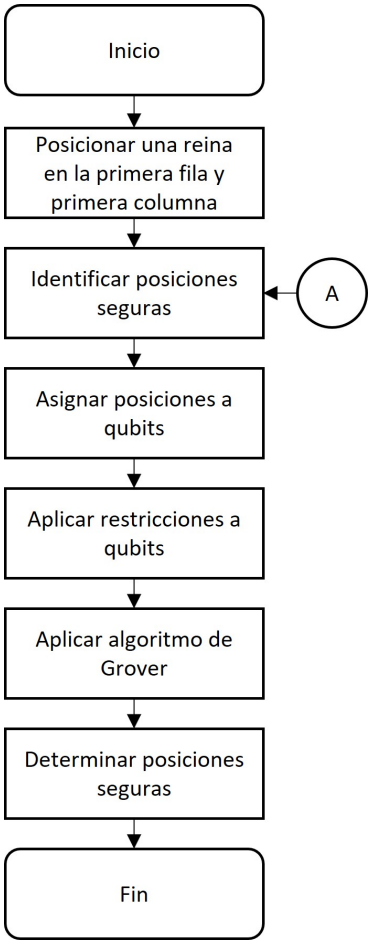


Figura 3.1: Diagrama general de funcionamiento

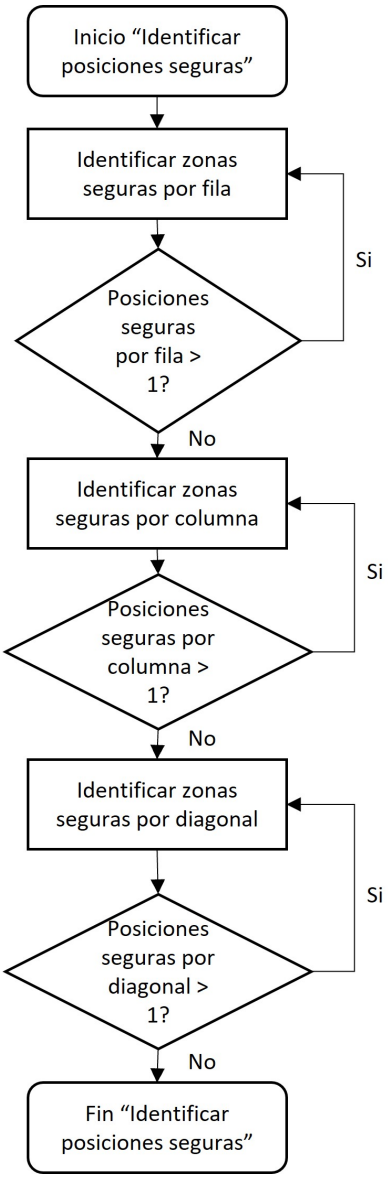


Figura 3.2: Diagrama de identificación de posiciones seguras

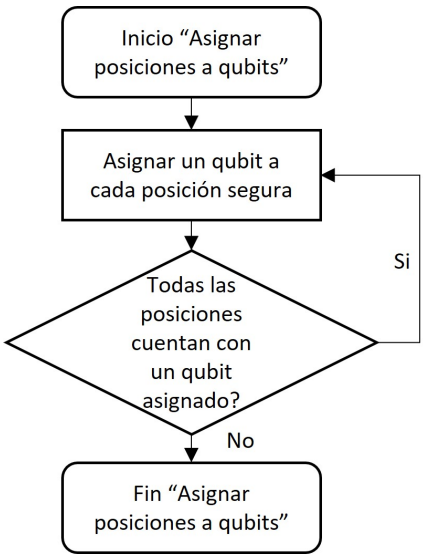


Figura 3.3: Diagrama de asignación de posiciones seguras a qubits

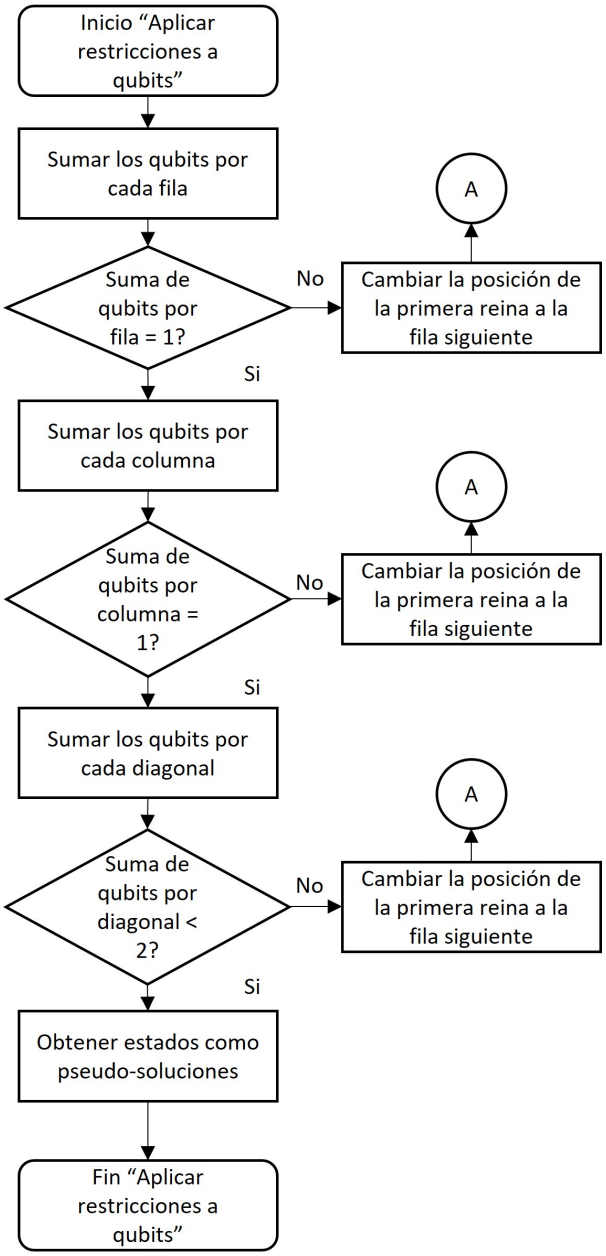


Figura 3.4: Diagrama de aplicación de restricciones a qubits

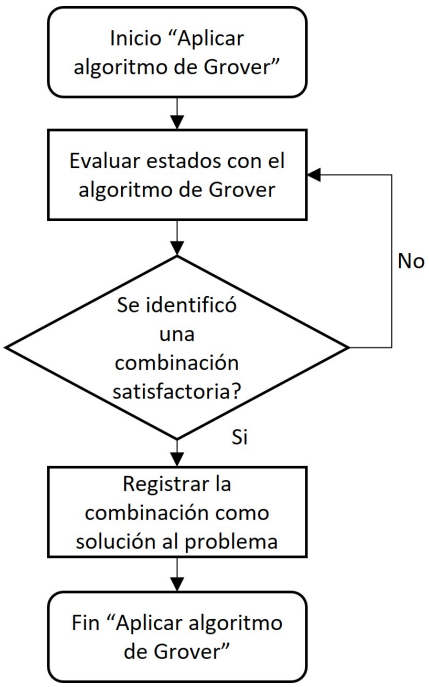


Figura 3.5: Diagrama de de aplicación del algoritmo de Grover

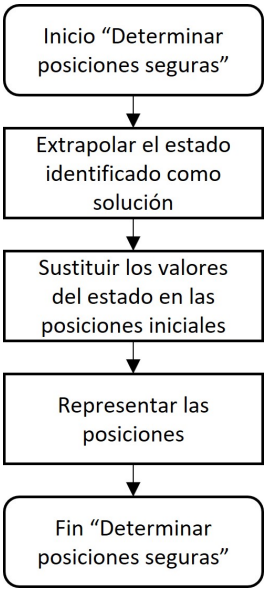


Figura 3.6: Diagrama para representar posiciones seguras

Algoritmo 4: Algoritmo de híbrido cuántico para resolver el problema de las *N-Reinas*

