



## MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS



VNIVERSITAT  
DE VALÈNCIA

### TRABAJO DE FIN DE MÁSTER

Contribuciones a la computación cuántica adiabática en *hardware* de átomos neutros

**AUTOR:**

Aitor Arribas Hernández

**TUTORES:**

José David Martín Guerrero  
Yolanda Vives Gilabert



Mayo, 2025



## Agradecimientos

Antes de empezar, me gustaría reservar este espacio para agradecer a mis tutores, José y Yolanda por el apoyo y la ayuda que me han ofrecido a lo largo de todo el curso. Asimismo, me gustaría agradecer a mis compañeros de laboratorio, en especial a Carlos y Josele por haberme acogido y haber acompañado estos meses. A mis amigos y familia por estar siempre ahí.

## Resumen

Este trabajo investiga la resolución del problema del conjunto independiente máximo ponderado (*Maximum Weighted Independent Set*, MWIS), un problema de optimización combinatoria en la teoría de grafos, mediante el uso de ordenadores cuánticos basados en átomos neutros. Para abordar este problema, se ha empleado un simulador cuántico donde se representa el grafo mediante una disposición de átomos en una cuadrícula. En esta configuración, los enlaces entre los nodos del grafo se reflejan en las interacciones entre los átomos. La solución se obtiene mediante una evolución adiabática del sistema, comenzando desde un estado inicial sencillo hasta alcanzar un estado final cuya configuración corresponde a la solución óptima del MWIS.

Para maximizar la probabilidad de obtener dicha solución óptima, se han aplicado tres algoritmos de optimización diferentes con el propósito de identificar aquellos protocolos que ofrecen el mejor rendimiento en la mayoría de los casos estudiados.

Además, el problema se ha analizado en profundidad, identificando características específicas de los grafos que dificultan su resolución. Se propone un nuevo parámetro (*Hardness Parameter*) basado en la estructura espectral de la función de coste del MWIS, que, combinado con otras variables espectrales, permite entrenar un modelo predictivo capaz de anticipar situaciones en las que el simulador podría fallar.

Finalmente, se evalúa el rendimiento de la metodología propuesta sobre un conjunto diverso de grafos y se analiza su escalabilidad al aplicarse a grafos de mayor tamaño.

**Palabras clave:** Optimización de *drivings* · *Quantum annealing* · Conjunto Independiente Máximo Ponderado · Computación Cuántica · Átomos neutros

## Abstract

This work investigates the resolution of the Maximum Weighted Independent Set (MWIS) problem, a combinatorial optimization problem in graph theory, through the use of neutral atom-based quantum computers. To address this problem, a quantum simulator has been employed where the graph is represented by an arrangement of atoms in a grid. In this configuration, the links between the nodes of the graph are reflected in the interactions between the atoms. The solution is obtained through an adiabatic evolution of the system, starting from a simple initial state until reaching a final state whose configuration corresponds to the optimal solution of the MWIS.

To maximize the probability of obtaining said optimal solution, three different optimization algorithms have been applied with the purpose of identifying those protocols that offer the best performance in most of the studied cases.

Furthermore, the problem has been analyzed in depth, identifying specific characteristics of the graphs that hinder their resolution. A new parameter (Hardness Parameter) is proposed based on the spectral structure of the MWIS cost function, which, combined with other spectral variables, allows for the training of a predictive model capable of anticipating situations in which the simulator might fail.

Finally, the performance of the proposed methodology is evaluated on a diverse set of graphs, and its scalability is analyzed when applied to larger graphs.

**Keywords:** Drivings optimization · Quantum annealing · Maximum Weighted Independent Set · Quantum Computing · Neutral atoms

## Resum

Aquest treball investiga la resolució del problema del conjunt independent màxim ponderat (*Maximum Weighted Independent Set*, MWIS), un problema d'optimització combinatòria en la teoria de grafs, mitjançant l'ús d'ordinadors quàntics basats en àtoms neutres. Per tal d'abordar aquest problema, s'ha fet servir un simulador quàntic on es representa el graf amb una disposició quadricular dels àtoms. En aquesta configuració, els enllaços entre els nodes del graf es reflectixen en les interaccions entre els àtoms. La solució s'obté mitjançant una evolució adiabàtica del sistema, començant des d'un estat inicial senzill fins abastar un estat final, la configuració del qual es correspon amb la solució òptima del MWIS.

Per a obtindre la solució òptima s'han aplicat tres diferents algorismes d'optimització, amb la finalitat d'identificar els protocols que oferixen un millor rendiment en la majoria de casos estudiats.

A més, el problema s'ha analitzat amb profunditat, identificant característiques específiques dels grafs que dificulten especialment la seua resolució. Es proposa un nou paràmetre (*Hardness Parameter*) basat en l'estructura espectral de la funció de cost del MWIS, que, combinat amb altres variables espectrals, permet entrenar un model predictiu capaç de trobar situacions on el simulador podria fallar.

Finalment, s'avalua el rendiment de la metodologia proposta en un conjunt divers de grafs i s'analitza la seua escalabilitat a l'aplicar-se a grafs de major tamany.

**Paraules Clau:** Optimització de *drivings* · *Quantum annealing* · Conjunt Independent Màxim Ponderat · Computació Quàntica · Àtoms neutres.

# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Motivación	1
1.2	Objetivos y estructura	2
<b>2</b>	<b>Marco Teórico</b>	<b>5</b>
2.1	Conjunto Independiente Máximo Ponderado y su resolución	5
2.1.1	Resolver el MWIS como un problema de programación lineal	6
2.1.2	Representación del MWIS como problema tipo QUBO	7
2.2	Introducción a la computación cuántica	7
2.2.1	Estados cuánticos, superposición y paralelismo	8
2.2.2	Teorema de la Adiabaticidad	10
2.3	Ordenadores cuánticos basados en átomos neutros	12
2.3.1	Descripción del <i>hardware</i> simulado	12
2.3.2	Átomos de Rydberg y oscilaciones de Rabi	13
2.3.3	Interacciones entre átomos y bloqueo de Rydberg	16
2.3.4	Hamiltoniano de Rydberg y su evolución	17
<b>3</b>	<b>Estado del arte</b>	<b>21</b>
3.1	Algoritmos cuánticos para resolver problemas de optimización combinatoria	21
3.2	Comparación de tecnologías para <i>Quantum Annealing</i>	23
3.3	<i>Quantum Annealing</i> para la resolución del problema MIS/MWIS	24
<b>4</b>	<b>Uso del simulador</b>	<b>25</b>
4.1	Conceptos clave	25
4.2	Ejemplo de uso y definición de métricas	27
<b>5</b>	<b>Resultados y discusión</b>	<b>33</b>
5.1	Generación de conjuntos de datos	33
5.2	Optimización de <i>drivings</i>	34
5.2.1	Objetivos de la optimización y desarrollo experimental	34
5.2.2	Comparación de métodos	36
5.2.3	Mejora de los parámetros fijos	43
5.2.4	Análisis de resultados	44
5.3	Generalización de los <i>drivings</i>	45
5.4	Optimización por grafo	50
5.5	Estudio de características de grafo que dificultan su resolución	52
5.5.1	Definición del <i>Hardness Parameter</i> (HP) para el MWIS	52
5.5.2	Entrenamiento del clasificador	55

5.6	Simulación de grafos de mayor tamaño . . . . .	58
<b>6</b>	<b>Conclusiones y líneas de futuro . . . . .</b>	<b>63</b>
6.1	Resumen de resultados . . . . .	63
6.2	Contribuciones principales . . . . .	64
6.3	Limitaciones del estudio . . . . .	65
6.4	Líneas de investigación futuras . . . . .	66
	<b>Bibliografía . . . . .</b>	<b>67</b>

---

# 1. Introducción

---

## 1.1. Motivación

Muchos de los retos tecnológicos actuales, desde planificar rutas óptimas hasta entrenar modelos de inteligencia artificial, comparten una misma raíz: la optimización combinatoria. Estos problemas consisten, en esencia, en explorar un espacio enorme de posibles soluciones para encontrar aquella que cumple ciertos criterios de forma óptima. Sin embargo, a medida que crece el tamaño del problema, el número de combinaciones posibles se dispara exponencialmente, haciendo que su resolución sea inviable incluso para los ordenadores clásicos más potentes. Esta limitación ha impulsado la búsqueda de nuevos enfoques computacionales que puedan sortear esta barrera, y entre ellos, la computación cuántica emerge como una de las alternativas más prometedoras [1–3].

Entre estos desafíos, destaca el problema del Conjunto Independiente Máximo Ponderado (*Maximum Weighted Independent Set*, MWIS) por su relevancia en la teoría de grafos y sus numerosas aplicaciones prácticas [3, 4]. El MWIS consiste en identificar un subconjunto de nodos donde ningún par de ellos está conectado por una arista, maximizando simultáneamente la suma de los pesos asociados a dichos nodos. Este problema clasificado como NP-duro [5, 6], representa un caso paradigmático de las limitaciones de los métodos clásicos para abordar problemas de optimización combinatoria.

El MWIS, aparece en múltiples contextos prácticos debido a su capacidad para modelar situaciones donde se requiere maximizar un criterio bajo restricciones de exclusión. Por ejemplo, en el diseño de redes inalámbricas [7], el problema consiste en seleccionar subconjuntos óptimos de nodos (puntos de acceso o estaciones) que no interfieran entre sí, maximizando así la calidad de la señal y minimizando las colisiones.

Asimismo, se puede encontrar en otros campos como el etiquetado de mapas (*map labeling*) [8], o en la planificación de turnos de trabajo en la industria [9]. Estas aplicaciones entre otras, destacan la versatilidad y la importancia práctica del problema MWIS, ya que puede ser planteado como un problema tipo QUBO (*Quadratic Unconstrained Binary Optimization*). Esto ha motivado el desarrollo de métodos avanzados para resolverlo, especialmente en instancias de gran escala donde los enfoques clásicos se vuelven computacionalmente inviables [5].

Es en este contexto donde surge la motivación por explorar las posibilidades de la computación cuántica. Esta, emerge como una herramienta revolucionaria, capaz de abordar problemas de una manera más eficiente y rápida, o incluso resolver problemas que son inabarcables para un ordenador clásico [10–12]. A diferencia de estos, los sistemas cuánticos aprovechan fenómenos físicos como la superposición, el entrelazamiento y la interferencia para explorar el espacio de soluciones de manera más eficiente [10, 12, 13]. Las ventajas de los ordenadores cuánticos han sido estudiadas tanto teórica como experimentalmente durante más de dos décadas para una variedad de algoritmos [14, 15].

En la computación cuántica la información se guarda en secuencias de bits cuánticos (cúbits) [12]. De manera análoga a un bit clásico, un cúbit es un objeto cuántico que

puede ocupar dos estados diferentes. Por ejemplo, si utilizamos átomos como cíbits, decimos que el estado fundamental de este corresponde al valor “0”, mientras que el estado excitado al “1”. Con un solo átomo, uno puede construir un número infinito de estados superpuestos utilizando los dos estados base del sistema, que corresponden al “0” y al “1” [12].

El concepto de superposición permite que un sistema cuántico represente simultáneamente múltiples posibles valores de entrada en un cálculo [10]. Esto es lo que conocemos como paralelismo cuántico [13]. A pesar de que este fenómeno no suponga ninguna ventaja en algoritmos para calcular la suma, producto o potencias, ha sido una pieza clave a la hora de desarrollar algoritmos que resuelvan problemas que fueran intratables para los ordenadores clásicos.

Un hito fundamental en la historia de la computación cuántica fue el algoritmo de factorización de números enteros propuesto por Shor [16], que generó un gran auge de interés en el campo [12]. En su artículo, Shor describe un algoritmo capaz de factorizar números enteros de manera exponencialmente más rápida que los mejores algoritmos clásicos [13], estableciendo así el primer algoritmo cuántico eficiente para un problema relevante [12]. De este modo, no solo demostró que la computación cuántica representaba una amenaza para la criptografía, sino que también mostró su enorme potencial, motivando a numerosos investigadores a explorar este campo.

Existen diversas formas de construir un ordenador cuántico [1], cada una basada en un sistema físico distinto y con métodos específicos para manipular los cíbits. Cada enfoque presenta ventajas y desafíos, y gran parte de la investigación actual se centra en determinar cuál de ellos es más adecuado para escalar los ordenadores cuánticos a un nivel práctico. Además, se espera que las características específicas de cada plataforma resulten más favorables para un conjunto limitado y particular de algoritmos [11].

A pesar de ello, analizar cada uno de ellos queda fuera de los objetivos de este proyecto. Por tanto, vamos a centrarnos en los sistemas de átomos neutros, que a pesar de ser una tecnología emergente, ha mostrado ser muy prometedora por su escalabilidad y la capacidad de controlar interacciones cuánticas mediante estados de Rydberg [2,11]. Esta tecnología, impulsada por los avances experimentales en matrices de pinzas ópticas y átomos neutros [4,17], ha demostrado ser una herramienta versátil tanto en el campo de la computación cuántica [18] como en el desarrollo de simuladores cuánticos [18,19].

## 1.2. Objetivos y estructura

En computación cuántica, es común codificar una función de coste en un Hamiltoniano, de modo que su estado fundamental (el de mínima energía) representa la solución óptima al problema planteado. Así, encontrar dicho estado equivale a resolver el problema de optimización original. En este contexto, los arreglos de átomos neutros manipulados mediante haces de luz han emergido como una de las tecnologías más prometedoras, gracias a su escalabilidad y al control preciso que permiten sobre sistemas compuestos por cientos o incluso miles de cíbits [4].

Para alcanzar la solución, se parte de un estado fundamental correspondiente a un Hamiltoniano inicial simple. Posteriormente, se realiza una transformación continua

de este Hamiltoniano hacia otro que codifica el problema de interés. Esta evolución se implementa mediante protocolos de luz controlada, conocidos como *drivings*. La forma en que se diseñen estos *drivings* será determinante: de ella depende que el sistema permanezca en su estado fundamental a lo largo del proceso, garantizando así que el estado final medido corresponda con la solución óptima.

Sin embargo, un aspecto clave del problema es que los *drivings* óptimos varían de un grafo a otro, y en general no se conocen de antemano. Esto limita la posibilidad de adaptar los protocolos de forma personalizada para cada instancia. Por esta razón, uno de los objetivos principales de este trabajo es identificar protocolos de *drivings* genéricos, que funcionen razonablemente bien para una amplia variedad de grafos sin necesidad de ajuste individual. Esta aproximación busca equilibrar rendimiento y aplicabilidad, permitiendo aplicar la misma estrategia a problemas desconocidos o en tiempo real.

Además, se explora la caracterización estructural de los grafos con el fin de identificar qué propiedades o características hacen que una instancia sea más difícil de resolver. Para ello, se introduce y analiza una métrica denominada *Hardness Parameter* (HP), que ayuda a cuantificar la dificultad empírica de cada grafo en función del rendimiento observado durante la simulación. Este análisis tiene como objetivo no solo entender qué grafos son más complicados, sino también propiedades o rasgos estructurales que puedan predecir esa dificultad.

El trabajo se organiza de la siguiente manera. En el Capítulo 2, además de describir en detalle el problema del Conjunto Independiente Máximo Ponderado (MWIS), se presentan los fundamentos de mecánica cuántica necesarios para comprender tanto el funcionamiento del ordenador cuántico como los resultados que se analizarán posteriormente. El Capítulo 3 está dedicado al estado del arte, donde se revisan los enfoques más relevantes en la literatura relacionados con la resolución cuántica de problemas de optimización.

En el Capítulo 4 se explica el funcionamiento del simulador utilizado y, mediante un ejemplo sencillo, se introducen las métricas que se emplearán para comparar los resultados obtenidos. A continuación, el Capítulo 5 recoge y discute los resultados experimentales, incluyendo los efectos de la optimización de *drivings* y el análisis estructural de la dificultad de los grafos.

Por último, en el Capítulo 6 se presentan las conclusiones generales del trabajo y se resumen las limitaciones del mismo. Además, se plantean posibles líneas de investigación futura.

A lo largo del trabajo, se hace referencia a diferentes *notebooks*, que se encuentran en el repositorio [GitHub](#)<sup>1</sup> del proyecto. El lector puede clonarlo y ejecutarlos siguiendo las indicaciones que se muestran en el mismo.

---

<sup>1</sup>[https://github.com/AitorArribas/TFM\\_Aitor\\_Arribas.git](https://github.com/AitorArribas/TFM_Aitor_Arribas.git)

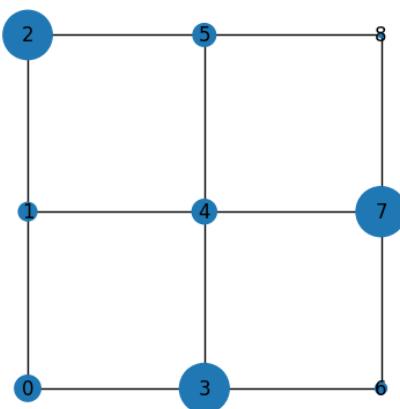


## 2. Marco Teórico

### 2.1. Conjunto Independiente Máximo Ponderado y su resolución

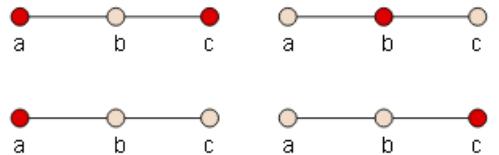
En este trabajo, se asume que el lector posee conocimientos básicos sobre teoría de grafos, como la definición de vértices, aristas y conceptos fundamentales relacionados. Para aquellos que no estén familiarizados con estos temas, recomendamos consultar el capítulo 2 de [21], donde se presentan estos conceptos de manera detallada.