Entrada: $Q \leftarrow$ Número de qubits, $N \leftarrow$ Número de reinas
mientras $i < N$ **hacer**
  Identificación de posición inicial
  si P_i = *evaluada* **entonces**
    $P_i = (i, j)$
  en otro caso
    $P_i = (i + 1, j)$
  fin
  Identificar Z = Posiciones seguras Identificación de posiciones seguras
  si $Z_{fila} \notin Z$ **entonces**
    Identificar posiciones faltantes por fila
  fin
  si $Z_{columna} \notin Z$ **entonces**
    Identificar posiciones faltantes por columna
  fin
  si $Z_{diagonal} \notin Z$ **entonces**
    Identificar posiciones faltantes por diagonal
  fin
  Asignar Q por cada Z Asignación de qubits
  Sumar qubits ubicados en fila
  si Q_{fila} = *1* **entonces**
    Sumar qubits por fila
  en otro caso
    Aumentar $i + 1$ en P_i
  fin
  Aplicación de restricciones
  Sumar qubits ubicados en columna
  si $Q_{columna}$ = *1* **entonces**
    Sumar qubits por diagonal
  en otro caso
    Aumentar $i + 1$ en P_i
  fin
  Sumar qubits ubicados en diagonal
  si $Q_{diagonal} < 2$ **entonces**
    Registrar $|s\rangle$ = pseudo soluciones
    Aumentar $j + 1$ **en** P_i
  en otro caso
    Aumentar $i + 1$ en P_i
  fin
fin
Aplicar algoritmo de Grover a estados en $|s\rangle$
mientras *Estados en* $|s\rangle$ *!= solución satisfactoria* **hacer**
  Ejecutar algoritmo de Grover a estados en $|s\rangle$
  $Sol = |s\rangle$
fin
Identificación de solución
Extrapolar *Sol*
Sustituir valores de *Sol* en posiciones iniciales
Representar posiciones seguras en tablero Representación de solución
Fin del algoritmo

Capítulo 4

Experimentos y Análisis de Resultados

En este capítulo se presentan los algoritmos que dan solución al problema de las *N-Reinas* utilizando una computadora tradicional y un simulador cuántico. Se evaluaron dos algoritmos de búsqueda exhaustiva (Backtracking y Branch and Bound) y uno de restricciones (Programación Lineal), obteniendo la media y la desviación estándar para un total de 17 reinas. En cada uno se describe el procesamiento realizado para obtener las soluciones y combinaciones que satisfacen al problema, identificando la complejidad computacional, las ventajas y desventajas que cada uno presenta comparado con el algoritmo híbrido cuántico propuesto. El alcance está limitado a la evaluación de los algoritmos clásicos de manera secuencial, evitando aceleraciones u optimizaciones como en la programación en paralelo.

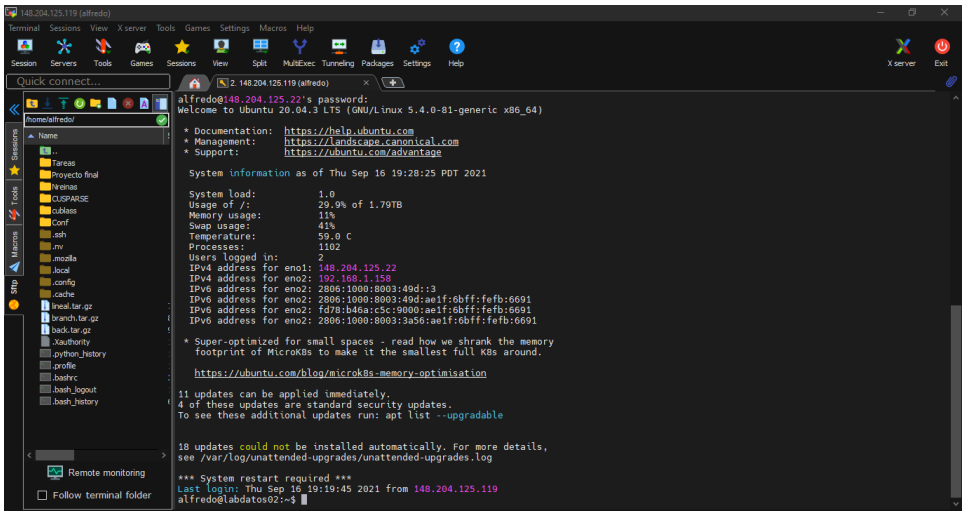


Figura 4.1: Programa para establecer la conexión SSH con el servidor del Centro de Investigación y Desarrollo de Tecnología Digital

Se realizaron las acciones siguientes para realizar la comparación entre los algoritmos:

- Evaluar con diferentes números de reinas en un rango desde 4 hasta 17.
- Ejecutar los algoritmos tradicionales en los servidores del Centro de Investigación y

Desarrollo de Tecnología Digital del Instituto Politécnico Nacional (CITEDI - IPN) que cuentan con un procesador procesador Xeon Gold 5218 @ 2.3 GHz. Una memoria RAM instalada con una capacidad de 512 GB. El sistema operativo instalado es Linux Ubuntu versión 20.04.2 LTS, mediante el archivo MobaXterm para una conexión SSH como se muestra en la Figura 4.1.

- Realizar los experimentos del algoritmo híbrido cuántico en los servidores de IBM Quantum Experience, como se muestra en la Figura 4.2.

- Ejecutar 20 veces las pruebas y obtener la media y la desviación estándar para cada caso.

- Compilar los algoritmos utilizando Python versión 3.8.10.

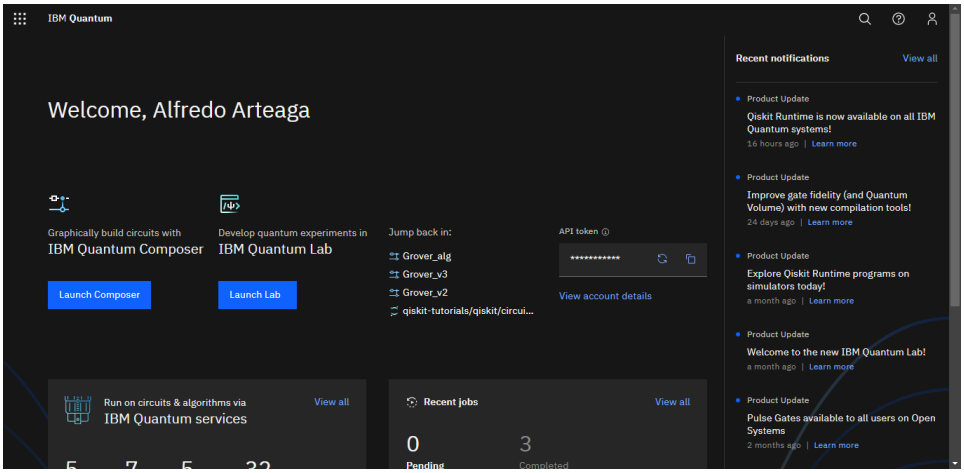


Figura 4.2: Interfaz para el uso de las computadoras cuánticas y simulador de IBM

4.1. Condiciones de experimentación

Para realizar la fase de experimentación se consideró lo siguiente:

- Respaldar la información en documentos de texto.

- Realizar el ejercicio con 4 hasta 17 reinas.

- Obtener el promedio utilizando la ecuación 4.1.

$$\bar{x} = \frac{\sum_{i=1}^{Z} x_i}{Z} \tag{4.1}$$

Donde:

- $\bar{x}$ es la media aritmética.
- $\sum_{i=1}^{Z}$ es la sumatoria de todos los resultados obtenidos producto de las 20 iteraciones.
- x_i es el tiempo de ejecución de cada iteración.
- Z es el número total de iteraciones realizadas.

4.2. Solución al problema de las *N-Reinas* con cómputo tradicional

El problema de las *N-Reinas* se resolvió utilizando diferentes algoritmos como Backtracking, Branch and Bound y Programación Lineal, los pseudocódigos para realizar la programación en Python y los resultados obtenidos producto de las múltiples iteraciones así como el promedio y la desviación estándar para cada caso se describe a continuación.

4.2.1. Caso de estudio 1: Aplicación del algoritmo de fuerza bruta - Backtracking

Este algoritmo realiza evaluaciones bajo el esquema de prueba y error, ejecuta las posibles soluciones, identifica si existe una combinación satisfactoria y en caso contrario, elimina las pseudo soluciones que no cumplen con las restricciones establecidas. Trabaja utilizando la fuerza bruta, aplicando recursividad e iteraciones dentro de un árbol de posibilidades. Se clasifica como un método con poca precisión y con un alto coste computacional [40].

El objetivo es el estudio del comportamiento del algoritmo al resolver el problema para diferentes instancias, variando el tamaño de la entrada mediante el aumento y disminución de reinas al momento de obtener las combinaciones que sean satisfactorias y aplicando las restricciones previamente establecidas. Para resolver el problema es necesario generar un espacio de búsqueda y establecer las limitaciones requeridas para encontrar una solución satisfactoria, tal como se menciona en la Sección 2.2.

Los resultados obtenidos se muestran en la Tabla 4.1. Se observa un crecimiento exponencial del tiempo para un total de 20 iteraciones en cada instancia, al igual que en la media y desviación estándar para cada N cuando $N \geq 4$ y $N \leq 17$.

Tabla 4.1: Tiempo de ejecución promedio y desviación estándar usando Backtracking

Número de reinas	Tiempo de ejecución promedio	Desviación estándar
4	0.00007 s	1.39046E-20
5	0.0002 s	2.78092E-20
6	0.0008 s	1.11237E-19
7	0.003405 s	2.23607E-05
8	0.015385 s	0.000113671
9	0.074975 s	0.000990415
10	0.22796 s	0.017698718
11	1.199305 s	0.021038798
12	6.877155 s	0.069985235
13	42.3444 s	0.744777032
14	278.702635 s	6.718637124
15	1,896.905275 s	35.63734744
16	13,887.82939 s	158.5726299
17	109,411.7678 s	4,167.875636

4.2.2. Caso de estudio 2: Aplicación del algoritmo optimizado de fuerza bruta - Branch and Bound

El objetivo es identificar el comportamiento del algoritmo a la hora de resolver problemas para diferentes casos variando el tamaño de la entrada realizando un aumentando o decremento de reinas con base en las restricciones establecidas y comparando con los demás casos de estudio. Está clasificado como un algoritmo de fuerza bruta, es similar a Backtracking pero cuenta con optimizaciones [68] que permiten la disminución del tiempo de procesamiento. Está diseñado para trabajar con problemas de optimización combinatoria y su funcionamiento consiste en dividir el problema en pequeñas partes omitiendo pseudo soluciones que no cumplan con las restricciones establecidas.

Los resultados obtenidos se muestran en la Tabla 4.2. Se observa un crecimiento exponencial del tiempo para 20 iteraciones, en la media y la desviación estándar para cada N cuando $N \geq 4$ y $N \leq 17$.

Tabla 4.2: Tiempo de ejecución promedio y desviación estándar usando Branch and Bound

Número de reinas	Tiempo de ejecución promedio	Desviación estándar
4	0.0004 s	5.56184E-20
5	0.0005 s	2.22474E-19
6	0.0009 s	1.11237E-19
7	0.00243 s	4.70162E-05
8	0.008565 s	8.75094E-05
9	0.03553 s	0.000231926
10	0.104655 s	0.01866814
11	0.471255 s	0.013809245
12	2.548845 s	0.021257432
13	14.530725 s	0.110244355
14	89.19125 s	0.622579839
15	583.959185 s	3.048646538
16	4,115.949215 s	40.31555712
17	30,531.33199 s	1152.404159

4.2.3. Caso de estudio 3: Aplicación del algoritmo de optimización por restricciones - Programación Lineal

La finalidad es poner a prueba este algoritmo y realizar una comparativa entre los algoritmos optimización por restricciones y los demás casos de estudio, para identificar el comportamiento y complejidad computacional registrada con respecto a los demás. Se busca identificar las limitaciones que pueden afectar su procesamiento al momento de incrementar o decrementar el tamaño de su entrada. Se basa en una metodología de modelado matemático para encontrar soluciones óptimas con base en una variedad limitada de recursos [69].

Los resultados obtenidos se muestran en la Tabla 4.3, se observa un crecimiento exponencial del tiempo para un total de 20 iteraciones en cada instancia, en la media y la desviación estándar para cada N cuando $N \geq 4$ y $N \leq 17$. Durante la experimentación se usó la API de OR-Tools, desarrollada por Google®, una de sus aplicaciones es la solución de problemas combinatorios, como se muestra en el Algoritmo 3.

Tabla 4.3: Tiempo de ejecución promedio y desviación estándar usando Programación Lineal

Número de reinas	Tiempo de ejecución promedio	Desviación estándar
4	0.006335 s	8.12728E-05
5	0.01257 s	0.000126074
6	0.01768 s	0.000119649
7	0.035475 s	0.007626883
8	0.11619 s	0.009265039
9	0.524145 s	0.012362783
10	2.00735 s	0.011765628
11	11.46305 s	0.05936269
12	63.38689 s	0.549429424
13	534.99516 s	1.906083907

4.3. Solución al problema de las *N-Reinas* con cómputo cuántico

Aquí se explicará a detalle la aplicación y funcionamiento del algoritmo de Grover, la forma de solucionar el problema de las *N-Reinas* utilizando el algoritmo híbrido cuántico y se describirán los pseudocódigos desarrollados al igual que los resultados obtenidos derivados de cada circuito cuántico creado.

4.3.1. Caso de estudio 4: Aplicación del algoritmo cuántico para búsqueda no estructurada - Grover

Este caso tiene como finalidad explicar el algoritmo utilizado al momento de resolver el problema de las *N-Reinas*. Se enfoca en la solución de problemas de búsqueda no estructurada, identificando algunas ventajas que presenta el cómputo cuántico al resolver problemas combinacionales. La solución se representa mediante una función f de forma que $f(x_s) = 1$ si y solo si x_s sea una solución, en caso contrario $f(x_s) = 0$. En el cómputo clásico cualquier algoritmo necesita calcular un total de $(x - 1)$ valores de la función $f(x)$, sin embargo, en el cómputo cuántico disminuyen los pasos y presentan una aceleración cuadrática comparada con su contra-parte [70, 71]. Su funcionamiento presenta una identificación de datos utilizando la búsqueda no estructurada y aplicando el algoritmo de Grover. En un conjunto de datos para dos qubits se tienen los siguientes valores:

$$|s\rangle = \begin{bmatrix} 00 \\ 01 \\ 10 \\ 11 \end{bmatrix} \tag{4.2}$$

El primer paso es la creación del circuito cuántico para 2 qubits inicializados con una

superposición a través de la aplicación de la compuerta Hadamard, tal como se muestra en la Figura 4.3, posterior se realiza la implementación del oráculo compuesto por una compuerta Z controlada y trabajando con los dos qubits simultáneamente, como se observa en la Figura 4.4. El valor $|s\rangle = |11\rangle$ ha cambiado su fase por lo que ahora se aplicará el difusor U_s, compuesto por compuertas Hadamard, Z, Z controlada y una segunda aplicación de la compuerta Hadamard, tal como se describe en la Sección 2.5.6. El circuito final se muestra en la Figura 4.5 y la secuencia utilizada se muestra en el Algoritmo 5.

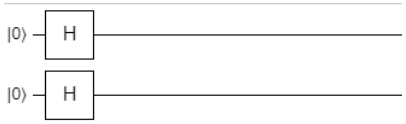


Figura 4.3: Inicialización de qubits con compuertas Hadamard

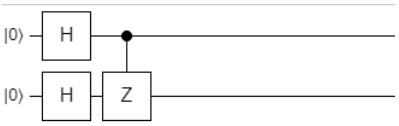


Figura 4.4: Aplicación del oráculo

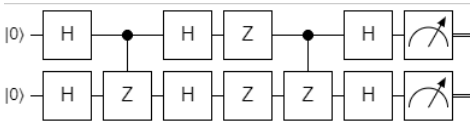


Figura 4.5: Implementación del difusor

Algoritmo 5: Algoritmo de Grover

Entrada: $Q \leftarrow$ Número de qubits, $Rep \leftarrow 0$, $N \leftarrow$Número de reinas, $M \leftarrow$ Número de combinaciones

Inicializar qubit Q_1 y Q_2 con compuertas Hadamard

mientras $Rep < \frac{\pi}{4}\sqrt{\frac{N}{M}}$ **hacer**

- **Aplicar** oráculo con compuerta Z controlada en qubit Q_1 y Q_2
- **Identificar** solución del estado $|s\rangle = |11\rangle$
- **Modificar** la fase con el difusor $U_s = 2\,|s\rangle\,\langle s| - 1$
- **Realizar** medición de qubits empleados
- **Aumentar** $Rep = Rep + 1$

fin

Fin del algoritmo

4.3.2. Caso de estudio 5: Aplicación del algoritmo híbrido cuántico para búsqueda no estructurada - Problema de las *N-Reinas*

En este caso se presenta una integración de las restricciones establecidas en el cómputo tradicional y el algoritmo de Grover para la identificación de búsquedas no estructuradas y probar el funcionamiento, identificando las ventajas, desventajas y limitaciones encontradas al momento de realizar la integración de ambas partes. Los resultados obtenidos se compararán contra los demás casos de estudio, identificando su eficiencia.