El problema del Conjunto Independiente Máximo (*Maximum Independent Set*, MIS) es un problema popular en teoría de grafos y optimización combinatoria [7, 15, 22–24], con aplicaciones en diversas áreas de la computación. Formalmente, podemos definir un conjunto independiente (*independent set*, IS) de un grafo  $G(V, E)$  como un subconjunto  $S \subset V$  de vértices, de manera que no existen arcos de  $G$  que unan dos vértices de  $S$ . Por otro lado, decimos que un conjunto independiente de  $G$  es maximal cuando este no se puede hacer más grande. El tamaño de un IS viene dado por el número de vértices que lo conforman. El MIS de  $G$  es el conjunto independiente de mayor tamaño [21].



**Figura 2.2:** Ilustración de un grafo con nodos ponderados. El número sobre cada nodo indica su etiqueta y el tamaño de los mismos es proporcional al peso asociado.

en un plano bidimensional y se conectan si su distancia es menor que una unidad  $r$ . La dependencia radial que comparte este tipo de grafos con los fundamentos de los



**Figura 2.1:** Todos los conjuntos independientes de un grafo formado por los vértices  $a$ ,  $b$ , y  $c$  y los arcos  $\{a, b\}$  y  $\{b, c\}$ . Imagen extraída de [20].

En la Figura 2.1, el grafo con los vértices  $a$ ,  $b$  y  $c$  y las aristas  $\{a, b\}$  y  $\{b, c\}$  tiene cuatro conjuntos independientes posibles. Los conjuntos  $\{a\}$  y  $\{c\}$  son independientes, pero no son maximales, ya que existe un conjunto mayor ( $\{a, c\}$ ) que los contiene. Por otro lado,  $\{b\}$  sí es un conjunto independiente maximal, ya que no puede ampliarse sin perder la independencia. Finalmente,  $\{a, c\}$  es el conjunto independiente máximo, pues es el más grande de todos.

En este trabajo nos centraremos en una clase particular de grafos llamada grafos de discos unitarios (*unit-disk graphs*, UDG). Estos grafos son fundamentales en ciertas implementaciones de computación cuántica con átomos neutros debido a su estrecha relación con el bloqueo de Rydberg, un fenómeno físico que juega un papel clave en estos sistemas. En estos grafos, los vértices se ubican

átomos neutros hace que estos sistemas sean una plataforma natural para estudiar el problema.

El problema MIS puede extenderse a su versión ponderada, conocida como *Maximum Weighted Independent Set* (MWIS). En este caso, cada nodo  $i$  tiene un peso  $w_i$ , y el objetivo ya no es maximizar el número de nodos en el conjunto independiente, sino la suma de sus pesos. Matemáticamente, para un grafo  $G(V, E)$ , el MWIS se puede obtener maximizando la siguiente función de coste:

$$C_{MWIS} = \sum_{i \in V} w_i x_i \quad \text{sujeto a} \quad x_i + x_j \leq 1 \quad \forall (i, j) \in E \quad (2.1)$$

donde  $w_i$  indica el peso del nodo  $i$  y  $x_i \in \{0, 1\}$  indica si el nodo pertenece ( $x_i = 1$ ) o no ( $x_i = 0$ ) a la solución.

En la Figura 2.2 se muestra el ejemplo de un grafo ponderado. El peso de los nodos se muestra mediante el tamaño de los mismos. En este caso, vemos claramente que la solución al MWIS sería el subconjunto  $\{2, 3, 7\}$ . Este ejemplo ilustra con claridad la diferencia entre MWIS y MIS, ya que la solución al MIS vendría dada por el subconjunto independiente más grande, que en este caso sería el  $\{0, 2, 4, 6, 8\}$ . Este ejemplo se desarrollará más en la Sección 4.2.

Tanto el MIS como el UD-MIS son problemas NP-duros [6, 19], lo que significa que, aunque en teoría es posible encontrar su solución exacta, el tiempo de cómputo que requiere crece exponencialmente con el tamaño del problema, volviéndose inabordable para instancias grandes. En la literatura, se han explorado diferentes algoritmos clásicos para abordar el problema buscando soluciones exactas [23, 24].

### 2.1.1. Resolver el MWIS como un problema de programación lineal

En este trabajo se ha utilizado el solucionador clásico Gurobi [25] para conocer la solución del MWIS en diferentes grafos y así poder comparar los resultados. Gurobi es un optimizador diseñado para resolver programas lineales, programas lineales enteros mixtos y programas cuadráticos, entre otros. Para encontrar el MWIS mediante el algoritmo de optimización de Gurobi definimos un archivo LP (*Linear Programming*) que describe nuestro problema. Siguiendo el ejemplo de la Figura 2.2, el LP asociado sería como se muestra a continuación:

```

Maximizar
  obj: w0x0 + w1+x1 + ... + w8x8
Sujeto a
  c1: x0 + x1 <= 1
  c2: x1 + x2 <= 1
  :
  c12: x7 + x8 <= 1
Variables binarias

```

```

x0 x1 x2 x3 x4 x5 x6 x7 x8
Fin

```

Las variables binarias corresponden a los nodos de  $G(V, E)$  y toman el valor 1 si y solo si pertenecen a la solución. Cada restricción representa el arco entre dos nodos. Por ejemplo, la restricción  $c1: x0 + x1 \leq 1$  corresponde al arco  $(0, 1) \in E$ , lo que impide que ambos nodos pertenezcan a la solución.

### 2.1.2. Representación del MWIS como problema tipo QUBO

Dado que las variables del MWIS son binarias, este problema puede reformularse como un problema de Optimización Binaria Cuadrática No Restringida (*Quadratic Unconstrained Binary Optimization*, QUBO) [22, 26]. Esta representación es especialmente útil en computación cuántica y algoritmos inspirados en mecánica cuántica, ya que puede implementarse en hardware especializado como ordenadores cuánticos basados en átomos neutros u otros sistemas basados en *quantum annealing*. De esta manera, la función objetivo queda:

$$C(\mathbf{x}) \equiv - \sum_i Q_{ii} x_i + \sum_{i,j} Q_{ij} x_i x_j \quad (2.2)$$

donde  $\mathbf{x} = (x_i) \in \{0, 1\}^{\otimes n}$  es un vector  $n$ -dimensional con elementos binarios, siendo  $n$  el número de nodos del grafo. La matriz  $\mathbf{Q}$  es una matriz simétrica donde  $Q_{ii}$ ,  $i \in \{1, \dots, n\}$  indica el peso de cada nodo y  $Q_{ij}$  las conexiones entre nodos [26, 27]. Si analizamos la Ecuación (2.2), vemos que el primer término favorece la activación de los nodos, mientras que el segundo impide que dos nodos conectados aparezcan en la solución siempre que  $Q_{ij}$  sea mayor que  $Q_{ii}$  para todo  $i, j$  [28, 29]. Como veremos más adelante, esta representación permite ver claramente por qué los átomos neutros son una herramienta muy apropiada para resolver el MWIS.

## 2.2. Introducción a la computación cuántica

El aprendizaje automático es un campo que busca desarrollar algoritmos capaces de identificar patrones y relaciones en los datos para comprender la naturaleza que los rige. Aunque esta visión es simplificada, la mayoría de estos algoritmos incorporan algún tipo de optimización basada en una función de coste.

La computación cuántica ofrece una nueva perspectiva para abordar estos problemas, aprovechando las propiedades de los sistemas cuánticos para desarrollar algoritmos con potenciales ventajas sobre los clásicos. Esto ha dado lugar al aprendizaje automático cuántico, una disciplina emergente que busca integrar herramientas cuánticas en el procesamiento de datos y la optimización.

La mecánica cuántica es un conjunto de teorías que describen la naturaleza a nivel fundamental [10]. En este trabajo, nos centraremos únicamente en los conceptos esenciales para comprender la aplicación de la computación cuántica en el aprendizaje automático. Estos conceptos se introducirán de manera conceptual, evitando en la me-

dida de lo posible los desarrollos matemáticos y otros aspectos teóricos más avanzados. No obstante, quienes deseen profundizar en estos temas pueden consultar [10] y [12].

### 2.2.1. Estados cuánticos, superposición y paralelismo

En computación clásica, la información se almacena en cadenas de bits. Un bit puede tomar el valor 1 o 0, dependiendo de la presencia o ausencia de una diferencia de voltaje, de manera que solo puede encontrarse en uno de estos dos estados. Físicamente, podríamos construir un bit con cualquier sistema que tenga dos estados distinguibles, como una bombilla encendida o apagada, o una moneda mostrando cara o cruz.

De manera análoga, un bit cuántico (o cúbit) se define mediante un sistema cuántico de dos niveles. Por ejemplo, en un átomo, se pueden utilizar dos estados cuánticos: el estado fundamental y un estado excitado. En este caso, decimos que el cúbit se encuentra en el estado  $|1\rangle$  si el átomo está excitado y en el estado  $|0\rangle$  si se muestra en su estado fundamental.

En mecánica cuántica, los estados se representan mediante vectores en el espacio Hilbert. En este caso, al considerar únicamente dos estados, podemos describir  $|0\rangle$  y  $|1\rangle$  como<sup>1</sup>

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (2.3)$$

Para expresar estos estados, utilizamos la llamada notación de Dirac, ampliamente usada en física cuántica. En ella, un vector se escribe como  $|\psi\rangle$ , donde el símbolo  $\psi$  es simplemente una etiqueta arbitraria que le asignamos y el símbolo  $|\cdot\rangle$  se conoce como *ket*. Dado que los estados son vectores y están relacionados con probabilidades, normalizamos los estados tal que  $\|\psi\| = 1$ . Asimismo, en este trabajo también aparecerá el término  $\langle\psi|$  (a  $\langle\cdot|$  se le llama *bra*). Matemáticamente,  $\langle\psi|$  es el conjugado transpuesto de  $|\psi\rangle$ , tal que

$$\langle\psi| = |\psi\rangle^\dagger. \quad (2.4)$$

Mediante esta notación, podemos escribir el producto escalar entre dos estados  $|\phi\rangle$  y  $|\psi\rangle$  como  $\langle\phi|\psi\rangle$ . De manera similar al producto escalar habitual, este, indica cómo de similares son los estados  $|\phi\rangle$  y  $|\psi\rangle$ . Cabe recalcar que  $\langle\phi|\psi\rangle$  (un escalar) no es lo mismo que  $|\phi\rangle\langle\psi|$  (un operador), de la misma manera que

$$(1 \ 0) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot (1 \ 0). \quad (2.5)$$

La diferencia principal entre un bit clásico y un cúbit radica en que, mientras un bit solo puede estar en uno de los dos estados posibles (0 o 1), un cúbit puede encontrarse en una combinación lineal de ambos estados [10], lo que se expresa matemáticamente como

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (2.6)$$

donde  $\alpha$  y  $\beta$  son números complejos, y  $|\psi\rangle$  representa el estado del cúbit.

---

<sup>1</sup>Los vectores que se muestran son únicamente un ejemplo ilustrativo, no tienen por qué ser esos.

Este fenómeno, conocido como **superposición cuántica**, implica que el cúbbit puede existir en ambos estados a la vez. Sin embargo, cuando se mide su estado, solo se observará  $|0\rangle$  o  $|1\rangle$  con una probabilidad determinada por los valores de  $|\alpha|^2$  y  $|\beta|^2$ , que deben cumplir:

$$|\alpha|^2 + |\beta|^2 = 1 . \quad (2.7)$$

Para entender la ventaja de los ordenadores cuánticos, consideremos un sistema de dos cúbits. Este sistema puede describirse mediante cuatro estados base:  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$  y  $|11\rangle$ . A primera vista, esto podría parecer similar a un sistema de dos bits clásicos, donde bastaría con especificar en qué estado se encuentra cada bit. Sin embargo, la mecánica cuántica permite que el sistema esté en una superposición de estos cuatro estados, lo que significa que su estado general puede expresarse como

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle , \quad (2.8)$$

donde ahora hay cuatro coeficientes  $(\alpha, \beta, \gamma, \delta)$  que describen completamente el estado del sistema.

En contraste, en un sistema clásico de dos bits, solo se necesita especificar en qué estado se encuentra cada bit, lo que requiere únicamente dos valores para describir el estado del sistema. La diferencia se hace aún más notable a medida que el número de cúbits aumenta. Un sistema de  $n$  cúbits requiere  $2^n$  coeficientes para describir su estado, mientras que un sistema clásico de  $n$  bits solo necesita  $n$  valores. Este aumento en el número de coeficientes exponencial es una de las claves del potencial de la computación cuántica.

Sin embargo, hay una limitación importante: no podemos observar directamente un estado en superposición. Siempre que realizamos una medición, decimos que el cúbbit colapsa a uno de los estados base ( $|0\rangle$  o  $|1\rangle$ ). Esto significa que, aunque un ordenador cuántico puede procesar información en paralelo, no podemos acceder directamente a toda esta información sin que la superposición se destruya.

Una de las características más relevantes de los ordenadores cuánticos es el **parallelismo cuántico**. Como hemos visto, un cúbbit puede estar en una superposición de los estados  $|0\rangle$  y  $|1\rangle$ , y un sistema de  $n$  cúbits puede estar en una superposición de los  $2^n$  estados posibles.

En computación clásica, una operación sobre un bit transforma un solo valor de entrada en un solo valor de salida. En cambio, en computación cuántica, si aplicamos una operación a un cúbbit que está en superposición, esa operación se aplica simultáneamente a cada uno de los términos de la superposición [10]. Por ejemplo, supongamos que tenemos un cúbbit en el estado

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (2.9)$$

y aplicamos una puerta cuántica  $U$ , entonces el resultado será

$$U|\psi\rangle = \frac{1}{\sqrt{2}}(U|0\rangle + U|1\rangle) . \quad (2.10)$$

Esto significa que, en un solo paso, la operación  $U$  ha sido aplicada simultáneamente a los dos estados base. De manera análoga, en un sistema de  $n$  cíbits, una operación cuántica puede procesar en paralelo los  $2^n$  estados de la superposición.

Sin embargo, es importante notar que este paralelismo cuántico no significa que podamos acceder a todos los resultados a la vez. En otras palabras, aunque la computación cuántica permite realizar muchas operaciones en paralelo, la información obtenida tras la medición es una única muestra del resultado final. Al aplicar un operador cuántico a un estado superpuesto obtendremos como resultado un solo valor al igual que en el caso clásico. Por esta razón, el paralelismo cuántico no es útil por sí solo. Para aprovecharlo, los algoritmos cuánticos están diseñados de manera que las interferencias cuánticas entre los estados permitan amplificar las soluciones correctas y cancelar las incorrectas.

En resumen, los ordenadores cuánticos no simplemente prueban todas las combinaciones posibles en paralelo y devuelven la respuesta correcta. Más bien, aprovechan el paralelismo cuántico y la interferencia para hacer que la solución correcta sea la más probable tras la medición.

Estos fenómenos como la superposición, paralelismo e interferencia son la gran ventaja de los ordenadores cuánticos, y a la vez la razón por la que no serán un reemplazo para los ordenadores clásicos. Un ordenador cuántico tendrá ventaja sobre uno clásico únicamente en los casos en los que estos fenómenos puedan ser utilizados a su favor. En el resto de situaciones (hacer búsquedas de internet, mirar videos, escribir documentos...), en el mejor de los casos funcionaría exactamente igual que un ordenador clásico.

### 2.2.2. Teorema de la Adiabaticidad

En física, los sistemas suelen describirse mediante Hamiltonianos, los cuales contienen la información sobre los objetos que los componen y la manera en la que interactúan entre sí. Al expresar toda esta información en una fórmula matemática, los Hamiltonianos permiten estudiar la evolución del sistema, calcular la energía de distintos estados y determinar cuáles son los estados más probables, lo que resulta fundamental en computación cuántica.

Para facilitar su comprensión, podemos establecer un símil con un concepto más familiar en el ámbito del aprendizaje automático: la función de coste. Al igual que el Hamiltoniano, esta función encapsula toda la información del problema a resolver. En muchos problemas computacionales, se define una función que asigna un valor numérico a cada posible solución, indicando qué tan buena o mala es. El objetivo suele ser encontrar la solución que minimiza esta función, es decir, aquella con el menor coste. De manera análoga, en sistemas cuánticos, los estados del sistema pueden asociarse con distintos niveles de energía determinados por el Hamiltoniano, y el estado fundamental es aquel que minimiza la energía total del sistema.