Planteamiento del problema de las *N-Reinas* para 4 reinas

Considera los mismos criterios y restricciones que en el cómputo tradicional. Para un total de 4 reinas en un tablero de 4×4 existen 2 combinaciones, una de ellas se muestra en la Figura 4.6.

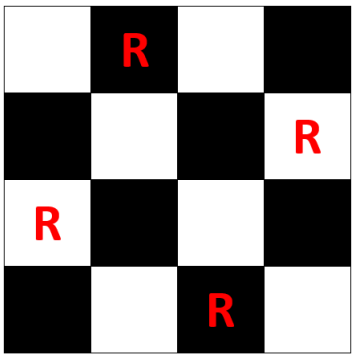


Figura 4.6: Combinación para 4 reinas

Para su solución se aplican las restricciones y se propone una posición que pertenezca a la solución, como se muestra en la Figura 4.7.

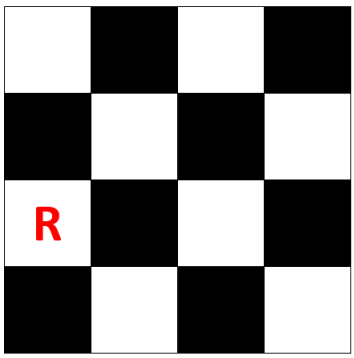


Figura 4.7: Posición segura que pertenece a una solución del problema de las *N-Reinas*

Se identifican las posiciones no amenazadas por la reina y se representan como R_n, donde n es el número de espacios disponibles para colocar una reina, como se muestra en la Figura 4.8.

Se aplican las restricciones de filas, columnas y diagonales de la siguiente manera:

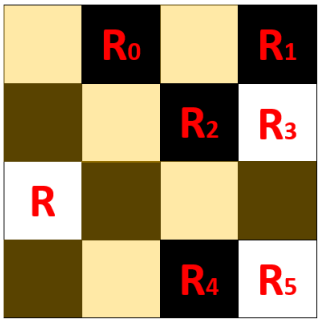


Figura 4.8: Identificación de posibles casillas seguras

- Restricción 1: Una reina por fila.
  - $R_0 + R_1 = 1$
  - $R_2 + R_3 = 1$
  - $R_4 + R_5 = 1$
- Restricción 2: Una reina por columna.
  - $R_0 = 1$
  - $R_2 + R_4 = 1$
  - $R_1 + R_3 + R_5 = 1$
- Restricción 3: Una reina por diagonal.
  - $R_0 + R_2 < 2$
  - $R_1 + R_2 < 2$

Solución al problema para 4 reinas utilizando un simulador de IBM-Quantum

El circuito cuántico que resuelve el problema trabaja con las restricciones previamente definidas. El primer paso es realizar la evaluación del número de reinas por fila y columna.

$$\begin{bmatrix} 0 & 1 & 6 \\ 2 & 3 & 7 \\ 4 & 5 & 8 \\ 2 & 4 & 9 \end{bmatrix} \tag{4.3}$$

La primera y segunda columna contienen los qubits a evaluar, donde se colocará la R_0 y R_1, R_2, R_3, R_4 y R_5 para las posiciones horizontales, R_2 y R_4 para las verticales, en la tercera contiene los qubits para almacenar el resultado, como se muestra en la Figura 4.9.

La suma y verificación de los qubits debe de ser tal que $Reinas = 1$, como se muestra en la Figura 4.10.

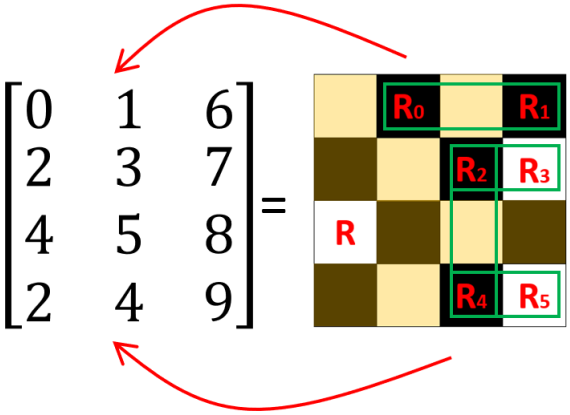


Figura 4.9: Relación de arreglo matricial (filas y columnas) y tablero de ajedrez

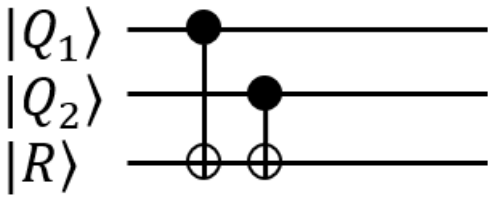


Figura 4.10: Circuito cuántico para realizar la suma de dos qubits.

Después se aplican las restricciones de diagonales positivas y negativas, generando un segundo arreglo, como se muestra a continuación.

$$\begin{bmatrix} 0 & 2 & 10 \\ 1 & 2 & 11 \end{bmatrix} \tag{4.4}$$

La evaluación del número de reinas por diagonal se ejecuta tal como se muestra en el circuito de la Figura 4.11, donde una compuerta Toffoli y una NOT identifican los qubits que se encuentran en el estado 1 determinando las posiciones que no son seguras para colocar a la reina.

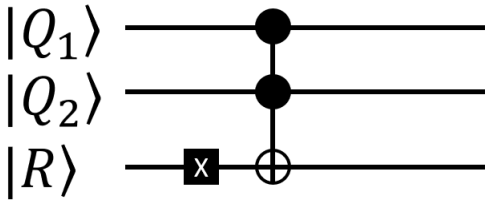


Figura 4.11: Circuito cuántico para validación de número de diagonales.

Las posiciones en dirección diagonal se agrupan dentro del arreglo que se muestra en la Figura 4.12.

Cuando tres o más casillas contienen al menos una reina se realiza la siguiente conside-

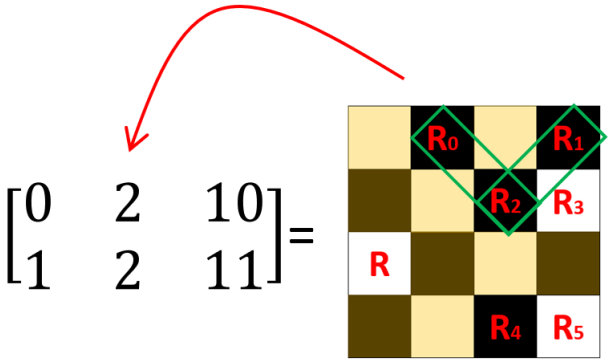


Figura 4.12: Relación de arreglo matricial (diagonales) y tablero de ajedrez

ración, como se muestra en el siguiente arreglo.

$$\begin{bmatrix} 1 & 3 & 5 & 12 & 13 \end{bmatrix} \tag{4.5}$$

Este representa una columna donde existen tres posibles posiciones seguras para colocar a la reina y encontrar una combinación, como se muestra en la Figura 4.13. A diferencia de las otras matrices, en esta se utilizaron dos qubits auxiliares en lugar de uno ya que la plataforma de IBM-Q no permite la suma de 3 elementos en una sola ejecución.

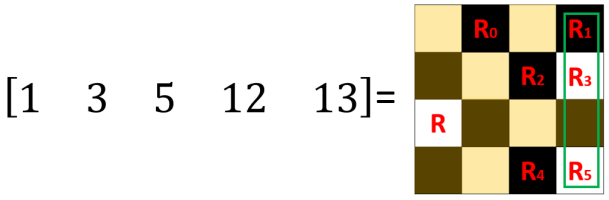


Figura 4.13: Relación de arreglo matricial (3 qubits) y tablero de ajedrez

La operación se realiza con dos qubits iniciales, se suman y el resultado se almacena en uno auxiliar, el cual se vuelve a sumar con un tercero inicial y la solución se guarda en un segundo auxiliar, como se muestra en la Figura 4.14.

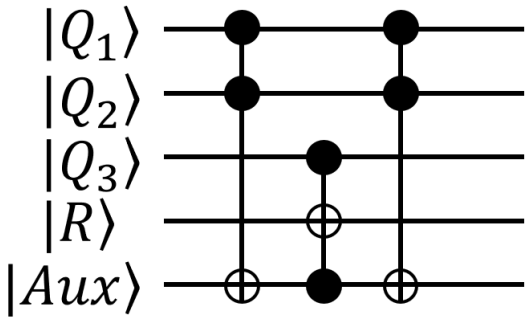


Figura 4.14: Relación de arreglo matricial (3 qubits) y tablero de ajedrez

El circuito que representa las restricciones trabaja con un número de qubits el cual es determinado por la cantidad de auxiliares y los que representan a las posiciones para colocar

las piezas. Para el caso de 4 reinas el circuito cuenta con 26 y se muestra en la Figura 4.15, donde el algoritmo de Grover se ejecuta y facilita la identificación de la solución. Puede ejecutarse varias veces para refinar la búsqueda de la solución. el circuito a utilizar se muestra en la Figura 4.16.

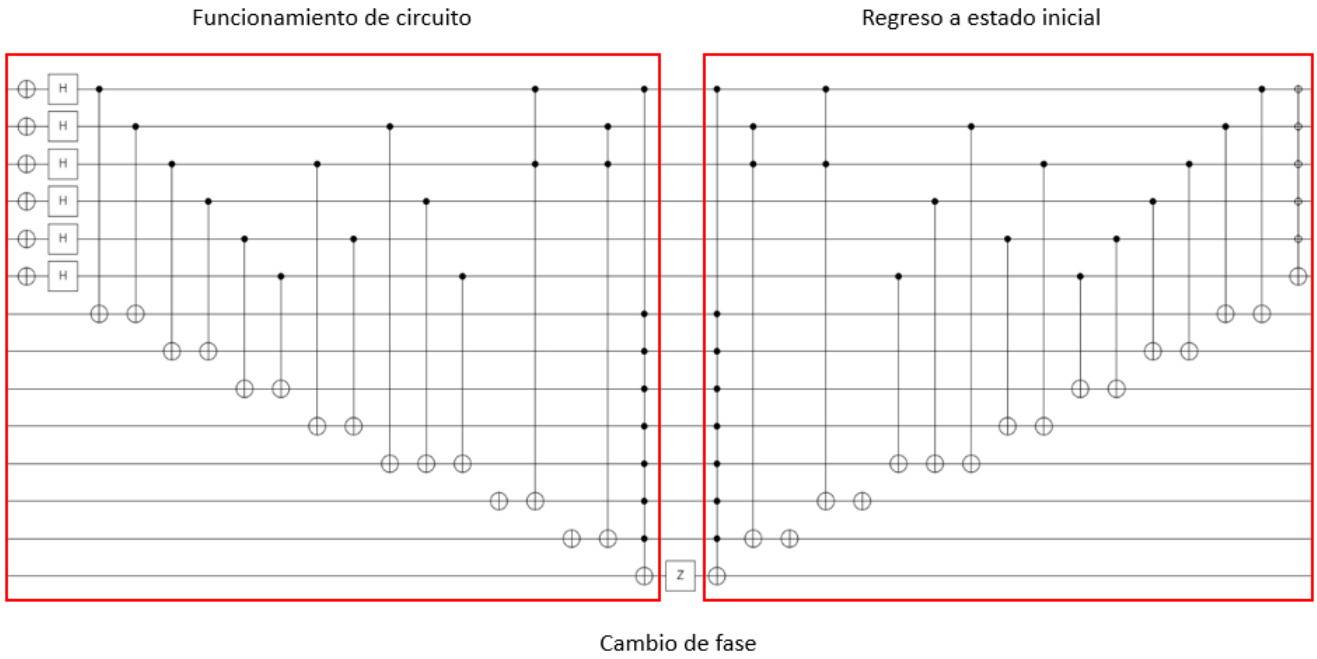


Figura 4.15: Circuito cuántico para la solución del problema de las *N-Reinas*

El algoritmo híbrido cuántico propuesto se ejecutó en el simulador de IBM "*ibmq_qasm_simulator*" y el resultado obtenido se muestra en la Figura 4.17 donde: $R_0 = 1, R_1 = 0, R_2 = 0, R_3 = 1, R_4 = 1$ y $R_5 = 0$.

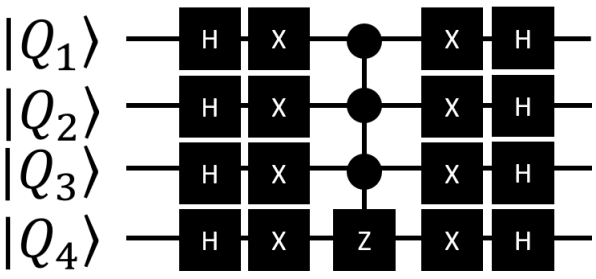


Figura 4.16: Circuito cuántico de Grover para identificar las posiciones seguras para el problema de las *N-Reinas*

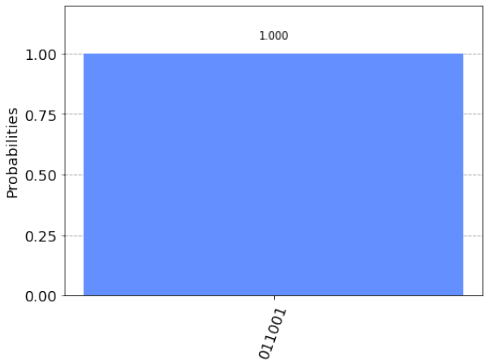


Figura 4.17: Histograma de la solución del problema de las *N-Reinas*

Sustituyendo los valores 0 como posiciones no seguras y 1 como seguras se genera una distribución las reinas que resuelve el problema, como se muestra en la Figura 4.18, mientras que en el Algoritmo 6 describe los pasos a seguir para alcanzar los resultados.

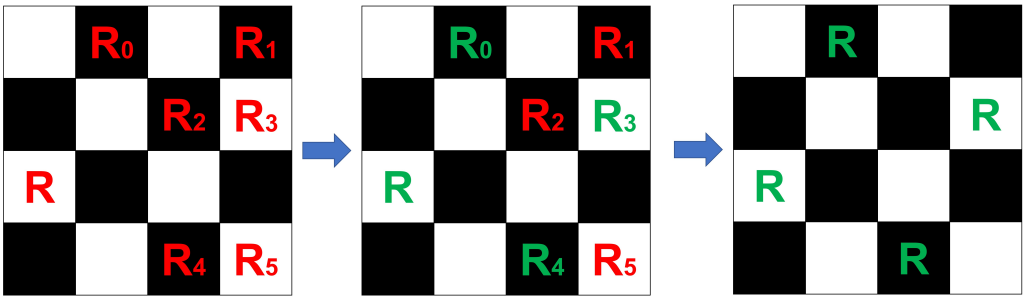


Figura 4.18: Identificación de posiciones partiendo de los resultados obtenidos

Algoritmo 6: Algoritmo de híbrido cuántico

Entrada: $Q \leftarrow$ Número de qubits, $N \leftarrow$ Número de reinas
Salida: $X[i,j] \leftarrow$ Posición segura
Seleccionar una posición al azar en la fila 0
Identificar posiciones seguras y no seguras
Crear restricciones por fila, columna y diagonal
Generar matriz de restricciones
Realizar $\sum_{R=1}^{N} = 1$ para cada restricción entre los qubits $Q_1 ... Q_N$
Verificar $N = 1$ para cada restricción
Aplicar qubits auxiliares para sumas con 3 o mas qubits
Ejecutar algoritmo de Grover
Identificar estado $|s\rangle \leftarrow$ Combinación de qubits
Extrapolar valores obtenidos $\rightarrow X[i,j]$
Fin del algoritmo

Planteamiento del problema de las *N-Reinas* para 5 reinas

El objetivo es identificar las posiciones seguras y no seguras, tal como se muestra en la Figura 4.19.