Cuando el Hamiltoniano del sistema es independiente del tiempo, la evolución del sistema puede calcularse de forma relativamente sencilla. Sin embargo, cuando el Hamiltoniano varía con el tiempo, la dinámica del sistema se vuelve más compleja [30]. No obstante, si esta variación ocurre de manera suficientemente lenta, la evolución

sigue una trayectoria relativamente simple y, en muchos casos, puede describirse de manera aproximada [31]. El teorema de la adiabaticidad es la formalización de esta aproximación.

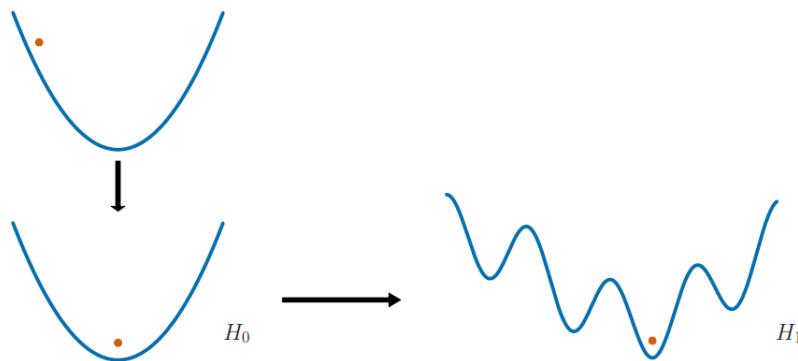
Este teorema, introducido en mecánica cuántica por Born y Fock [32], tiene múltiples aplicaciones en computación cuántica. Su enunciado establece que:

*Si un sistema cuántico comienza en un autoestado del Hamiltoniano inicial y la evolución del Hamiltoniano ocurre de manera lo suficientemente lenta, entonces el sistema permanecerá en el autoestado instantáneo correspondiente en cada instante de tiempo, siempre que exista un gap energético entre este autoestado y el resto del espectro del Hamiltoniano [30].*

En términos más simples, consideremos un sistema descrito por el Hamiltoniano dependiente del tiempo

$$H(\lambda) = (1 - \lambda)H_0 + \lambda H_1, \quad (2.11)$$

donde  $\lambda = \lambda(t)$  es un parámetro temporal y varía entre  $[0, 1]$ . En general, los autoestados cambian con el tiempo y, por tanto, las soluciones no pueden expresarse simplemente como estados estacionarios del sistema completo [30]. Sin embargo, en el límite en el que la transformación de  $H(\lambda)$  ocurre de manera infinitamente lenta, si el sistema comienza en un estado estacionario, evolucionará siguiendo los estados estacionarios intermedios de  $H(\lambda)$  en cada instante [30].



**Figura 2.3:** Ilustración de cómo mediante el teorema de la adiabaticidad podemos obtener el estado fundamental de un Hamiltoniano sencillo  $H_0$  y mediante una evolución adiabática, obtener el estado fundamental del Hamiltoniano  $H_1$ . Esta imagen se ha obtenido de [13].

Así pues, supongamos que el sistema comienza en el estado fundamental de  $H(0) = H_0$ . Si la transición se lleva a cabo de manera que aumentamos  $\lambda$  lo suficientemente despacio, entonces acabaremos en el estado fundamental de  $H(1) = H_1$  siempre y cuando haya un *gap* entre el estado fundamental y el resto del espectro [13].

## 2.3. Ordenadores cuánticos basados en átomos neutros

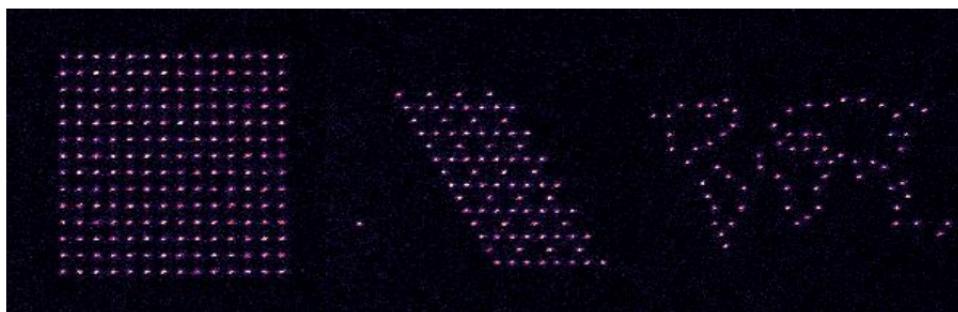
En la computación cuántica simulada, existen dos enfoques principales, los Circuitos Cuánticos Digitales, y los Simuladores de Hamiltonianos Analógicos. Los primeros utilizan una secuencia de puertas cuánticas (como Hadamard, CNOT, etc.) que permiten manipular estados cuánticos y hacer cálculos, de manera similar a un circuito clásico. Por otro lado, están los simuladores de hamiltoniano analógicos, que utilizan sistemas cuánticos con el objetivo de simular la evolución temporal del hamiltoniano de otro sistema. Así, los estados evolucionan a lo largo del tiempo debido a campos externos (láseres, campos magnéticos...). La simulación se da de manera continua a lo largo del tiempo y no mediante puertas discretas, es por eso que se le llama analógico [33].

En este caso se ha hecho uso del enfoque de Simulación del Hamiltoniano Analógico, donde tenemos un sistema cuántico definido por la organización de átomos, y lo perturbamos mediante una fuerza externa que serían los láseres. Variando los parámetros de estos láseres a lo largo del tiempo hacemos evolucionar el hamiltoniano que describe el sistema. Así, el objetivo es que el hamiltoniano final imite el problema que queremos estudiar; en este caso, la función de coste del MWIS.

### 2.3.1. Descripción del *hardware* simulado

En este trabajo se describen los componentes principales y las ideas clave del dispositivo experimental. Para obtener una descripción más detallada del ordenador cuántico, el lector puede consultar la guía publicada por los mismos desarrolladores [34].

Se ha utilizado el simulador del ordenador cuántico basado en átomos neutros Aquila, un dispositivo accesible a través del servicio en la nube Braket, de Amazon Web Services (AWS). Aquila se describe como una matriz de cúbicos programables en el campo, con una arquitectura configurable por el usuario, capaz de ejecutar dinámicas cuánticas coherentes programables en hasta 256 cúbicos de átomos neutros. La Figura 2.4 ilustra la versatilidad del sistema para definir las configuraciones de átomos.



**Figura 2.4:** Ejemplo de tres configuraciones de átomos diferentes. Izquierda: matriz de átomos. Centro: cúbicos dispuestos en una red Kagome. Derecha: cúbicos organizados en la forma de las costas del mundo para codificar un problema de optimización geográfica. Esta imagen se ha obtenido de [34].

Aquila es un dispositivo cuántico capaz de operar a temperatura ambiente. Para ello, utiliza átomos neutros de rubidio-87 ( $\text{Rb-87}$ ), los cuales son atrapados y enfriados a temperaturas del orden de microkelvin mediante láseres dentro de una cámara de vacío. Estos átomos actúan como cúbits, utilizando sus estados electrónicos para almacenar y manipular información cuántica. Cada cúbbito representa uno de los nodos del grafo que se quiere resolver, de manera que la disposición de los átomos será una representación física del problema.

La manipulación y reorganización de los átomos se lleva a cabo mediante haces de luz focalizados, que actúan como pinzas ópticas utilizando la fuerza dipolar óptica. Este proceso se basa en inducir un dipolo eléctrico en el átomo mediante un láser que resuena con un estado intermedio. Focalizando el láser en un solo punto, se logra atrapar al átomo. Sin embargo, existen limitaciones experimentales en el diseño de la disposición de átomos. Debido a la capacidad de focalización del láser, dos átomos no pueden estar a menos de  $4 \mu\text{m}$  de distancia. Además, el área disponible para colocar los átomos es de  $75 \mu\text{m} \times 76 \mu\text{m}$ .

Una vez establecida la disposición de los átomos deseada, las trampas ópticas se apagan para evitar que interfieran en las interacciones cuánticas. La evolución del sistema se controla mediante dos láseres. Por un lado, un láser global ilumina homogéneamente todo el conjunto, permitiendo controlar las transiciones de estado y las interacciones entre los átomos de forma simultánea. Por otro lado, un láser local actúa de manera selectiva sobre átomos individuales, permitiendo modular sus transiciones de estado de forma independiente y controlada, lo que posibilita la excitación o inhibición selectiva de cada átomo. La evolución y el control del sistema se logran mediante estos láseres, que transforman el sistema de manera que el Hamiltoniano final codifica la solución al grafo en su estado fundamental. Como se ha comentado anteriormente, la computación cuántica funciona cuando conseguimos que la solución al problema deseado sea el resultado más probable. En este caso, esto se consigue mediante el Teorema de Adiabaticidad.

Por último, la medida del resultado se realiza mediante el fenómeno de fluorescencia, el cual depende directamente del estado electrónico del átomo. Al acabar la evolución, se vuelven a encender las trampas ópticas, lo que hace que los cúbits colapsen a uno de los estados base ( $|0\rangle$  o  $|1\rangle$ ). Los átomos en el estado fundamental son reatrappados, mientras que los que se encuentren en el estado excitado son expulsados por la interacción con el láser. A continuación, el estado se mide por fluoerescencia, por la presencia o ausencia de átomo. Cuando no se observe el átomo, significa que estaba en el estado excitado y viceversa.

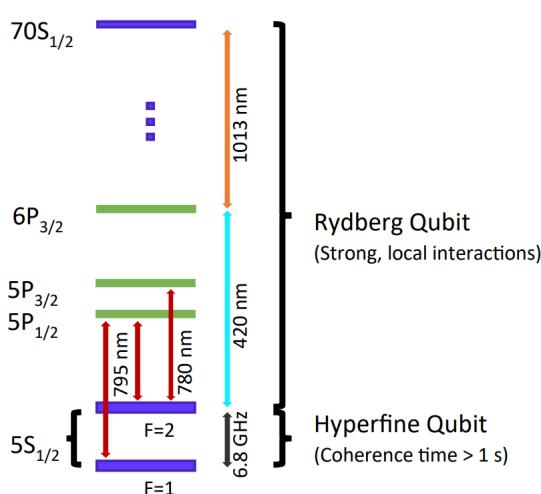
### 2.3.2. Átomos de Rydberg y oscilaciones de Rabi

Cuando un átomo se encuentra en un estado excitado con un número principal  $n$  alto, el electrón de valencia se sitúa a una gran distancia del núcleo, provocando que se comporte de manera similar a un átomo de hidrógeno. Es decir, el electrón de valencia experimenta el resto del átomo como una carga puntual [35]. Estos estados, conocidos como estados de Rydberg, presentan propiedades notables, como su gran tamaño y el hecho de que el electrón excitado está débilmente ligado al núcleo [4, 36].

Los átomos de Rydberg presentan una serie de propiedades extraordinarias que los hacen especialmente interesantes en múltiples aplicaciones cuánticas [37]. Entre ellas, destacan su elevado momento dipolar de transición y sus largos tiempos de vida. El alto momento dipolar de transición es consecuencia directa del gran tamaño de estos átomos y la separación de carga entre el núcleo y el electrón altamente excitado [4, 38]. Esta propiedad los hace especialmente sensibles a campos electromagnéticos externos, lo que permite su manipulación precisa [37].

Por otro lado, los largos tiempos de vida de los estados de Rydberg se deben principalmente a la baja probabilidad de desexcitación espontánea. Esto se explica porque, al encontrarse el electrón tan alejado del núcleo, el solapamiento entre su función de onda y los niveles de menor energía (incluyendo el estado fundamental) es muy bajo. Como resultado, las transiciones radiativas hacia estados de menor energía se ven fuertemente suprimidas, reduciendo la emisión espontánea de fotones [35].

Esta característica es crucial para la fiabilidad de las mediciones en experimentos cuánticos, ya que una desexcitación espontánea podría alterar la lectura de los resultados. Además, gracias a su alta sensibilidad a campos electromagnéticos externos y su larga coherencia cuántica, los átomos de Rydberg permiten un control preciso de sus estados mediante la aplicación de campos electromagnéticos [37, 39].



**Figura 2.5:** Estados electrónicos del Rb-87 que Aquila utiliza para manipular el átomo a modo de cúbit. Las flechas indican transiciones que se pueden inducir mediante haces de luz. Las líneas moradas indican los estados que representa el cúbit, mientras que las verdes indican otros estados que se utilizan para manipularlo. Esta imagen ha sido extraída de [13].

incidir un láser cuya frecuencia es cercana a la frecuencia de transición del átomo, se produce una interacción entre el campo electromagnético del láser y el átomo. Esta interacción induce transiciones entre los dos estados, dando lugar a la absorción y emisión de fotones: cuando el átomo absorbe un fotón, pasa al estado excitado  $|r\rangle$ ; al emitir un fotón, retorna al estado fundamental  $|g\rangle$ .

En la Figura 2.5 se muestra la estructura electrónica del átomo de Rb-87. El cúbit de Rydberg es representado mediante dos estados. El estado fundamental, que se define como el  $|0\rangle \equiv |g\rangle = |5S_{1/2}\rangle$ , y el estado de Rydberg, que viene dado por  $|1\rangle \equiv |r\rangle = |70S_{1/2}\rangle$ .

### Interacción con la luz

Las transiciones entre distintos estados electrónicos ocurren mediante la absorción y emisión de fotones. Cuando un átomo se expone a luz cuya energía coincide con la de una transición específica, puede absorber o emitir un fotón de manera cuántica, lo que provoca su transición entre dos estados diferentes [40].

Consideremos un sistema cuántico compuesto por un átomo con dos posibles estados  $|g\rangle$  y  $|r\rangle$ , donde  $|g\rangle$  corresponde al estado base y  $|r\rangle$  al estado excitado. Al

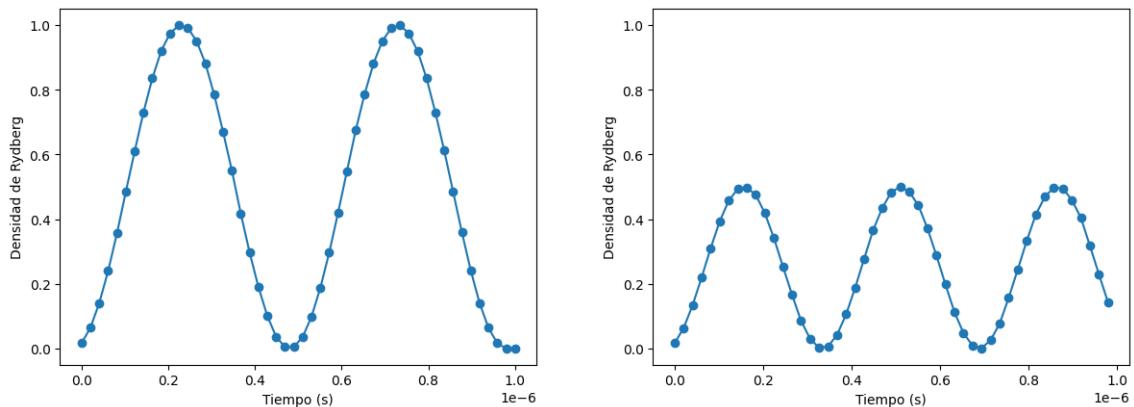
Este sistema compuesto por un átomo y el láser, se puede describir mediante el Hamiltoniano

$$H_{drive} = \frac{\Omega}{2} (e^{i\phi} |g\rangle \langle r| + e^{-i\phi} |r\rangle \langle g|) - \Delta |r\rangle \langle r|, \quad (2.12)$$

donde  $\Omega$  es la amplitud de Rabi, expresa la frecuencia a la que un átomo oscila entre el estado  $|g\rangle$  y  $|r\rangle$  y es directamente proporcional a la amplitud del láser. Asimismo,  $\Delta = \omega_{gr} - \omega_0$  se conoce como el *detuning* y describe la diferencia de frecuencia entre la frecuencia de transición del átomo  $\omega_{gr}$  y el láser  $\omega_0$ . Por último,  $\phi$  es un término de fase del estado. Sin embargo, para seguir este trabajo podemos no tenerlo en cuenta, ya que los términos complejos no tienen efectos observables.

Analicemos la Ecuación 2.12 por partes. El primer término es el que induce una superposición entre los estados  $|g\rangle$  y  $|r\rangle$ . Esto se debe a que el término  $|g\rangle \langle r|$  lleva los átomos que estén en el estado  $|g\rangle$  al estado  $|r\rangle$ . Asimismo, esta superposición es regulada mediante  $\Omega$ , por lo que al aumentar la amplitud del láser, aumentamos la superposición en el átomo.

Por otro lado, en el segundo término vemos que únicamente aparece el estado  $|r\rangle$ . Esto indica que esta contribución al Hamiltoniano es no nula solo cuando el átomo se encuentra en el estado  $|r\rangle$ . El término de *detuning* indica cuán fuera de resonancia se encuentra el láser y modulando su valor, podemos establecer si el estado  $|r\rangle$  será energéticamente más favorable o no.



**Figura 2.6:** Densidad de Rydberg en función del tiempo para un solo átomo. Estos resultados se han obtenido simulando el Hamiltoniano de la Ecuación (2.12) con  $\Omega = 2$  MHz,  $\Delta = 0$  MHz (izquierda) y  $\Omega = \Delta = 2$  MHz (derecha). La densidad de Rydberg se ha obtenido promediando 20.000 simulaciones.