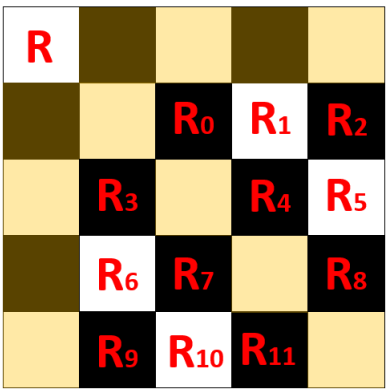


Figura 4.19: Identificación de espacios para colocar reinas en un tablero de 5×5

Para su solución se aplican las restricciones y se propone una posición que pertenezca a la solución. Las restricciones definidas para este caso son las siguientes:

- Restricciones de filas:

$$\begin{aligned} Q_0 + Q_1 + Q_2 &= 1 \\ Q_3 + Q_4 + Q_5 &= 1 \\ Q_6 + Q_7 + Q_8 &= 1 \\ Q_9 + Q_{10} + Q_{11} &= 1 \end{aligned} \tag{4.6}$$

- Restricciones de columnas:

$$\begin{aligned} Q_3 + Q_6 + Q_9 &= 1 \\ Q_0 + Q_7 + Q_{10} &= 1 \\ Q_1 + Q_4 + Q_{11} &= 1 \\ Q_2 + Q_5 + Q_8 &= 1 \end{aligned} \tag{4.7}$$

- Restricciones de diagonales positivas:

$$\begin{aligned} Q_3 + Q_0 &< 2 \\ Q_6 + Q_1 &< 2 \\ Q_9 + Q_7 + Q_4 + Q_2 &< 2 \\ Q_{10} + Q_5 &< 2 \\ Q_{11} + Q_8 &< 2 \end{aligned} \tag{4.8}$$

- Restricciones de diagonales negativas:

$$\begin{aligned} Q_6 + Q_{10} &< 2 \\ Q_3 + Q_7 + Q_{11} &< 2 \\ Q_0 + Q_4 + Q_8 &< 2 \\ Q_1 + Q_5 &< 2 \end{aligned} \tag{4.9}$$

Solución al problema de las para 5 reinas utilizando un simulador de IBM-Quantum

El circuito se implementó en el simulador "*simulator_mps*" debido a que el simulador "*ibmq_qasm_simulator*" no cuenta con el número necesario de qubits. Las restricciones se clasificaron dependiendo del número de qubits con los que se trabajó, verificando el número

de reinas por fila y columna. Para 3 qubits se tiene el siguiente arreglo.

$$\begin{bmatrix} 0 & 1 & 2 & 12 & 13 \\ 3 & 4 & 5 & 14 & 15 \\ 6 & 7 & 8 & 16 & 17 \\ 9 & 10 & 11 & 18 & 19 \\ 3 & 6 & 9 & 20 & 21 \\ 0 & 7 & 10 & 22 & 23 \\ 1 & 4 & 11 & 24 & 25 \\ 2 & 5 & 8 & 26 & 27 \end{bmatrix} \tag{4.10}$$

Las tres primeras columnas de izquierda a derecha representan las posiciones de las reinas y las ultimas dos los qubits auxiliares. El siguiente arreglo representa todas las restricciones para las diagonales con 2 qubits, tal como se muestra a continuación.

$$\begin{bmatrix} 3 & 0 & 28 \\ 6 & 1 & 29 \\ 10 & 5 & 30 \\ 11 & 8 & 31 \\ 6 & 10 & 32 \\ 1 & 5 & 33 \end{bmatrix} \tag{4.11}$$

La evaluación de las diagonales con 3 qubits se realizó aplicando un circuito equivalente a la compuerta Toffoli, como se muestra en la Figura 4.20.

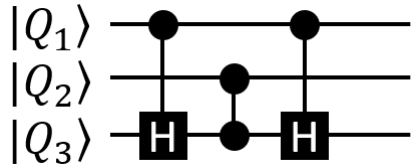


Figura 4.20: Circuito equivalente a la compuerta Toffoli para 3 qubits

Esta equivalencia trabaja con las restricciones para las diagonales con 3 qubits y se aplica al arreglo donde las 3 primeras columnas de izquierda a derecha representan a las posiciones y la última almacena el resultado.

$$\begin{bmatrix} 3 & 7 & 11 & 34 \\ 0 & 4 & 8 & 35 \end{bmatrix} \tag{4.12}$$

Las diagonales con 4 qubits utilizaron el mismo método que con 3 y el arreglo generado es el siguiente:

$$\begin{bmatrix} 9 & 7 & 4 & 2 & 36 \end{bmatrix} \tag{4.13}$$

Posterior se ejecuta el algoritmo de Grover para identificar la solución al problema. El resultado obtenido se muestra en la Figura 4.21.

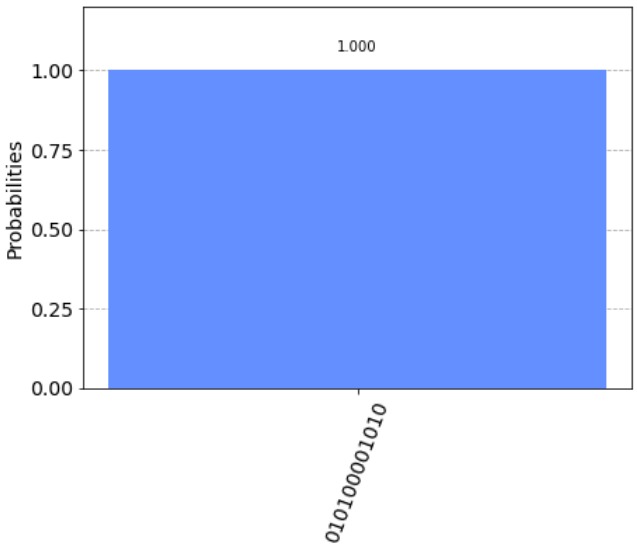


Figura 4.21: Combinación de qubits que soluciona el problema para 5 reinas.

Los valores que se muestran en el histograma se sustituyen en las posiciones identificadas en la Figura 4.19 obteniendo la siguiente relación: $R_0 = 0$, $R_1 = 1$, $R_2 = 0$, $R_3 = 1$, $R_4 = 0$, $R_5 = 0$. $R_6 = 0$, $R_7 = 0$, $R_8 = 1$, $R_9 = 0$, $R_{10} = 1$ y $R_{11} = 0$. La representación gráfica de la solución se muestra en la Figura 4.22

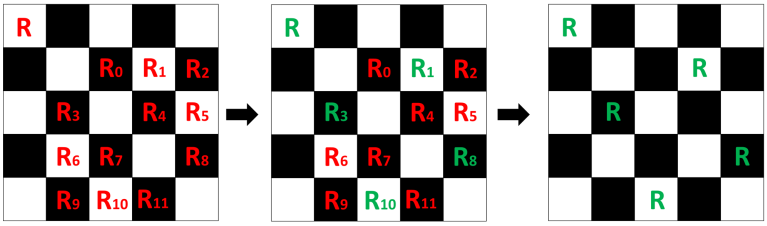


Figura 4.22: Sustitución de valores del histograma en las posibles casillas para colocar a las reinas

4.4. Análisis de resultados y comparaciones

En esta sección se compararán los resultados, tiempos de procesamiento y complejidad computacional de acuerdo a los algoritmos ejecutados y la estructura con la que cuentan.

Los algoritmos de Backtracking, Branch and Bound y Programación lineal se ejecutaron 20 veces, se calculó el valor promedio para cada número de reinas y los resultados obtenidos se presentan en la Tabla 4.4 (los resultados mostrados como "- -"no se obtuvieron debido a la gran cantidad de tiempo de procesamiento necesario para obtener todas las combinaciones).

Tabla 4.4: Tiempo de ejecución promedio con Backtracking, Branch and Bound y Programación lineal

Número de reinas	Tiempo Backtracking	Tiempo Branch and Bound	Tiempo P. Lineal
4	0.00007 s	0.0004 s	0.006335 s
5	0.0002 s	0.0005 s	0.01257 s
6	0.0008 s	0.0009 s	0.01768 s
7	0.003405 s	0.00243 s	0.035475 s
8	0.015385 s	0.008565 s	0.11619 s
9	0.074975 s	0.03553 s	0.524145 s
10	0.22796 s	0.104655 s	2.00735 s
11	1.199305 s	0.471255 s	11.46305 s
12	6.877155 s	2.548845 s	63.38689 s
13	42.3444 s	14.530725 s	534.99516 s
14	278.702635 s	89.19125 s	- -
15	1,896.905275 s	583.959185 s	- -
16	13,887.82939 s	4,115.949215 s	- -
17	109,411.7678 s	30,531.33199 s	- -

Con base en estas mediciones se determinó que el algoritmo de Branch and Bound es el que mejor despeño tiene al momento de calcular las combinaciones para cada número de reinas, sin embargo, únicamente se consideran los algoritmos de cómputo tradicional. Para comparar estos resultados y los obtenidos con el algoritmo híbrido cuántico se optó por determinar la complejidad computacional de cada uno e identificar su eficiencia. La complejidad computacional de estos algoritmos se estableció con base en la estructura de cada código, por lo que Backtracking cuenta con una complejidad $O(N^2+2^N(N+2logN))$, Branch and Bound se definió como $O(N^2+3N+NlogN+2^N)$ y Programación lineal como $O(N!)$ mientras que híbrido cuántico se presenta una complejidad de $O(\sqrt{N})$. La complejidad, al ser notación asintótica, se simplificó para trabajar únicamente con el término más significativo, por lo que las complejidades se evaluarán de la siguiente manera:

$$\begin{aligned} \text{Backtracking} &= O(2^N) \\ \text{Branch and Bound} &= O(2^N) \\ \text{Programación lineal} &= O(N!) \\ \text{Híbrido cuántico} &= O(\sqrt{N}) \end{aligned} \tag{4.14}$$