En la Figura 2.6 se observa el efecto en la densidad de población del estado de Rydberg para  $\Delta = 0$  y  $\Delta \neq 0$  cuando incidimos en un átomo con un haz<sup>2</sup>.

- $\Delta = 0$ : La frecuencia del láser coincide con el de la frecuencia de transición y el término de Rabi es el único que aparece en el Hamiltoniano. En este caso el átomo transiciona entre el estado  $|g\rangle$  y  $|r\rangle$  de manera sinusoidal.
- $\Delta \neq 0$ : La frecuencia del láser ya no coincide con la frecuencia de transición, por lo que el átomo oscila de manera sinusoidal entre el estado  $|g\rangle$  y algún estado en

<sup>2</sup>El lector puede dirigirse al notebook `Oscilaciones_de_rabi.ipynb` y observar este efecto para diferentes valores de  $\Omega$  y  $\Delta$  si lo desea.

superposición.

Esta idea de las oscilaciones de Rabi es clave para llegar a la solución, ya que sin  $\Omega$ , el sistema sería incapaz de salir del estado fundamental y no podría explorar el espacio de soluciones. Asimismo, si el *detuning* es muy grande, las oscilaciones de Rabi no se alejarán tampoco del estado fundamental y tendrán dificultades para explorar el mismo.

### 2.3.3. Interacciones entre átomos y bloqueo de Rydberg

El bloqueo de Rydberg es un fenómeno que ocurre cuando dos átomos cercanos están excitados en estados altamente energéticos (estados de Rydberg). En estos estados, los átomos muestran una interacción a larga distancia dipolo-dipolo [4, 36], de manera que el hecho de que un átomo se encuentre en el estado de Rydberg impide la excitación de otro átomo cercano [41, 42].

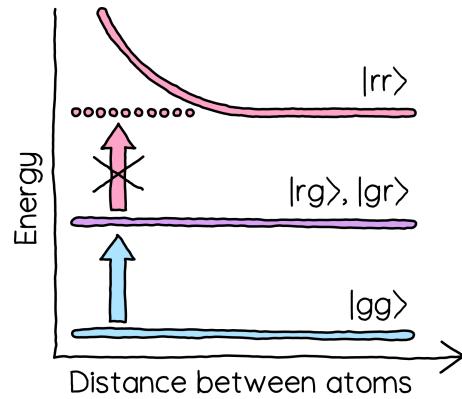
Consideremos un sistema formado por dos átomos de Rydberg cercanos que pueden estar en el estado fundamental  $|g\rangle$  o el estado de Rydberg  $|r\rangle$ . Cuando ambos se encuentran en el estado fundamental, la energía del sistema es igual a la suma de energías del estado de cada átomo  $E_g + E_g$ . Cuando uno de los átomos es excitado, una vez más, la energía del sistema es igual a la suma de las energías de los estados,  $E_g + E_r$  aunque ahora es mayor. No obstante, cuando ambos átomos se encuentran excitados, es necesario considerar la energía de interacción entre los átomos, por lo que la energía del sistema es  $E_r + E_r + E_{int}$ . Como vemos en la Figura 2.7, la energía de esta interacción decrece con la distancia, de manera que para átomos lejanos es despreciable.

En un sistema de átomos de Rydberg la interacción entre cúbicos viene dada por la interacción van der Waals entre átomos [41, 44]. Este, puede ser descrito mediante el Hamiltoniano

$$H_{int} = \sum_{i < j} \frac{C_6}{|\vec{x}_i - \vec{x}_j|^6} \hat{n}_i \hat{n}_j, \quad (2.13)$$

donde  $\hat{n}_i = |n_i\rangle \langle n_i|$  es el operador número actuando sobre el átomo  $i$ . El término  $\hat{n}_i \hat{n}_j$  toma valor 1 cuando ambos átomos se encuentran en el estado  $|r\rangle$  y 0 en el resto de casos. Por otro lado,  $|\vec{x}_i - \vec{x}_j|$  refleja la distancia entre los átomos  $i$  y  $j$  y  $C_6$  es una constante que viene dada por la naturaleza de los estados involucrados. En este caso, toma un valor de  $5,42 \cdot 10^{-24}$  rad m<sup>6</sup>/s para el estado  $|r\rangle = |70S_{1/2}\rangle$  del <sup>87</sup>Rb [34, 36, 44].

En la Ecuación (2.13), observamos que la intensidad de la interacción depende de la sexta potencia de la distancia. Es este aumento tan abrupto de la energía de interacción lo que bloquea que ambos átomos se encuentren excitados a la vez. A partir de una distancia, la energía de esto se vuelve tan grande que no es favorable energéticamente. La distancia a la que dos vecinos se impiden la excitación simultánea



**Figura 2.7:** Niveles de energía del estado base  $|gg\rangle$ , una sola excitación de Rydberg  $|rg\rangle$  o  $|gr\rangle$  y la doble excitación de Rydberg  $|rr\rangle$ . Esta imagen ha sido extraída de [43].

de manera efectiva se conoce como el radio de bloqueo  $R_b$  [43]. Esta distancia se define como la distancia a la que la energía de interacción de Rydberg es igual a la frecuencia de Rabi

$$\frac{C_6}{R_b^6} = \Omega. \quad (2.14)$$

En la práctica, no podemos considerar que dos átomos no van a estar excitados de manera simultánea para distancias menores que  $R_b$ , ya que esta distancia es simplemente al distancia a partir de la cual la energía de la interacción de Rydberg comienza a ser mayor que el término de Rabi. Para una mejor aproximación, es necesario que los átomos se encuentren a una distancia aún menor.

Para ver cómo de fuerte es esta interacción, supongamos un caso típico en este trabajo. Fijemos una distancia entre átomos de  $5\text{ }\mu\text{m}$ . El potencial de interacción entre ellos es de  $\approx 347 \cdot 10^6$  rad/s, que es considerablemente mayor que el valor máximo de  $\Omega$  ( $25 \cdot 10^6$  rad/s). Como resultado, la probabilidad de que ambos estados se encuentren excitados simultáneamente es muy baja.

### 2.3.4. Hamiltoniano de Rydberg y su evolución

La dinámica del sistema está determinada por tres elementos fundamentales: la configuración electrónica de los átomos, la interacción dipolar entre ellos y su interacción con los láseres que inciden sobre el sistema. Estas interacciones quedan recogidas en el Hamiltoniano dependiente del tiempo:

$$H(t) = \underbrace{\frac{\Omega(t)}{2} \sum_i e^{i\phi(t)} |g_i\rangle \langle r_i| + e^{-i\phi(t)} |r_i\rangle \langle g_i|}_{\text{Rabi}} - \underbrace{\Delta_{\text{global}}(t) \sum_i \hat{n}_i}_{\text{Detuning global}} \\ - \underbrace{\Delta_{\text{local}}(t) \sum_i w_i \cdot \hat{n}_i}_{\text{Detuning local}} + \underbrace{\sum_{i < j} \frac{C_6}{|\vec{x}_i - \vec{x}_j|^6} \hat{n}_i \hat{n}_j}_{\text{Interacción de Rydberg}}, \quad (2.15)$$

donde  $\hat{n}_i = |r_i\rangle \langle r_i|$  es el operador número que toma valor 1 si el átomo  $i$  se encuentra en el estado excitado  $|r\rangle$ , y 0 si está en el estado fundamental  $|g\rangle$ .

Cada uno de los términos del Hamiltoniano tiene una función concreta. El término de Rabi induce transiciones coherentes entre los estados  $|g\rangle$  y  $|r\rangle$ , permitiendo la superposición. El detuning global penaliza energéticamente los estados excitados de forma uniforme, mientras que el detuning local introduce penalizaciones específicas por átomo, moduladas por el peso  $w_i$ . Finalmente, la interacción de Rydberg introduce una repulsión entre átomos excitados próximos, reflejando el bloqueo de Rydberg.

Al definir los parámetros  $\Omega(t)$ ,  $\phi(t)$ ,  $\Delta_{\text{global}}(t)$  y  $\Delta_{\text{local}}(t)$ , se controla la evolución del sistema. Para que el estado medido al final de la evolución coincida con la solución del problema MWIS, deben cumplirse dos condiciones:

- El Hamiltoniano final debe codificar correctamente la función de coste del MWIS.

- El sistema debe encontrarse en el estado fundamental de ese Hamiltoniano.

Para garantizar esto, se diseña un protocolo de evolución dividido conceptualmente en tres fases. Aunque se presenten por separado, la transición entre ellas es suave y continua.

### **Fase 1: Preparación del estado inicial**

Para que el Teorema de la Adiabaticidad sea aplicable, el sistema debe comenzar en el estado fundamental del Hamiltoniano inicial. Este estado puede elegirse libremente, pero por simplicidad se escoge el estado en el que todos los átomos están en su nivel fundamental:  $|g\rangle^{\otimes N}$ .

Reparando a la Ecuación (2.15), se pueden deducir cuales son las condiciones necesarias para ello. El término de Rabi debe estar desactivado, ya que su activación generaría transiciones entre  $|g\rangle$  y  $|r\rangle$ . El *detuning* local también debe ser nulo para no favorecer ningún átomo en particular. El *detuning* global debe ser negativo, de modo que los estados excitados sean penalizados. La interacción de Rydberg está siempre activa, pero en esta fase no contribuye energéticamente ya que todos los átomos están en el estado  $|g\rangle$ .

Por tanto, las condiciones que el Hamiltoniano debe cumplir al inicio de la evolución son:

$$\Omega(0) = 0, \Delta_{\text{local}}(0) = 0 \text{ y } \Delta_{\text{global}}(0) < 0.$$

### **Fase 2: Exploración del espacio de estados**

En esta fase el objetivo es que el sistema explore el espacio de posibles soluciones. Es decir, que cualquier combinación de estados  $|g\rangle$  y  $|r\rangle$  sea accesible. Para ello, el término de Rabi se activa gradualmente hasta llegar a su valor máximo, induciendo una superposición de estados. De forma simultánea, se modifica el detuning global para reducir su valor absoluto y acercarlo a cero, eliminando la penalización general a la excitación. Por otro lado, el detuning local comienza a aumentar, introduciendo diferencias de energía que guiarán el sistema hacia la solución del problema.

En resumen, esta fase permite que el sistema explore el espacio de soluciones posibles, favoreciendo aquellas configuraciones que, según el Hamiltoniano, presentan menor energía.

### **Fase 3: Codificación de la función de coste**

En esta última fase el objetivo es llegar a un Hamiltoniano que simule la función de coste del MWIS, de manera que por el Teorema de la Adiabaticidad, el sistema sigue en el estado fundamental y el resultado que se mida corresponderá con la solución del MWIS. Para ello, hacemos una analogía entre el Hamiltoniano y la función de coste de un problema tipo QUBO (Ecuación (2.2)). Vemos que el primer término equivale al término de *detuning* local y el segundo con el término de interacción de Rydberg,

obteniendo las siguientes equivalencias

$$\sum_i Q_{ii} \longleftrightarrow \sum_i \Delta_{\text{local}}(t) \omega_i \quad (2.16)$$

$$\sum_{i < j} Q_{ij} \longleftrightarrow \sum_{i < j} \frac{C_6}{|\vec{x}_i - \vec{x}_j|^6}. \quad (2.17)$$

Por tanto, las condiciones que el Hamiltoniano debe cumplir son:

$$\Omega(t_{\text{fin}}) = 0, \Delta_{\text{local}}(t_{\text{fin}}) > 0 \text{ y } \Delta_{\text{global}}(t_{\text{fin}}) = 0.$$

En resumen, diseñar una evolución efectiva del Hamiltoniano requiere equilibrar dos objetivos fundamentales: permitir que el sistema explore el espacio de configuraciones mediante la superposición cuántica, y guiarlo suavemente hacia un Hamiltoniano final que represente la función de coste del problema. Todo ello debe hacerse respetando las condiciones del Teorema de la Adiabaticidad, de forma que el sistema permanezca en su estado fundamental durante toda la evolución y, al finalizar, se obtenga la solución óptima del MWIS con alta probabilidad.



---

### 3. Estado del arte

---

Los algoritmos de computación cuántica son comúnmente utilizados para abordar problemas de optimización combinatoria (*Combinatorial Optimization Problems*, COP) debido a la complejidad que pueden mostrar a la hora de buscar el mínimo global. Esta complejidad proviene del aumento de posibles soluciones con el tamaño del problema. Algunos ejemplos de este tipo de problemas son el Problema del Vendedor Viajante, Max-Cut o el MIS, los cuales pueden ser computacionalmente inabarcables para un ordenador clásico cuando las instancias se vuelven muy grandes.

#### 3.1. Algoritmos cuánticos para resolver problemas de optimización combinatoria

Actualmente, nos encontramos en una revolución cuántica, donde los científicos aplican los conocimientos teóricos de la mecánica cuántica para en el desarrollo de tecnologías prácticas [1]. Aunque la idea de los ordenadores cuánticos surgió inicialmente en los años ochenta, desde entonces se han propuesto numerosos algoritmos cuánticos que, en teoría, superan a sus contrapartes clásicas en ciertos tipos de problemas específicos. Sin embargo, para implementar eficazmente estos algoritmos se requieren ordenadores cuánticos avanzados, con un número suficiente de cíbits y bajos niveles de ruido [45].

La realidad actual es que, estos requisitos técnicos aún no son completamente alcanzables debido a las limitaciones prácticas en términos de escalabilidad y control del ruido. Por este motivo, la computación cuántica actual se encuentra en la denominada era NISQ (*Noisy Intermediate-Scale Quantum*). El término “escala intermedia” se refiere a sistemas que poseen desde alrededor de 50 hasta unos pocos cientos de cíbits (en circuitos cuánticos), mientras que “ruidosa” refleja la dificultad para controlar adecuadamente estos cíbits y evitar errores que impactan negativamente en el rendimiento de los algoritmos [46].

Precisamente en este contexto NISQ han surgido los algoritmos híbridos cuánticos-clásicos. Estos algoritmos combinan el uso de procesadores cuánticos, que generan estados cuánticos de múltiples cíbits, con ordenadores clásicos que analizan los resultados medidos y proponen ajustes en la preparación de los estados. Este procedimiento iterativo continúa hasta que se alcanza una solución aproximada al problema de interés [46].

Dependiendo del objetivo específico del problema, estos algoritmos híbridos pueden clasificarse en dos grandes familias. Cuando el objetivo es encontrar el estado de menor energía (estado fundamental) de un Hamiltoniano, estos algoritmos se conocen como Solucionadores de Autovalores Cuánticos Variacionales (*Variational Quantum Eigensolver*, VQE) [45]. Por otro lado, cuando estos métodos se emplean para resolver problemas clásicos de optimización combinatoria, reciben el nombre de Algoritmos de Optimización de Aproximación Cuántica (*Quantum Approximate Optimization Algorithm*, QAOA) [47].

Cabe destacar que estos algoritmos híbridos están diseñados específicamente para ordenadores cuánticos digitales, los cuales operan mediante puertas lógicas cuánticas, de manera similar a como operan los ordenadores clásicos con puertas lógicas binarias. No obstante, en este trabajo se ha hecho uso de un simulador de ordenador cuántico analógico. En estos sistemas, el equivalente a los algoritmos QAOA son los métodos conocidos como *Quantum Annealing* (QA).

Los algoritmos de QA se basan en el Teorema de la Adiabaticidad, el cual garantiza que, si un sistema evoluciona de manera suficientemente lenta (adiabáticamente) desde el estado fundamental de un Hamiltoniano inicial hacia un Hamiltoniano final, el estado del sistema permanecerá en el estado fundamental en todo momento. Así, se consigue alcanzar la solución deseada del problema de optimización.

Mientras que los simuladores cuánticos digitales se muestran como una tecnología prometedora, la implementación real de algoritmos cuánticos en ellos sigue siendo un reto a día de hoy [19]. Es por eso que, tecnologías como los simuladores cuánticos analógicos pueden ser una solución en esta era NISQ, ya que son más tolerantes a ciertos tipos de ruido [33] y pueden operar a temperatura de ambiente.

En su artículo, Pelofske *et al.* (2023) llevaron a cabo un estudio comparativo entre un algoritmo de QA implementado en el *hardware* de D-Wave y un algoritmo de QAOA implementado en *hardware* de IBMQ [48]. En este, los algoritmos se pusieron a prueba con diferentes instancias de Hamiltonianos Ising generados aleatoriamente. En las 10 instancias que estudiaron, QA mostró un mayor rendimiento que QAOA con algoritmos de baja profundidad.

Por otro lado, Mazumder *et al.* (2024) realizaron una comparación entre algoritmos basados en QA y QAOA para resolver instancias del *Number Partitioning Problem* (NPP), conocido por ser NP-duro [49]. En dicho estudio, implementaron diferentes algoritmos QAOA utilizando diversos optimizadores metaheurísticos para encontrar los parámetros óptimos. Los resultados mostraron que el método QA fue superior a todas las variantes de QAOA analizadas, tanto en términos de tiempo de ejecución como en rendimiento general. Los autores sugieren que QAOA podría llegar a competir en rendimiento con algoritmos de optimización específicos para ciertos problemas, aunque los resultados obtenidos por ellos reflejan claramente la ventaja actual de QA frente a QAOA.

No obstante, a día de hoy no se sabe cual de las dos tecnologías será escalable en el futuro [48]. Cada una presenta ventajas y retos que los investigadores deben enfrentarse. Los algoritmos de simuladores cuánticos analógicos son capaces de funcionar a pesar de la presencia de ruido (bajo cierta tolerancia) [33].

En definitiva, actualmente existen diversos algoritmos cuánticos para la resolución de problemas de optimización combinatoria, cada una siendo específica del *hardware* en la que se implementa. Además, cada uno de estos sistemas puede tener diferentes arquitecturas, como es el caso de los superconductores. Como se comentará en la siguiente sección, D-Wave posee diferentes sistemas semiconductores, cada una con una topología diferente, por lo que el mapeado del problema al *hardware* real es específico del mismo.

A día de hoy, todavía resulta imprescindible adaptar los algoritmos y protocolos específicamente a cada tecnología y arquitectura concreta, ya que no existe una solución

genérica universalmente aplicable. Lo que es efectivo para una plataforma puede no ser fácilmente trasladable a otra, o puede requerir modificaciones. Por tanto, resulta necesario seleccionar la tecnología y topología que mejor se ajusten al problema que se estudie.

## 3.2. Comparación de tecnologías para *Quantum Annealing*

La implementación de algoritmos de QA se ha explorado en distintas plataformas físicas, cada una con características específicas que afectan su rendimiento, escalabilidad y aplicabilidad en la resolución de problemas de optimización combinatoria.

D-Wave ha sido pionera en la comercialización de ordenadores cuánticos basados en técnicas QA, utilizando cúbits superconductores organizados en arquitecturas topológicas como Chimera, Pegasus y la futura Zephyr. Su Unidad de Procesamiento Cuántico (*Quantum Processing Unit*, QPU) más avanzada, Advantage, alcanza los 5760 cúbits conectados mediante la topología Pegasus [50]. Esta tecnología está específicamente orientada a resolver problemas de tipo QUBO o Ising, mostrando buen rendimiento en problemas binarios cuadráticos y, más recientemente, también en algunos casos de programación lineal entera mixta mediante flujos híbridos. Sin embargo, la necesidad de embebido debido a su topología fija, junto con limitaciones en el control individual de cúbits, afecta la calidad de las soluciones y la escalabilidad [50]. Además, el *hardware* está expuesto a diversas fuentes de error como el ruido térmico y la decoherencia [50].

Las trampas de iones representan una de las plataformas más maduras para computación cuántica universal. Destacan por ofrecer fidelidades superiores al 99.9 % en puertas de dos cúbits y tiempos de coherencia extremadamente largos, alcanzando más de 50 segundos, gracias al aislamiento de los iones en vacío [51]. Sin embargo, enfrentan retos materiales importantes para su escalado, como el ruido eléctrico superficial en los electrodos, que puede calentar el movimiento iónico y dificultar las operaciones entre varios cúbits. A futuro, se espera que técnicas como la miniaturización basada en CMOS, la integración fotónica y tratamientos de superficie logren mitigar estos problemas y permitan una mayor escalabilidad sin comprometer la fidelidad de operación [51].

Por su parte, los ordenadores cuánticos basados en átomos neutros se perfilan como una alternativa emergente con gran potencial. Esta tecnología ofrece buena escalabilidad, tiempos de coherencia largos y conectividad intermedia y reconfigurable. Una ventaja destacable es que los cúbits son físicamente idénticos, eliminando la necesidad de calibración individual y reduciendo la heterogeneidad en el ruido [2]. Se han alcanzado fidelidades en puertas de dos cúbits superiores al 99.5 %, superando los umbrales requeridos para corrección de errores cuánticos [11]. No obstante, su principal limitación reside en los largos tiempos de preparación y la pérdida de cúbits tras cada medida, lo que obliga a reiniciar todo el sistema. Aunque actualmente esta plataforma es menos madura que las de iones atrapados o cúbits superconductores, se espera que en los próximos años avances en miniaturización, mejoras en fidelidad y la integración

en entornos locales consoliden su posición en el campo de la computación cuántica aplicada a optimización [2].

### 3.3. *Quantum Annealing* para la resolución del problema MIS/MWIS

Diversos trabajos han explorado el uso de algoritmos de QA para abordar el problema del conjunto independiente máximo (MIS) o su versión ponderada (MWIS). Este enfoque aprovecha la capacidad de los sistemas cuánticos para explorar de forma paralela el espacio de soluciones de problemas combinatorios difíciles.

Pichler *et al.* (2018) fueron pioneros en utilizar el bloqueo de Rydberg para explorar problemas conocidos por ser NP-duros como UD-MIS [19], mostrando el potencial de estas tecnologías para la resolución de problemas de optimización combinatoria.

Por otro lado, Finžgar *et al.* (2024) observaron que los protocolos lineales estándar en QA pueden atravesar transiciones de fase que reducen drásticamente la fidelidad del resultado [52]. Para mitigar este efecto, proponen parametrizaciones flexibles del protocolo de *annealing*, optimizadas mediante un algoritmo de optimización bayesiana, con el objetivo de resolver instancias del MIS de forma más eficiente. En su trabajo, utilizan como métrica principal la *Success Probability*, una medida que, como se discutirá más adelante, puede ser demasiado restrictiva e introducir sesgos. También señalan la lentitud en el proceso de lectura del resultado en el *hardware* cuántico, lo que justifica el uso de optimizadores inteligentes que minimicen el número de evaluaciones necesarias.

Por último, el rendimiento de la optimización se analiza en función de una métrica llamada *Hardness Parameter* (HP), introducida por Ebadi *et al.* (2020), y que será de gran relevancia en este trabajo [28]. Tanto Finžgar *et al.* como Ebadi *et al.* encuentran una correlación entre este parámetro y la dificultad del problema, observando que las instancias en las que la energía del estado fundamental es cercano a la de otras configuraciones, suelen ser más difíciles de resolver.

En el enfoque que se presenta en este trabajo, la métrica que se utilizará para evaluar los resultados será el *Approximation Ratio*, ya que puede representar los resultados de una manera fiel, al tratarse del promedio de todas las repeticiones de la simulación realizadas. Asimismo, la optimización se realiza mediante un optimizador global y otro local, comparando los resultados tanto en rendimiento como en tiempo de ejecución. Además, el parámetro HP no es extrapolable al problema ponderado del MIS, por lo que se propone una nueva definición para este problema. Como se ha mencionado, varios autores discuten la relación entre una disminución del rendimiento y algunas propiedades de los grafos, por lo que en este trabajo también se ahondará en este tema.

---

# 4. Uso del simulador

---

El simulador utilizado en este trabajo se encuentra disponible a través de [Amazon Braket](#)<sup>1</sup>, un servicio de AWS que permite a los usuarios ejecutar tareas en distintos ordenadores y simuladores cuánticos.

En este capítulo se introducen los conceptos esenciales para comprender cómo funciona el simulador y cómo se interpretan sus resultados. Además, mediante algunos ejemplos, se ilustran los resultados y métricas más relevantes del desarrollo del proyecto. Para una descripción más detallada, se recomienda consultar el repositorio [GitHub](#)<sup>2</sup> de la librería.

## 4.1. Conceptos clave

Para realizar una simulación, es necesario definir tres elementos principales: el dispositivo cuántico, la evolución del Hamiltoniano y el grafo que representa el problema.

**Grafo:** Los grafos se han generado utilizando la librería [NetworkX](#)<sup>3</sup>. Esta librería permite generar e ilustrar los grafos de una manera muy sencilla, además de poder extraer propiedades que serán de utilidad más adelante. No obstante, para ejecutar la simulación, únicamente es necesario definir un objeto `AtomArrangement()`, al cual se le van añadiendo los nodos de manera secuencial mediante una tupla (en formato  $(x, y)$ ) con las coordenadas del átomo.

**Dispositivo:** El dispositivo es el ordenador cuántico o simulador en el que se ejecuta la tarea. A través de la librería, es posible acceder a servicios en la nube y lanzar tareas a ordenadores cuánticos reales. Sin embargo, los *hardware* cuánticos son poco accesibles por su alto coste, por lo que los análisis iniciales se realizan en simuladores. En este trabajo se ha utilizado el simulador ‘braket\_ahs’, que se ejecuta localmente en el entorno del usuario. Este simulador emula el comportamiento del ordenador cuántico de átomos neutros de Aquila descrito en la Sección 2.3.1. Para especificarlo, es necesario definir el objeto `LocalSimulator()`, al que se le introduce como parámetro el nombre del dispositivo deseado.

**Evolución del Hamiltoniano:** Al seleccionar el dispositivo AHS, se asume que el sistema evoluciona bajo el Hamiltoniano definido en la Ecuación (2.15). Esta evolución está gobernada por un conjunto de parámetros que varían en el tiempo:  $\Omega(t)$ ,  $\Delta_{\text{global}}(t)$ ,  $\Delta_{\text{local}}(t)$  y  $\phi(t)$ . Este conjunto de funciones hace referencia a los *drivings*, ya que son los que producen y dirigen la evolución dinámica del sistema. En física cuántica, el término *driving* se utiliza comúnmente para describir un campo externo que induce o controla la dinámica del sistema.

---

<sup>1</sup><https://docs.aws.amazon.com;braket/latest/developerguide/what-is-braket.html>

<sup>2</sup><https://github.com/amazon-braket/amazon-braket-examples/tree/main>

<sup>3</sup><https://networkx.org/>

Para definir cada uno de ellos, es necesario hacerlo mediante el objeto `TimeSeries()` de manera individual, y después encapsular  $\Omega(t)$ ,  $\phi(t)$  y  $\Delta_{\text{global}}(t)$  en un objeto `DrivingField()`. En el caso de que el usuario desee hacer uso del *detuning* local, este se define directamente como un objeto `LocalDetuning()` cuyos parámetros son dos listas que contienen la evolución de este, y una lista que se conoce como *pattern*. El *pattern* hace referencia al peso que cada nodo tiene asociado, por lo que tendrá una longitud igual al número de nodos del grafo.

Finalmente, se construye el hamiltoniano mediante el objeto `AnalogHamiltonianSimulation()`, cuyos parámetros son los *drivings* que controlan la evolución (objetos `DrivingField()` y `LocalDetuning()`) y la disposición de átomos (objeto `AtomArrangement()`).

Aunque los protocolos de *drivings* pueden diseñarse libremente, deben cumplir ciertas condiciones para que la simulación sea válida. En este trabajo se ha optado por parametrizar su evolución mediante funciones concretas:

- $\Omega(t)$  : Se define como una función suave que depende de dos parámetros: el valor máximo de la curva ( $\Omega_{\text{max}}$ ) y la duración total de la evolución ( $t_{\text{max}}$ ):

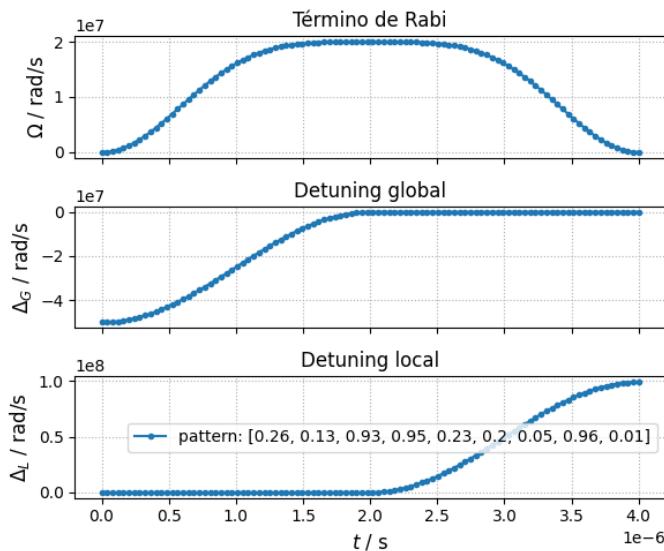
$$\Omega(t) = \Omega_{\text{max}} \sin^2 \left( \frac{\pi}{2} \sin \left( \pi \cdot \frac{t}{t_{\text{max}}} \right) \right) \quad (4.1)$$

- $\Delta_{\text{global}}(t)$ : Comienza con un valor constante mínimo ( $-\Delta_{\text{global}_{\text{max}}}$ ) y realiza una transición de tipo seno hasta alcanzar el valor cero. Esta transición se controla mediante dos parámetros:  $a$ , que define su duración, y  $b$ , que define el instante central:

$$\Delta_{\text{global}}(t) = \begin{cases} -\Delta_{\text{global}_{\text{max}}} & \text{si } t < b - a/2 \\ -\frac{\Delta_{\text{global}_{\text{max}}}}{2} + \frac{\Delta_{\text{global}_{\text{max}}}}{2} \cdot \sin \left( \frac{\pi}{a(t-b)} \right) & \text{si } b - a/2 \leq t \leq b + a/2 \\ 0 & \text{si } t > b + a/2 \end{cases} \quad (4.2)$$

- $\phi(t)$ : Como se ha introducido en la Sección 2.3.4, este parámetro se ha fijado en 0 y por tanto, no se volverá a hacer referencia a este parámetro a lo largo del trabajo.
- $\Delta_{\text{local}}(t)$  : Tiene la misma forma funcional que  $\Delta_{\text{global}}(t)$ , con la diferencia de que comienza en cero y realiza la transición hacia un valor máximo. En la figura inferior de la Figura 4.1, el patrón mostrado representa el peso (entre 0 y 1) asignado a cada nodo. Este se utiliza para modular individualmente la intensidad del *detuning* aplicado a cada átomo.

A menos que se indique lo contrario, los *drivings* se parametrizan con cuatro valores: los parámetros de transición del *detuning* global ( $a_{\text{global}}, b_{\text{global}}$ ) y los del *detuning* local ( $a_{\text{local}}, b_{\text{local}}$ ). Así, el resto de parámetros se han dejado fijos. Esto se debe a que añadir más parámetros aumentaría significativamente el coste computacional de los procesos de optimización. Por lo tanto, el protocolo completo queda definido por la lista de parámetros  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}]$ .



**Figura 4.1:** Ejemplo de evolución temporal de los *drivings* empleados. Desde la imagen superior a la inferior se representan las funciones  $\Omega(t)$ ,  $\Delta_{\text{global}}(t)$  y  $\Delta_{\text{local}}(t)$ . El *pattern* que aparece en el *detuning* local indica el peso asignado a cada nodo (entre 0 y 1), utilizado para modular la intensidad del mismo. La configuración mostrada corresponde a los parámetros  $[1/2, 1/4, 1/2, 3/4]$ .

Una vez definidos estos tres elementos, el sistema está listo para ejecutar la simulación y analizar su comportamiento bajo diferentes condiciones.

## 4.2. Ejemplo de uso y definición de métricas

Para ilustrar de forma más clara el funcionamiento del simulador, los resultados que se obtienen y las métricas empleadas, vamos a analizar un ejemplo concreto. Si el lector lo desea, puede seguir esta sección paralelamente al *notebook* `Ejemplo.ipynb` de la carpeta ‘Uso del simulador’, donde se muestra el código utilizado para obtener estos resultados.

El grafo seleccionado para este ejemplo es el mismo que aparece en la Figura 2.2, y los *drivings* empleados se muestran en la Figura 4.1. Los parámetros utilizados son  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}] = [1/2, 1/4, 1/2, 3/4]$ , ya que, de forma experimental, se observó que ofrecían buenos resultados en distintos grafos. La duración de la evolución es de  $4 \mu\text{s}$ , que supone la duración máxima permitida por el simulador. Se ha escogido este valor para facilitar transiciones más suaves y adiabáticas.

A la hora de ejecutar la simulación, el usuario puede configurar diversos hiperparámetros para tener mayor control sobre el experimento. En este trabajo, se ha definido la siguiente configuración y se ha mantenido fija a lo largo del proyecto:

- **shots:** número de repeticiones de la simulación. Cuando mayor es el número de repeticiones, mayor será la precisión estadística, pero también aumentará el coste computacional. Se ha utilizado un valor de 20.000, que proporciona resultados estables.

- **steps**: número de pasos intermedios en la evolución. Si los *drivings* son abruptos, se recomienda aumentar este valor para mejorar la precisión de la integración. En este caso, se ha fijado en 1.000.
- **blockade\_radius**: establece la distancia mínima permitida entre dos átomos excitados. Este parámetro aplica la condición de bloqueo de Rydberg de forma anticipada, descartando configuraciones inválidas antes de simular, lo que reduce considerablemente el tiempo de cálculo en instancias grandes.