Reemplazando el valor N por el diferente número de reinas se registran los resultados que se muestran en la Tabla 4.5:

Tabla 4.5: Complejidad computacional de los algoritmos Backtracking, Branch and Bound, Programación lineal e Híbrido cuántico

Número de reinas	Complejidad Backtracking	Complejidad Branch and Bound	Complejidad P. Lineal	Complejidad Hib. Cuántico
4	16	16	24	2
5	32	32	120	2.23
6	64	64	720	2.44
7	128	128	5,040	2.64
8	256	256	40,320	2.82
9	512	512	362,880	3
10	1,024	1,024	3,628,800	3.16
11	2,048	2,048	39,916,800	3.31
12	4,096	4,096	479,001,600	3.46
13	8,192	8,192	6,227,020,800	3.6
14	16,384	16,384	87,178,291,200	3.74
15	32,768	32,768	1.30767E+12	3.87
16	65,536	65,536	2.09228E+13	4
17	131,072	131,072	3.55687E+14	4.12

Con base en los registros de complejidad obtenidos se determinó que los resultados que presenta el algoritmo híbrido cuántico son mucho menores que los logrados con los tres algoritmos clásicos ya que el código cuenta con instrucciones sencillas y por el uso de los qubits y las compuertas cuánticas no es necesario un procesamiento que demande un gran costo computacional, ya que tiene como base el algoritmo de Grover que trabaja con datos de manera paralela y en conjunto con el Oráculo y el Difusor de Grover, facilita la tarea de la identificación de posiciones seguras para colocar las piezas en el tablero, disminuyendo la complejidad tal como se muestra en la Tabla 4.6 con base en la siguiente formula.

$$\%\text{ de mejora} = \frac{\text{Complejidad alg. clásico} - \text{Complejidad alg. hib. cuántico}}{\text{Complejidad alg. hib. cuántico}} \times 100 \tag{4.15}$$

En la Tabla 4.6 se compara la complejidad computacional del algoritmo híbrido cuántico contra los algoritmos de Branch and Bound, Backtracking y Programación Lineal. Se observa que el porcentaje de mejora comienza a partir de un 700 % e incrementa de acuerdo al

número de reinas a evaluar, determinando que el uso de este algoritmo disminuye el tiempo de procesamiento necesario al momento de solucionar este problema.

Tabla 4.6: Porcentaje de mejora del algoritmo híbrido cuántico contra Backtracking, Branch and Bound y Prog. lineal

Número de reinas	Hib. cuántico vs Backtracking	Hib. cuántico vs Branch and Bound	Hib. cuántico vs P. lineal
4	700.00 %	700.00 %	1,100.00 %
5	1,331.08 %	1,331.08 %	5,266.56 %
6	2,512.79 %	2,512.79 %	2.93E+04 %
7	4,737.95 %	4,737.95 %	1.90E+05 %
8	8,950.97 %	8,950.97 %	1.43E+06 %
9	16,966.67 %	16,966.67 %	1.21E+07 %
10	32,281.72 %	32,281.72 %	1.15E+08 %
11	61,649.52 %	61,649.52 %	1.20E+09 %
12	118,141.34 %	118,141.34 %	1.38E+10 %
13	227,105.20 %	227,105.20 %	1.73E+11 %
14	437,780.82 %	437,780.82 %	2.33E+12 %
15	845,966.12 %	845,966.12 %	3.38E+13 %
16	1.64E+06 %	1.64E+06 %	5.23E+14 %
17	3.18E+06 %	3.18E+06 %	8.63E+15 %

Capítulo 5

Conclusiones y trabajo futuro

La propuesta presentada en este documento se ejecutó en los simuladores cuánticos de IBM, tiene la capacidad de resolver el problema y de obtener las posiciones consideradas "seguras". El objetivo principal de esta investigación se cumplió y se mantiene en fase experimental, ya que los servidores están en una constante actualización, desarrollo e innovación, por lo que la estructura y código utilizado puede mejorar a futuro.

Los circuitos se desarrollaron con base en investigaciones previas presentadas en la Sección 1.1 y tienen la capacidad de identificar las soluciones para 4 y 5 reinas, sin embargo presentan variaciones en las implementaciones aplicadas para sustituir herramientas que aún no se encuentran desarrolladas en los servidores, como la compuerta Toffoli. Otro obstáculo identificado es el uso de diferentes simuladores, los cuales cuentan con un número limitado de qubits y no todas las herramientas o funciones se encuentran presentes.

Los experimentos realizados fueron seleccionados para demostrar la capacidad y funcionalidad al aplicar el algoritmo híbrido cuántico, comparando sus resultados con los obtenidos con algoritmos tradicionales de fuerza bruta y de optimización combinatoria en cómputo clásico. El algoritmo de Branch and Bound presentó el menor tiempo de procesamiento sin embargo, su complejidad es elevada comparada con el híbrido cuántico, sin embargo, la complejidad computacional obtenida del algoritmo híbrido cuántico resulta ser mucho menor al de Branch and Bound, lo cual se traduce a un menor tiempo de procesamiento teórico, cubriendo la segunda pregunta de investigación. Se encontraron algunas limitaciones en el cómputo tradicional, lo cual da respuesta a la primera pregunta de investigación y estas son: la programación secuencial observada en los algoritmos de fuerza bruta, el incremento del uso de la memoria con base en la estructura del programa, la cantidad de ciclos y condicionales utilizadas en el algoritmo de optimización combinatoria y el tiempo de ejecución para resolver problemas NP-Completo.

Por otra parte, el algoritmo híbrido cuántico cuenta con la capacidad de resolver problemas que trabajan con un tiempo polinomial, disminuyó la complejidad computacional y demostró que el tiempo de procesamiento es mucho menor al obtener el resultado, permitiendo realizar búsquedas exhaustivas en un espacio de posibles soluciones, respondiendo

así la tercera pregunta de investigación. Algunas limitaciones identificadas son: el número de qubits a utilizar por cada número de reinas, algunos simuladores no cuentan con un número suficiente de qubits para ejecutar los circuitos, la capacidad que las computadoras y simuladores cuánticos tienen al utilizar las compuertas, es por esto que se desarrollaron circuitos equivalentes que cumplan con la misma función, otro obstáculo identificado es el uso limitado de las computadoras cuánticas; durante el proceso de experimentación estas fueron inhabilitadas temporalmente con motivo de actualización e implementación de mejoras, disminuyendo el número de qubits y de dispositivos disponibles, direccionando esta fase a una ejecución en simuladores.

5.1. Trabajo futuro

Algunas áreas de oportunidad identificadas en esta investigación para ampliar, profundizar y desarrollar en este trabajo, son:

- Mejoras al circuito desarrollado automatizando la búsqueda e identificación de todas las soluciones al problema.
- Generalización de restricciones para su ejecución en todos los casos, trabajando con un número de qubits para un número variable de reinas.
- Creación y desarrollo de funciones que simulen el uso de compuertas equivalente para un N número de qubits, mitigando sus limitaciones a un máximo de 2.
- Mejorar la ejecución, migrando de simuladores a computadoras que cuenten con el número necesario de qubits.
- Reducción y simplificación de los circuitos creados en este documento, para disminuir el número de qubits a utilizar y el tamaño del circuito a ejecutar.

Referencias

[1] V. Jain y J. S. Prasad. Solving n-queen problem using genetic algorithm by advance mutation operator. *International Journal of Electrical and Computer Engineering*, 2018.

[2] M. Schuld y F. Petruccione. Machine learning with quantum computers. *Springer*, 2021.

[3] B. Li R. Roy y D. Tiwari T. Patel, A. Potharaju. Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications. 2020.

[4] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 1997.

[5] D. Deutsch y R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. Lond. A*, 439, 1992.

[6] L. Grover. Fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, 1996.

[7] I. Vardi y P. Zimmermann I. Rivin. The n -queens problem. *The American Mathematical Monthly*, 1994.

[8] D. Sacaluga. An alternative algorithm for the n–queens puzzle. *Recreational Mathematics Magazine*, 2021.

[9] M. Engelhardt. A group-based search for solutions of the n-queens problem. *Discrete Mathematics*, 2007.

[10] C. Letavec y J. Ruggiero. The n-queens problem. *Informs Transactions on Education*, 2002.

[11] D. Davcev y V. Trajkovik B. R. Stojkoska. N-queens-based algorithm for moving object detection in distributed wireless sensor networks. *Journal of Computing and Information Technology*, 2008.