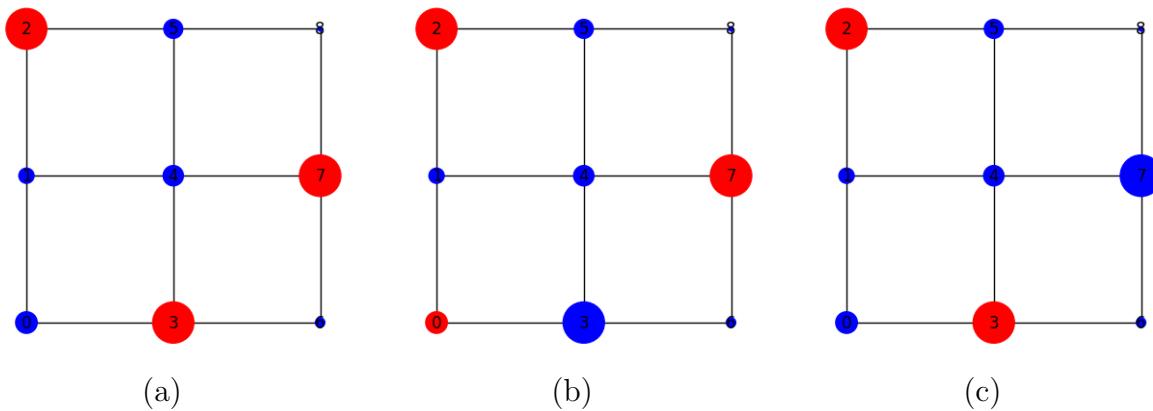
Al lanzar la simulación, se genera una evolución temporal que se repite tantas veces como se indique con **shots**. En cada repetición, se mide el estado del sistema y se guarda en una cadena binaria denominada *bitstring*, formada por ‘0’ y ‘1’ (o ‘g’ y ‘r’), que indica el estado medido de cada átomo. En este ejemplo, al haber 9 nodos, cada *bitstring* tiene una longitud de 9 elementos. El resultado final es un diccionario que asocia a cada *bitstring* su frecuencia de aparición, el cual puede representarse en un DataFrame como el mostrado en la Figura 4.2.

Para este ejemplo, se han utilizado valores reducidos de **shots** = 1000 y **steps** = 30 con fines ilustrativos.

Los resultados pueden representarse gráficamente de distintas maneras. En la Figura 4.3 se muestra el estado más probable tras la simulación, que coincide con la solución del MWIS. Sin embargo, el segundo estado más frecuente tiene una probabilidad similar, y aunque es subóptimo, difiere solo en un nodo con respecto al óptimo.

bitstring	Counts
16 ggrrgggrg	226
26 rgrrgggrg	213
14 ggrrggggg	162
10 ggrrgggrg	156
15 ggrrgggr	53

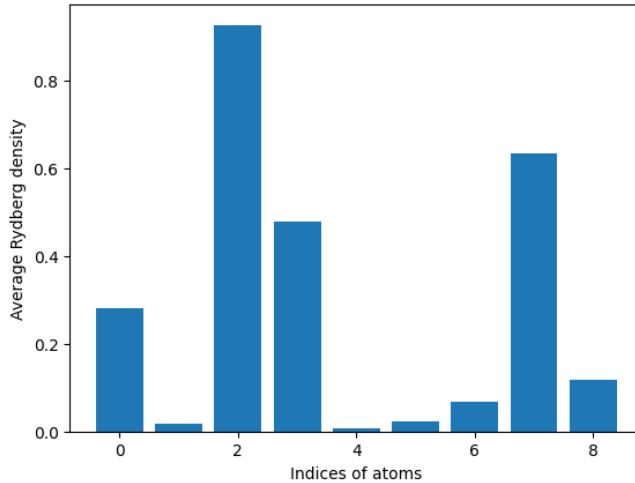
**Figura 4.2:** Primeras cinco instancias del DataFrame con los resultados de la simulación, ordenadas de forma descendente por la variable ‘Counts’.



**Figura 4.3:** Ilustración del *bitstring* (a) más frecuente, (b) segundo más frecuente y (c) tercero más frecuente. Se han coloreado en rojo los átomos que se han medido en el estado excitado (‘r’) y en azul en el estado fundamental (‘g’).

También podemos analizar los resultados a nivel individual para cada átomo, observando con qué frecuencia ha sido medido en el estado excitado. Esto se representa

con la densidad promedio de Rydberg, una métrica que indica el porcentaje de veces que un átomo ha aparecido en estado ‘r’ a lo largo de la simulación.



**Figura 4.4:** Diagrama de barras de la densidad de Rydberg promedio de cada átomo.

La Figura 4.4 ilustra que los átomos con mayor densidad de Rydberg corresponden a los que forman parte de la solución del MWIS. Sin embargo, en un caso ideal, la densidad de estos átomos sería 1 y 0 para el resto. En este ejemplo, la solución óptima solo aparece en un 22,6 % de las repeticiones, siendo apenas un 6,1 % más probable que el siguiente mejor estado.

### Métricas para evaluar los resultados

Para comparar simulaciones entre sí, es necesario definir métricas que reflejen qué tan cerca están los resultados obtenidos de la solución óptima. En este trabajo se utilizan dos métricas principales:

**Success Probability (SP):** probabilidad de que el estado medido coincida exactamente con la solución óptima. Es ampliamente utilizada en este contexto [19, 52, 53] y se define como:

$$SP = \frac{N_{\text{éxito}}}{N_{\text{shots}}}, \quad (4.3)$$

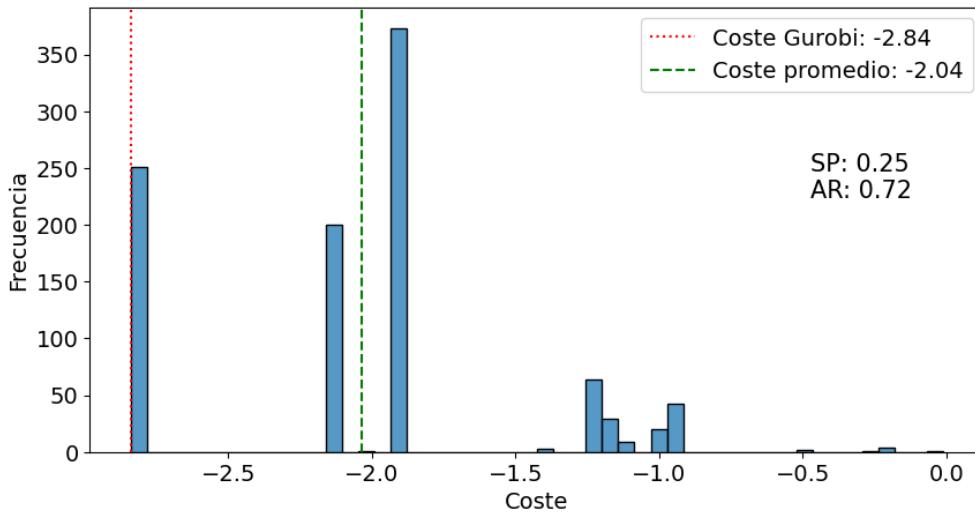
donde  $N_{\text{shots}}$  es el número de veces que se ha repetido la simulación y  $N_{\text{éxito}}$  es el número de veces que se ha medido la solución correcta.

**Approximation Ratio (AR):** cociente entre el valor promedio de la función de coste en los resultados medidos y el valor de la solución óptima [53, 54]:

$$AR = \frac{\overline{C_{\text{obt}}}}{C_{\text{sol}}}, \quad (4.4)$$

donde  $\overline{C_{\text{obt}}}$  es el promedio de la función de coste evaluada sobre el resultado obtenido en cada *shot* y  $C_{\text{sol}}$  es el coste de la solución óptima.

Así pues, evaluando estas métricas en los resultados del ejemplo, obtenemos  $SP=0,226$  y  $AR=0,711$ . En la Figura 4.5 se muestra el histograma obtenido al evaluar cada *bitstring* medido experimentalmente en la función de coste. La línea roja punteada muestra el coste de la solución óptima que se ha obtenido mediante el algoritmo de Gurobi. Asimismo, la línea intermitente verde indica el valor de la función de coste promedio  $\overline{C}_{\text{opt}}$ . En el caso ideal, la distribución se concentraría en el *bitstring* óptimo, de manera que la línea verde y roja se superpondrían y tanto el AR como el SP tomarían un valor de 1.



**Figura 4.5:** Histograma del coste de cada *bitstring* medido.

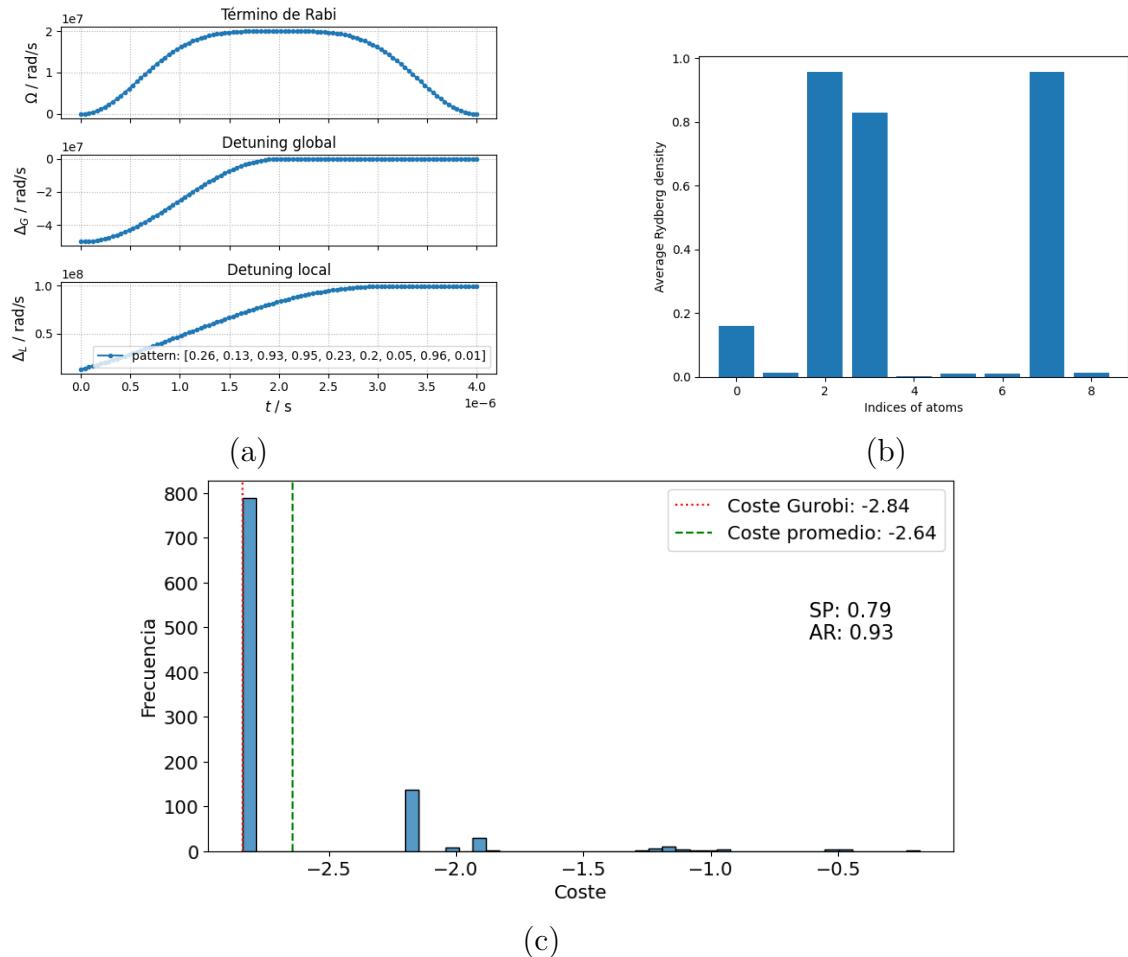
### Optimización de los *drivings*

A partir de los resultados obtenidos en esta primera simulación, podemos ver que la elección de los *drivings* influye en cómo evoluciona el sistema y, en consecuencia, en la calidad de la solución final. Esto nos lleva a plantear la posibilidad de optimizar los *drivings* con el objetivo de mejorar métricas como el AR. Ajustando adecuadamente los parámetros que definen los *drivings*, es posible favorecer una evolución más adiabática y aumentar así la probabilidad de que el sistema alcance el estado fundamental del Hamiltoniano final, que representa la solución óptima del problema.

En este caso, se ha optimizado únicamente el protocolo del *detuning* local por simplicidad, teniendo como objetivo maximizar el AR.

En la Figura 4.6 se muestran los resultados de la optimización del *driving* local. En la Figura 4.6 (a) se observa que parece favorable que la transición se produzca de manera lineal, lo cual podría ayudar a una transición más adiabática al tener variaciones menos bruscas. Sin embargo, como se verá más adelante, los parámetros dependen entre sí, por lo que el protocolo del *detuning* local depende del global y del término de Rabi que fijemos.

Por otro lado, los resultados de la simulación también han mejorado considerablemente. En la Figura 4.6 (b) vemos que las densidades de Rydberg promedio son mayores que 0,8 para los cíbits que forman parte de la solución y casi nulo para el



**Figura 4.6:** Resultados de la simulación tras la optimización. (a) protocolo de *drivings* obtenido tras optimizar el *detuning* local. (b) densidad de Rydberg promedio. (c) histograma del coste de los resultados obtenidos.

resto, con excepción del primer cúbico. En cuanto al histograma de la Figura 4.6 (c), se observa una clara mejoría, ya que la distribución ahora se encuentra concentrada en el estado óptimo.

**Tabla 4.1:** Comparación del AR y SP antes y después de optimizar el *driving* local.

Caso	AR	SP
Sin optimización	0,72	0,25
Con optimización	0,93	0,79

La mejora de los resultados la cuantificamos mediante las métricas comentadas. En la Tabla 4.1 vemos que tanto el AR como SP han mejorado un 25% y 216% respectivamente. Cabe destacar que a pesar de haber optimizado con AR como variable objetivo, es el SP el que ha tenido una mayor mejora. Esto se debe por un lado a que mejorar el AR concentra la distribución en la solución correcta, por lo que es natural que el SP mejore.

Asimismo, el SP es una métrica muy restrictiva y puede introducir sesgos. Esto se debe a que el simulador puede quedarse atrapado en un estado subóptimo, pero con

un coste similar al óptimo. Bajo cierta tolerancia, podríamos llegar a considerar este escenario como un buen resultado. Sin embargo, el valor de SP seguiría siendo cercano a 0.

---

## 5. Resultados y discusión

---

### 5.1. Generación de conjuntos de datos

Para estudiar el comportamiento del sistema y analizar el efecto de los *drivings* sobre los resultados, se ha generado un conjunto de grafos sintéticos sobre los que se han realizado las simulaciones. Estos grafos actúan como instancias del problema MWIS y permiten evaluar la robustez de distintos protocolos de *drivings*.

Cada grafo se construye mediante una cuadrícula bidimensional fija sobre la que colocan los nodos. A continuación, se establecen las conexiones entre nodos. En este caso, al estudiar grafos de discos unitarios, se define un radio fijo y se establece un arco entre todos los nodos que se encuentre a una distancia menor. La distancia escogida conecta únicamente los primeros y segundos vecinos (diagonales).

A partir de esta malla completa se van eliminando nodos de manera aleatoria hasta que el número de nodos coincide con el deseado. De esta manera, conseguimos que los nodos se distribuyan por la malla de una manera controlada pero diversa.

Después, se asigna un peso aleatorio en el rango  $[0, 1]$  a cada nodo, lo cual introduce una variabilidad adicional en las instancias, ya que grafos con la misma topología pueden dar lugar a soluciones distintas en función de los pesos asignados.

Finalmente, existe la posibilidad de que, al eliminar nodos, el grafo resultante se haya separado en dos (grafo inconexo). Estos fueron descartados automáticamente y únicamente se ha trabajado con grafos conexos. Es decir, aquellos en los que existe un camino entre cualquier par de nodos. De cada grafo resultante, se guarda únicamente la semilla necesaria para generarla.

El código de la función que genera cada grafo `Graph_gen()` y el que genera el conjunto de datos (`Graph_seed_gen`) se puede ver en `Generador_grafos_n_nodos.ipynb`. A la hora de definir el conjunto de datos, se ha decidido generar 1000 grafos por cada tamaño de grafo diferente. Esta cantidad permite obtener resultados estadísticamente significativos sin que el coste computacional de las simulaciones sea inasumible.

A pesar de que se hayan generado conjuntos de grafos de entre 7 y 25 nodos, la gran mayoría del trabajo analiza resultados basados en grafos de 7 nodos, ya que realizar estudios intensos sobre grafos más grandes era inabordable. Por tanto, el resto de conjuntos se utilizó únicamente para estudiar la capacidad de generalización de los resultados.

No obstante, esta limitación está relacionada con el uso de simuladores clásicos, que requieren tiempos de ejecución elevados para instancias grandes del problema. Si resolvíramos el problema en un *hardware* cuántico real, el tiempo de ejecución dejaría de ser una restricción significativa. Sin embargo, en este caso, aparecerían diversas fuentes de error inherentes a la tecnología actual. Por tanto, aunque estos algoritmos son prometedores, su implementación en dispositivos reales solo será viable cuando se logren reducir los niveles de ruido y mejorar la fidelidad de las operaciones cuánticas.