[12] S. W. Yang C. N. Wang and C. M. Liu y T. Chiang. A hierarchical decimation lattice based on n-queen with an application for motion estimation. *Signal Processing Letters, IEEE*, 2003.

[13] R. Sosic y J. Gu. A polynomial time algorithm for the n-queens problem. *ACM SIGART Bulletin*, 1996.

[14] M. Tanik y Z. Aliyazicioglu C. Erbas. Linear congruence equations for the solutions of the n-queens problem. *Inf. Process. Lett.*, 1992.

[15] C. Liu y T. Chiang C. Wang, S. Yang. A hierarchical decimation lattice based on n-queen with an application for motion estimation. *IEEE Signal Processing Letters*, 2003.

[16] J. L. Mason. Validation of context preserving thread-level speculative execution using n-queens: Comparison of non-cpse and cpse-enabled applications. In *2017 18th International Workshop on Microprocessor and SOC Test and Verification (MTV)*, 2017.

[17] E. Osaghae. Solution to n-queens problem: Heuristic approach. *Transactions on Machine Learning and Artificial Intelligence*, 2021.

[18] S. Naim y A. Boraik A. Al-Gburi. Hybridization of bat and genetic algorithm to solve n-queens problem. *Bulletin of Electrical Engineering and Informatics*, 2018.

[19] A. Kamran K. A. Sani y A. Hussain A. Ahmed, A. Shah. Particle swarm optimization for n-queens problem. *Journal of Advanced Computer Science Technology*, 2012.

[20] Y. Wang y H. Guo J. Cao, Z. Chen. Parallel implementations of candidate solution evaluation algorithm for n-queens problem. *Complexity*, 2021.

[21] W. Yuxin y G. He C. Jianli, C. Zhikui. Parallel genetic algorithm for n-queens problem based on message passing interface-compute unified device architecture. *Computational Intelligence*, 2020.

[22] D. Janssen y A. W. C. Liew. Acceleration of genetic algorithm on gpu cuda platform. 2019.

[23] Y. Ruriko. Linear algebra and its applications with r. *Chapman and Hall/CRC*, 2021.

[24] M. Al-Rudaini. N-queens problem solving using linear programming in gnu linear programming kit (glpk), 2016.

[25] J. Ulson y A. Souza I. Silva. Development of neurofuzzy architecture for solving the n-queens problem. *International Journal of General Systems*, 2005.

[26] M. Waqas y A. Bhatti. Optimization of n+1 queens problem using discrete neural network. *Neural Network World*, 2017.

[27] Y. Takenakay S. Nishikawa N. Funabiki. A maximum neural network approach for n-queen problems. *Biol Cybern*, 1997.

[28] A. J. Lakshmi y V. Muthuswamy. A predictive context aware collaborative offloading framework for compute-intensive applications. *Journal of Intelligent & Fuzzy Systems*, 2020.

[29] V. Baugh y S. Allehaibi S. Güldal. N-queens solving algorithm by sets and backtracking. 2016.

[30] F. Souza y F. Mello. N-queens problem resolution using the quantum computing model. *IEEE Latin America Transactions*, 2017.

[31] H. Talbi y M. Batouche A. Draa, S. Meshoul. A quantum-inspired differential evolution algorithm for solving the n-queens problem. *Int. Arab J. Inf. Technol.*, 2010.

[32] B. Korte y J. Vygen. Combinatorial optimization: Theory and algorithms. *Springer Berlin Heidelberg*, 2008.

[33] J. Bell y B. Stevens. A survey of known results and research areas for n-queens. *Discrete Mathematics*, 309:1–31, January 2009.

[34] R. Back y J. von Wright. *The N-Queens Problem*. 1998.

[35] P. Chanda y P. Pathak S. Mukherjee, S. Datta. Comparative study of different algorithms to solve n queens problem. *International Journal in Foundations of Computer Science Technology*, 2015.

[36] B. Al-Khateeb y W. Tareq. Solving 8-queens problem by using genetic algorithms, simulated annealing, and randomization method. 2013.

[37] S. Kurgalin y S. Borzunov. Concise guide to quantum computing: Algorithms, exercises, and implementations. 2021.

[38] B. Nadel. Representation selection for constraint satisfaction: A case study using n-queens. *IEEE Expert*, 5:16–23, June 1990.

[39] D. Kreher y D. Stinson. Backtracking algorithms. 2020.

[40] H. Priestley y M. Ward. A multipurpose backtracking algorithm. *Journal of Symbolic Computation*, 1994.

[41] G. Kondrak y P. van Beek. A theoretical evaluation of selected backtracking algorithms. *Artificial Intelligence*, 1995.

[42] D. Whitaker. Wiley statsref: Statistics reference online. *John Wiley & Sons, Ltd*, 2014.

[43] P. Narendra y K. Fukunaga W. Koontz. A branch and bound clustering algorithm. *Computers, IEEE Transactions on*, 1975.

[44] H. Stone y J. Stone. Efficient search techniques — an empirical study of the n-queens problem. *IBM Journal of Research and Development*, 1987.

[45] A. J. Joseph. Health, safety, and environmental data analysis - a business approach. *CRC Press*, 1997.

[46] P. Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 1980.

[47] J. Atik y V. Jeutner. Quantum computing and computational law. *Law, Innovation and Technology*, 2021.

[48] H. Ritsch y W. Lechner V. Torggler, P. Aumann. A quantum n-queens solver. *Quantum*, 2018.

[49] T. Mehmet. *Future Networks, Services and Management: Underlay and Overlay, Edge, Applications, Slicing, Cloud, Space, AI/ML, and Quantum Computing*. Springer, 2021.

[50] D. McMahon. *Quantum Computing Explained*. Wiley-Interscience : IEEE Computer Society, 2008.

[51] M. S. Zubairy. Quantum mechanics for beginners - with applications to quantum communication and quantum computing. *Oxford University Press*, 2020.

[52] C. R. Wie. Two-qubit bloch sphere. *Physics*, 2020.

[53] A. Perry R. Sun y J. Turner C. Hughes, J. Isaacson. *Quantum Gates*. 2021.

[54] Y. Yu. Advancements in applications of quantum entanglement. *Journal of Physics: Conference Series*, 2021.

[55] B. Kommadi. Quantum computing solutions: Solving real-world problems using quantum computing and algorithms. *Apress*, 2021.

[56] M. Umezaki Y. Ota y M. Nakahara Y. Tanaka, T. Ichikawa. Quantum oracles in terms of universal gate set. *International Journal of Quantum Information*, 2010.

[57] S. Kak. Measurement complexity and oracle quantum computing. *NeuroQuantology*, 2014.

[58] R. Kabil y M. Bennai Z. Sakhi, A. Tragha. Grover algorithm applied to four qubits system. *Computer and Information Science*, 2011.

[59] K-C. Chen T-W. Huang M-C. Hsu N-P. Cao B. Zeng S-G. Tan y C-R. Chang C-G. Cho, C-Y. Chen. Quantum computation: Algorithms and applications. *Chinese Journal of Physics*, 2021.

[60] M. A. Nielsen e I. Chuang. Quantum computation and quantum information. *American Journal of Physics*, 2002.

[61] Y. Wang y M. Perkowski. Improved complexity of quantum oracles for ternary grover algorithm for graph coloring. *Proceedings of The International Symposium on Multiple-Valued Logic*, 2011.

[62] X. Li y H. Huang D. Li. Phase condition for the grover algorithm. *Theoretical and Mathematical Physics*, 2005.

[63] R. LaPierre. Introduction to quantum computing. *Springer*, 2021.

[64] A. Santos. The ibm quantum computer and the ibm quantum experience. *Revista Brasileira de Ensino de Física*, 2016.

[65] K. Ohshiro y A. Mandviwalla B. Ji. Implementing grover's algorithm on the ibm quantum computers. 2018.

[66] W. Drabent. On correctness and completeness of an n queens program. *Theory and Practice of Logic Programming*, 2021.

[67] C. Echevaría y L. Steinberg H. Manzano. Quantum algorithm for n-queens problem. 2021.

[68] R. Martí y G. Reinelt. The linear ordering problem: Exact and heuristic methods in combinatorial optimization. *Springer-Verlag Berlin Heidelberg*, 2011.

[69] K. J. Vinod y K. Atul S. Vikrant. An introduction to optimization techniques. *Chapman and Hall/CRC*, 2021.

[70] P. Salas. Noise effect on grover algorithm. *The European Physical Journal D*, 2008.

[71] G. Long. Grover algorithm with zero theoretical failure rate. *Physical Review A*, 2001.