## 5.2. Optimización de *drivings*

### 5.2.1. Objetivos de la optimización y desarrollo experimental

El objetivo principal de esta sección es estudiar cómo afectan los protocolos *drivings* en la resolución del problema MWIS mediante simulación cuántica. Como se ha comentado, uno de los aspectos clave que determina el éxito de las simulaciones es la elección de los parámetros que controlan la evolución temporal del sistema. Estos *drivings*, definen la evolución temporal de los términos del Hamiltoniano. Además, es fundamental que esta trayectoria esté cuidadosamente diseñada para que la evolución lleve el sistema al estado fundamental del Hamiltoniano final (el cual codifica la solución óptima del problema). Si los *drivings* no están bien ajustados, la evolución puede no ser suficientemente adiabática y el sistema puede acabar en un estado excitado en lugar del estado fundamental, lo que se traduce en una solución subóptima. Asimismo, resulta necesario definirlos teniendo en cuenta que el Hamiltoniano final debe simular correctamente la función de coste del MWIS.

El objetivo de esta sección es, por tanto, encontrar un conjunto de *drivings* que, sin necesidad de personalización por grafo, permita alcanzar soluciones de alta calidad en un amplio rango de instancias. Para ello, se han estudiado las métricas de *Approximation Ratio* y *Success Probability* descritas recientemente. Aunque ambas métricas son relevantes, en este trabajo se ha escogido el AR como métrica objetivo durante la optimización, por varias razones:

1. Es más estable estadísticamente, ya que se basa en un promedio ponderado de todas las soluciones.
2. Tiene una interpretación práctica clara: indica cuán cerca, en promedio, estamos del estado óptimo.
3. A diferencia del SP, no penaliza fuertemente las soluciones casi óptimas, que pueden ser razonables desde un punto de vista aplicado. Así, el SP puede dar resultados cercanos a 0 si el sistema se estanca en un estado subóptimo, independientemente de lo cerca que se encuentren.

La optimización de los *drivings* se ha centrado en ajustar los parámetros que controlan las transiciones de los términos de *detuning* (local y global), manteniendo fijo el perfil de la amplitud de Rabi para reducir la complejidad computacional. Es decir, los parámetros que se han variado son el conjunto  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}]$  descritos en el Capítulo 4. Asimismo, el rango en el que estos parámetros pueden tomar valores es  $[0, 1]$ , el cual se encuentra normalizado por la duración de la evolución. La elección de este rango se debe a que fuerza a que las transiciones sucedan a lo largo de la evolución pero sigue siendo un rango lo suficientemente pequeño como para no suponer una limitación computacional en el estudio.

En la Tabla 5.1 se muestran los valores escogidos para el resto de parámetros que definen los *drivings*.

**Tabla 5.1:** Valores de los parámetros fijos de los *drivings* utilizados en la optimización.

Parámetro	Valor
$t_{\max}$	$4 \mu s$
$\Omega_{\max}$	$20 \text{ rad}/\mu s$
$\Delta_{\text{global},\max}$	$99,007768 \text{ rad}/\mu s$
$\Delta_{\text{local},\max}$	$99,007768 \text{ rad}/\mu s$

Para escoger estos valores de los parámetros fijos nos hemos basado en el conocimiento previo que se tiene del problema y de los resultados obtenidos previamente. La duración de la evolución se ha definido como la máxima permitida por el sistema, ya que cuanto más larga sea la solución, más suaves serán las transiciones y más fácil será realizar la evolución de manera adiabática.

Como se recomienda en la propia guía del ordenador cuántico de Aquila [34], utilizar valores altos de  $\Omega_{\max}$  ayuda a llegar a la solución deseada, ya que cuanto mayor sea la superposición, mejor se explora el espacio de posibles soluciones. Sin embargo, se ha optado por no escoger el valor máximo ( $25 \text{ rad}/\mu s$ ) para evitar variaciones bruscas. En cuanto a los valores de  $\Delta_{\text{global},\max}$  y  $\Delta_{\text{local},\max}$ , cuyo valor máximo es  $125 \text{ rad}/\mu s$ , se ha aplicado el mismo argumento.

Debido al aumento del coste computacional con el número de nodos de grafo, y a su vez, su complejidad [28], la optimización de grafos se ha realizado únicamente para grafos de 7 nodos. Asimismo, fue necesario buscar algoritmos que no necesitasen realizar cálculos de gradientes y no hicieran falta demasiadas iteraciones. No obstante, una vez tuvimos acceso al servidor [LluisVives](#)<sup>1</sup> del Servicio de Computación Científica de la Universitat de València, pudimos lanzar códigos con mayor coste computacional, por lo que se decidió realizar un *grid search* de los cuatro parámetros, lo cual permitió estudiar la manera en la que varía la función de coste. Más tarde, se optó por realizar otro *grid search* para algunos de los parámetros fijos de la Tabla 5.1.

En todas las optimizaciones realizadas se ha hecho uso de una misma función de coste. Esta, toma como parámetros un DataFrame con las propiedades de los grafos que se van a evaluar (número de nodos, tamaño de la malla y semilla que lo genera) y los parámetros de los *drivings* que se van a utilizar. Dentro, calcula los *drivings* y evalúa el AR de cada grafo a la vez que guarda estos resultados en una lista. Finalmente, se devuelve el AR promedio.

El primer algoritmo de optimización utilizado ha sido el algoritmo de Optimización Restringida por Aproximación Lineal (*Constrained Optimization BY Linear Approximation*, COBYLA), un algoritmo que no requiere el cálculo de gradientes y ha demostrado ser efectivo incluso en sistemas de hasta 17 cíbits [18]. Al ser un algoritmo que depende del punto inicial que se escoja, la optimización se ha repetido con puntos diferentes para una mayor seguridad del resultado.

Asimismo, se ha probado a realizar una optimización Bayesiana. Este es un algoritmo ampliamente utilizado en casos donde evaluar la función de coste sea difícil

---

<sup>1</sup><https://www.uv.es/uvweb/servicio-informatica/es/lluisvives-1285890326008.html>

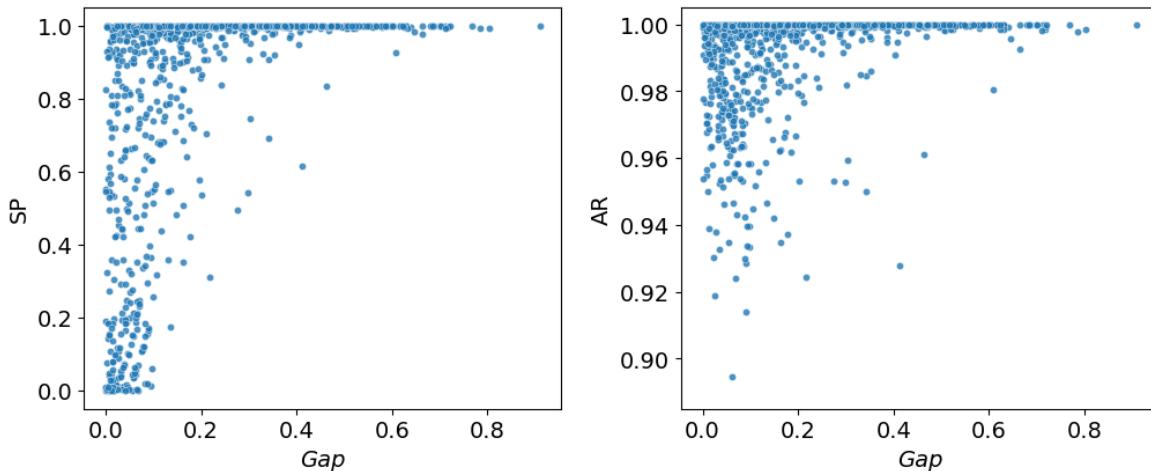
y toma mucho tiempo. Los optimizadores Bayesianos crean un modelo subrogado para cuantificar la incertidumbre del modelo de la función de coste. Después, utilizan esta información para decidir el siguiente punto a evaluar, de manera que hace una búsqueda inteligente, reduciendo el número de evaluaciones necesarias [55]. Asimismo, no necesita hacer cálculos de derivadas, lo cual ayuda a reducir el tiempo de cálculo.

En resumen, mediante tres algoritmos diferentes se ha hecho una búsqueda de los *drivings* que sean capaces de proporcionar los mejores resultados para un conjunto de 1000 grafos. Después, se han comparado estos tres métodos y se discuten algunas cuestiones que quedan abiertas tras el análisis de los resultados.

### 5.2.2. Comparación de métodos

Dado que inicialmente no se disponía de resultados previos con los que comparar las diferentes estrategias de optimización, se tomó como referencia el rendimiento obtenido utilizando los *drivings* empleados en el ejemplo de la Sección 4.2, obteniendo así un AR promedio de 0,99222 y un SP promedio de 0,81794. Estos resultados nos servirán como base a la hora de comparar simulaciones.

La Figura 5.1 muestra los resultados de simular los 1000 grafos con dichos *drivings*. Se ha optado por representar las métricas en función de la diferencia entre el valor de coste del estado óptimo y el subóptimo (denominada *gap*), ya que se ha observado experimentalmente una relación significativa entre ambas variables.



**Figura 5.1:** Dispersión de la *Success Probability* (izquierda) y el *Approximation Ratio* (derecha) frente al *gap* entre el estado fundamental y el subóptimo.

En ambos casos se observa que el rendimiento disminuye a medida que el *gap* tiende a cero. Sin embargo, el AR se mantiene por encima de 0.9 en la mayoría de las instancias, mientras que el SP muestra una mayor variabilidad, oscilando entre 0 y 1 en los casos con menor *gap*.

Aunque este comportamiento se analizará con más detalle más adelante, la Figura 5.1 ya pone de manifiesto la naturaleza restrictiva del SP. Si el sistema se mide en un estado subóptimo muy próximo, el SP penaliza severamente este resultado asignándole

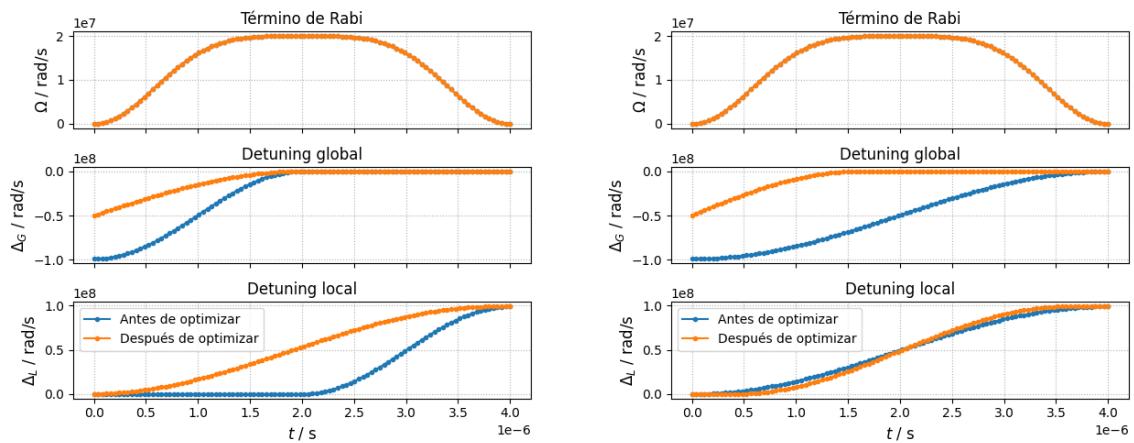
un valor nulo. En cambio, si el coste del estado observado es similar al óptimo, el AR permanece cercano a 1, reflejando con mayor fidelidad la calidad de la solución.

## Optimización COBYLA

La optimización se ha llevado a cabo utilizando el algoritmo COBYLA, disponible en la librería `scipy`<sup>2</sup>. Para ello, se han definido una serie de hiperparámetros. Aunque se ha fijado el rango permitido de los parámetros entre  $[0, 1]$ , el optimizador puede evaluar puntos fuera de este intervalo durante el proceso. No obstante, el resultado final debe encontrarse dentro del dominio especificado. Además, se ha establecido un número máximo de iteraciones igual a 100, con el fin de evitar que el algoritmo quede atrapado sin converger. Al finalizar cada optimización, se indica si esta ha sido exitosa o si se ha alcanzado el máximo de iteraciones, permitiendo así valorar la fiabilidad de la solución obtenida.

Dado el elevado coste computacional, no se optimizaron los 1000 grafos del conjunto completo. En su lugar, se seleccionaron aquellos que, utilizando los *drivings* mostrados en la Figura 5.1, habían obtenido un valor de AR inferior a 0,99. Esto dio lugar a un subconjunto de 223 grafos sobre los que se aplicó la optimización.

Puesto que COBYLA es un algoritmo sensible a las condiciones iniciales, se decidió realizar una segunda ejecución con un punto de partida diferente. La Figura 5.2 compara los *drivings* óptimos obtenidos a partir de ambas inicializaciones. Como se puede observar, los resultados son prácticamente idénticos. Por tanto, se decidió utilizar uno de estos conjuntos de parámetros, concretamente aquel que alcanzó el mayor AR promedio, para continuar el análisis.

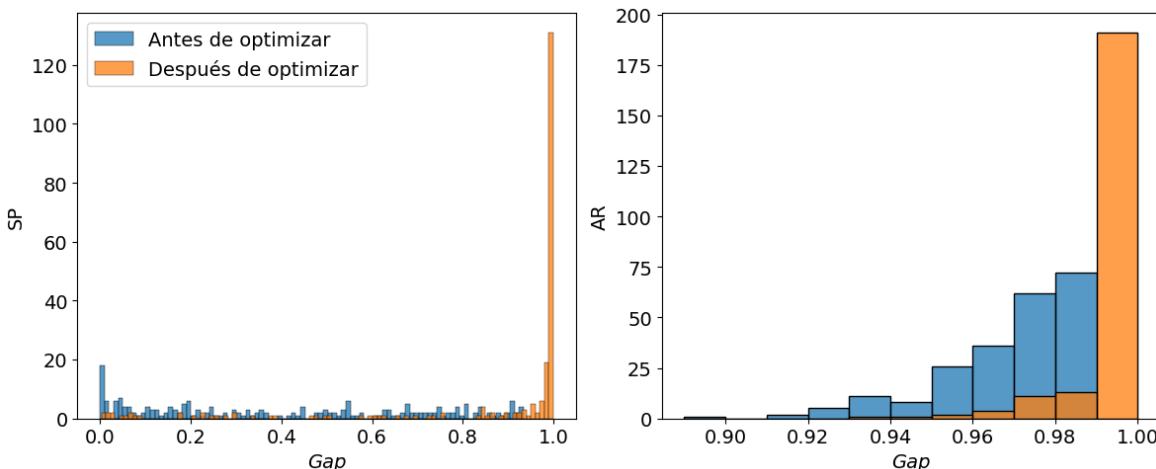


**Figura 5.2:** Comparación de los resultados de la optimización por COBYLA realizada en dos puntos iniciales diferentes. (a) El punto óptimo encontrado es  $[1.0, 0.00347, 0.98485, 0.47573]$ . (b) El punto óptimo encontrado es  $[0.80971, 0.0, 0.80569, 0.50326]$ .

Los parámetros seleccionados tras la optimización fueron  $[1.0, 0.00347, 0.98485, 0.47573]$ . La Figura 5.3 muestra la distribución de las métricas SP y AR para las

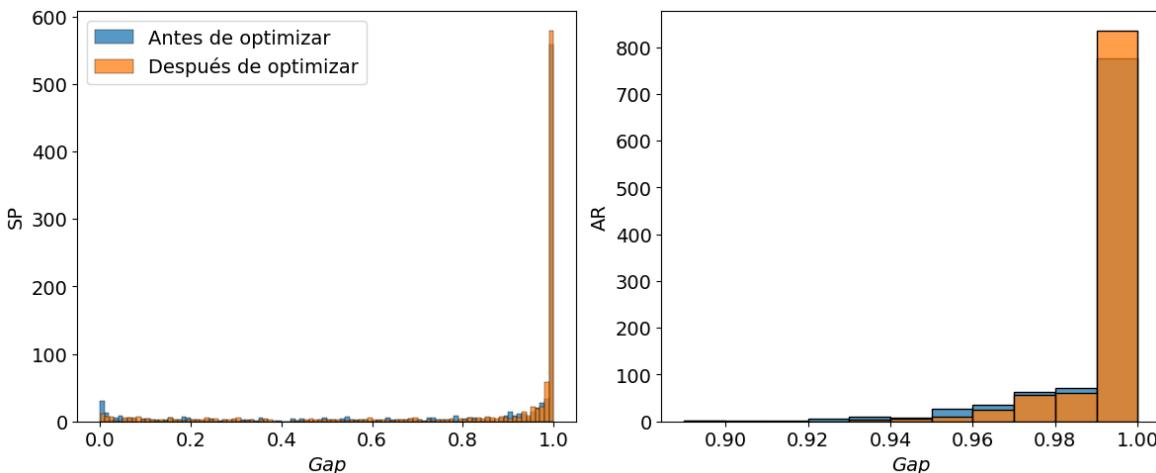
<sup>2</sup><https://docs.scipy.org/doc/scipy/reference/optimize.html>

223 instancias optimizadas. Se aprecia una mejora clara en los resultados: antes de la optimización, ningún grafo alcanzaba un SP superior a 0.95, mientras que tras aplicar COBYLA, la mayoría de las instancias superan ese umbral. En cuanto al AR, también se observa una mejora significativa, con la mayoría de las instancias desplazándose hacia valores superiores a 0,99.



**Figura 5.3:** Comparación del histograma de SP (izquierda) y AR (derecha) tras la optimización COBYLA. Estos resultados se han obtenido al optimizar los 223 grafos que peor AR habían proporcionado previamente.

Motivados por esta mejora, se decidió aplicar estos mismos *drivings* a todo el conjunto de 1000 grafos de siete nodos. Los resultados obtenidos se presentan en la Figura 5.4, donde la mejora sigue presente, aunque es menos pronunciada.

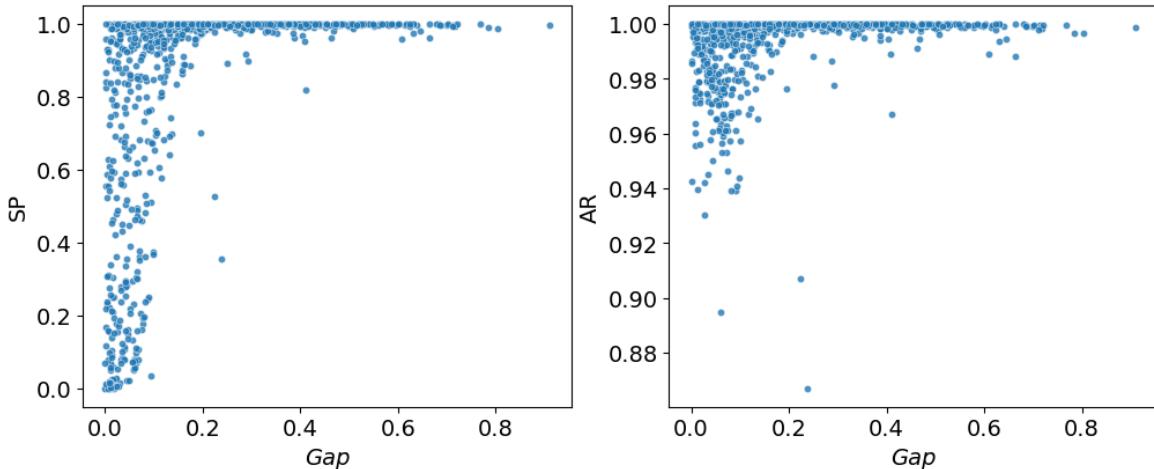


**Figura 5.4:** Comparación del histograma de SP (izquierda) y AR (derecha) tras la optimización COBYLA. Estos resultados se han obtenido al optimizar los 1000 grafos del conjunto de datos.

El AR promedio obtenido fue de 0,99434 y el SP promedio de 0,84557, lo que representa una mejora del 0,21 % en AR y del 3,38 % en SP respecto al caso inicial. Aunque estas mejoras puedan parecer modestas a primera vista, es importante tener

en cuenta que aproximadamente el 80 % de los grafos ya presentaban un AR superior a 0,99 antes de la optimización. Si todos los grafos hubiesen alcanzado un AR de 1, la mejora máxima posible habría sido tan solo del 0,78 %, por lo que el incremento observado es, en realidad, significativo en términos relativos.

Además, en la Figura 5.5 se muestran nuevamente las gráficas de dispersión SP y AR frente al *gap*, lo que permite apreciar visualmente la mejora de forma más clara y cuantitativa. En esta, se observa que los puntos se encuentran más concentrados en zonas específicas y se observan dos regiones claramente diferentes, cuando el *gap* es menor que 0,2 y cuando no lo es.



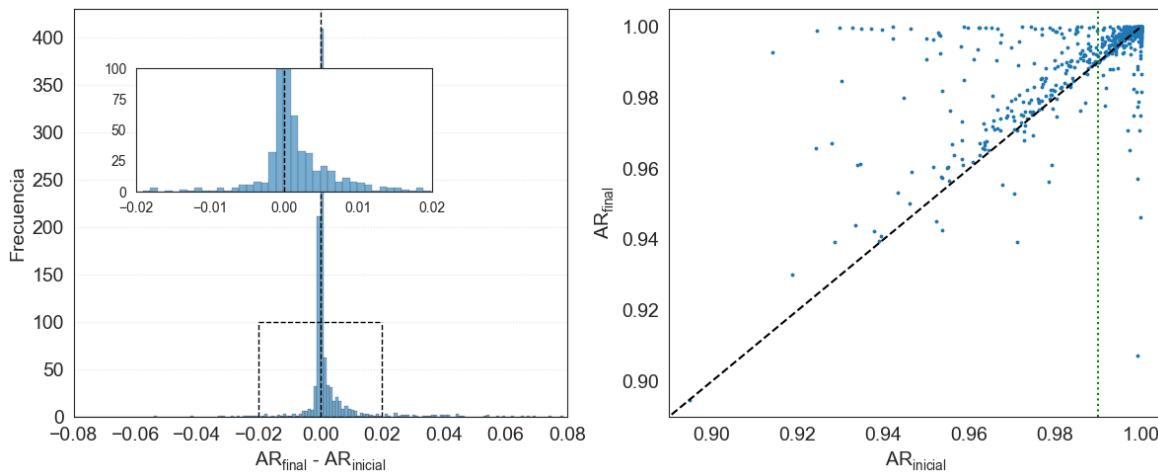
**Figura 5.5:** Dispersión del SP (izquierda) y AR (derecha) al realizar la simulación con los *drivings* obtenidos mediante la optimización por COBYLA.

La Figura 5.6 (izquierda) muestra el histograma de la diferencia de AR antes y después de aplicar la optimización. A pesar de que un número considerable de instancias han empeorado su rendimiento, esto se ve compensado por los casos donde el AR ha aumentado. De esta manera, se observa una mejora neta positiva, como se ha observado en el AR promedio.

En la gráfica de dispersión de la derecha en la Figura 5.6, se aprecia cierta simetría en torno a la línea  $AR_{\text{inicial}} = AR_{\text{final}}$ . Esto refleja que, aunque un número considerable de grafos ha mejorado su rendimiento hasta alcanzar el valor máximo posible, también existen muchos otros que partían con  $AR_{\text{inicial}} = 1$  y que han empeorado tras la optimización. Este efecto explica por qué la mejora global es más visible en la optimización de los 223 grafos que en el conjunto completo.

En resumen, se ha aplicado el algoritmo COBYLA para optimizar los *drivings* en aquellas instancias que inicialmente presentaban un AR inferior a 0,99. A partir de dos inicializaciones distintas se obtuvieron soluciones prácticamente equivalentes, lo cual refuerza la estabilidad de las soluciones encontradas.

Después, se seleccionó la configuración que mayor AR promedio proporcionaba. Este conjunto de parámetros se utilizó posteriormente para simular un conjunto de grafos completo. Como era de esperar, algunos grafos mejoraron su rendimiento pero otros lo empeoraron, lo cual pone de manifiesto el carácter individual de cada instancia. Por ello, parece más adecuado seleccionar de manera aleatoria las semillas que formarán parte de la optimización en lugar de centrarse en los peores casos.



**Figura 5.6:** (Izquierda) Histograma de la diferencia de AR antes y después de la optimización. Mediante una subfigura, se ilustra la región definida por las líneas intermitentes aumentada. (Derecha) Dispersión del AR por grafo antes ( $AR_{\text{inicial}}$ ) y después ( $AR_{\text{final}}$ ) de optimizar. La línea intermitente representa los casos donde el AR no varía. La línea punteada vertical separa los grafos utilizados en la optimización (izquierda) del resto (derecha).

### Optimización Bayesiana

La optimización Bayesiana se ha llevado a cabo mediante la librería `bayes_opt`<sup>3</sup>. Como se ha mencionado previamente, la selección de las instancias a optimizar se realizó de forma aleatoria, seleccionando un total de 250 grafos. Tanto los parámetros optimizados como los rangos de valores definidos fueron los mismos que en el caso del algoritmo COBYLA.

Una de las principales ventajas del optimizador Bayesiano es que aprovecha la información obtenida en iteraciones previas para guiar la búsqueda de soluciones óptimas, lo que permite realizar una exploración más eficiente. Esto no solo permite almacenar y retomar la optimización en un momento posterior, sino que también posibilita evaluar puntos específicos definidos por el usuario.

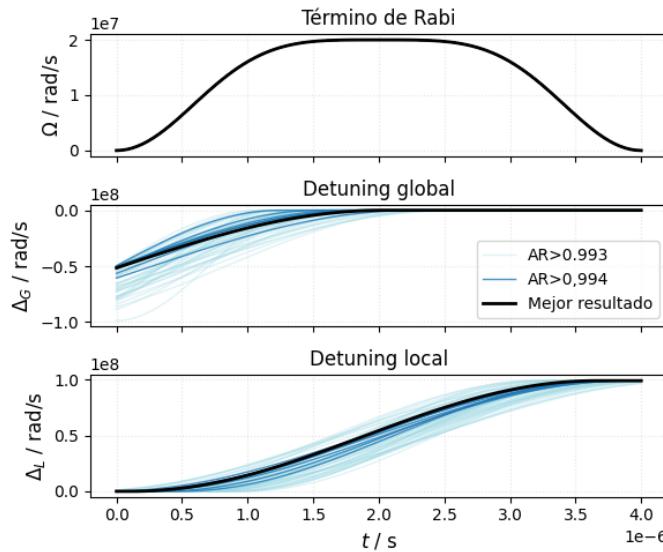
En este trabajo se ha empleado el modelo de adquisición *Upper Confidence Bound* (UCB) debido a su habilidad para controlar explícitamente el balance entre exploración (probar regiones nuevas del espacio de parámetros) y explotación (profundizar en regiones ya conocidas que han ofrecido buenos resultados). Al poder ajustar este balance mediante un parámetro explícito, UCB permite mantener una exploración activa durante más iteraciones, reduciendo así la probabilidad de converger prematuramente a soluciones subóptimas. Esta característica lo hace especialmente adecuado para problemas complejos como el presente, donde la superficie objetivo puede contener múltiples máximos locales y se desconoce la posición exacta del óptimo global.

Inicialmente, la optimización se ejecutó en un entorno local. Sin embargo, tras obtener acceso al servidor LluisVives de la universidad, se adaptó el código a un archivo ejecutable en formato .py con el fin de lanzarlo en el servidor. Este cambio permitió

<sup>3</sup><https://bayesian-optimization.github.io/BayesianOptimization/2.0.3/>

además parallelizar el proceso, distribuyendo la evaluación de los 250 grafos de cada iteración entre 32 CPUs, lo que redujo significativamente el tiempo de ejecución.

A pesar de que en 200 iteraciones aproximadamente se consiguió mejorar los resultados obtenidos hasta este momento, se llegaron a hacer 856 evaluaciones. Dentro de estas, 60 configuraciones de los parámetros obtuvieron un AR > 0,993 y ocho superaron el 0,994. En la Figura 5.7 se muestran los parámetros que mejores resultados proporcionaron.



**Figura 5.7:** Mejores configuraciones de *drivings* obtenidos mediante la optimización Bayesiana. En azul claro se muestran los que obtuvieron un AR > 0,993, en azul oscuro AR > 0,994 y en negro la mejor configuración.

Las líneas en azul claro muestran las soluciones correspondientes a las mejores 60 configuraciones, en azul oscuro a las mejores ocho y en negro la mejor configuración obtenida. Se observa que todas las curvas presentan formas similares, lo cual indica una buena robustez en los resultados de la optimización, ya que pequeñas variaciones en los parámetros no producen grandes diferencias en los perfiles óptimos. Asimismo, la mejor configuración obtenida es [0.994101, 0.013208, 0.887301, 0.472217], que proporciona un AR promedio de 0,99424 y es una solución prácticamente equivalente a la obtenida mediante la optimización por COBYLA.

En resumen, del conjunto de grafos se han seleccionado 250 de manera aleatoria y se ha realizado una optimización Bayesiana sobre ellos. Con 856 evaluaciones y un tiempo de ejecución de 3 horas y 30 minutos en el servidor, fue suficiente para alcanzar un valor de AR promedio de 0,99424. Asimismo, se ha observado que el conjunto de *drivings* que mejores resultados han dado se encuentran en una misma región, lo que proporciona los primeros indicios de que el AR promedio depende de los parámetros de *drivings* de una manera suave.

### Grid search

Un *grid search* de los parámetros no solo es un estudio exhaustivo que permite la validación de los resultados obtenidos, sino que nos permite estudiar la manera en la

que varía la función de coste en función de los parámetros escogidos y ver si lo hace de una manera suave o presenta discontinuidades, lo cual aporta una visión más profunda sobre la estructura del problema.

Para diseñar un *grid search* que se adecue tanto a nuestras necesidades como a la capacidad del servidor, se realizó una estimación del número de puntos posibles a evaluar y el coste computacional de este. Como el número de combinaciones posibles crece exponencialmente con el número de parámetros, se decidió restringir el estudio a los mismos cuatro parámetros utilizados en las optimizaciones anteriores. Además, a partir de los tiempos medidos durante la optimización Bayesiana, se pudo estimar la duración de cada evaluación.

El tiempo total de ejecución se estimó mediante la siguiente ecuación:

$$T_T = \frac{N^4 \cdot T_{\text{eval}}}{\text{CPUs por job} \cdot \text{Número de jobs}}, \quad (5.1)$$

donde

- $N^4$  = número de evaluaciones total ( $N$  es el número de puntos que evaluamos en cada uno de los 4 parámetros).
- $T_{\text{eval}}$  = el tiempo que toma una evaluación en 1 CPU.
- $T_T$  = Tiempo de ejecución total.
- CPUs por *job* = Número de CPUs asignadas a cada *job*.
- Número de *jobs* = Número de tareas que se ejecutan simultáneamente.

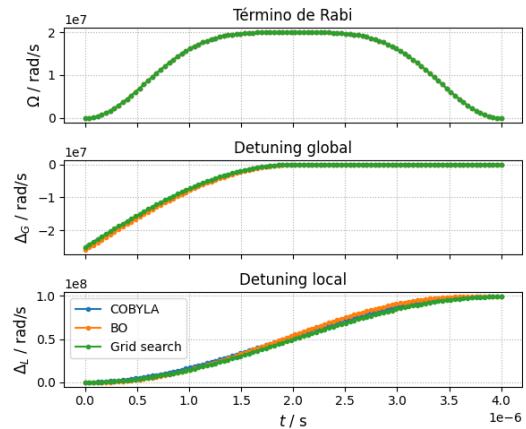
Según las medidas obtenidas en la optimización Bayesiana, el tiempo medio de evaluación en una CPU fue de aproximadamente 576 segundos (0,16 horas). Despejando  $N$  en la Ecuación (5.1) y considerando diferentes configuraciones posibles, se optó finalmente por usar 32 CPUs por tarea, lanzar 3 tareas en paralelo y evaluar 15 valores por cada parámetro, resultando en un total de 50.625 evaluaciones.

Los resultados del *grid search* se almacenaron en un archivo CSV que incluye todas las combinaciones de parámetros evaluadas junto con su correspondiente AR promedio. A pesar de que nos vayamos a centrar en el resultado óptimo del *grid search*, más adelante se realizará un estudio más profundo. El AR promedio máximo al que se llegó fue 0.9942 y se logró mediante la configuración de parámetros [1, 0, 1, 0.5]. Como se observa en la Tabla 5.2, los resultados de los tres métodos coinciden y proporcionan valores de AR promedio similares.

Por tanto, podemos concluir que para la parametrización y rango de valores escogidos los resultados proporcionados por el algoritmo de optimización COBYLA son los mejores encontrados. Sin embargo, quedan algunas cuestiones abiertas como los resultados que proporcionan valores fuera de estos rangos y el efecto de variar los parámetros fijos que se mencionan en la Tabla 5.1. Este último aspecto se abordará en la siguiente subsección.

**Tabla 5.2:** Valores óptimos de los parámetros de los *drivings* obtenidos mediante las optimizaciones y *grid search*, junto con el valor de AR promedio obtenido.

	Parámetros	AR
COBYLA	[1.0, 0.0035, 0.9849, 0.4757]	0.9943
BO	[0.9941, 0.0132, 0.8873, 0.4722]	0.9942
<i>Grid search</i>	[1.0, 0.0, 1.0, 0.5]	0.9942



**Figura 5.8:** Representación de los *drivings* que se muestran en la Tabla 5.2.

### 5.2.3. Mejora de los parámetros fijos

Una vez conseguido el perfil de los *drivings* que dieran los mejores resultados, ya que se disponía de la capacidad computacional, se decidió realizar otro *grid search* de algunos de los parámetros que se había dejado fijos. De esta manera, se trató de encontrar los valores de  $\Omega_{\max}$ ,  $\Delta_{\text{global},\max}$  y  $\Delta_{\text{local},\max}$  que mejores resultados dieran. Se optó por estudiar estas tres variables porque se intuía que podrían influir en los resultados considerablemente. Además, esto nos permitía tener un argumento más sólido para su elección.

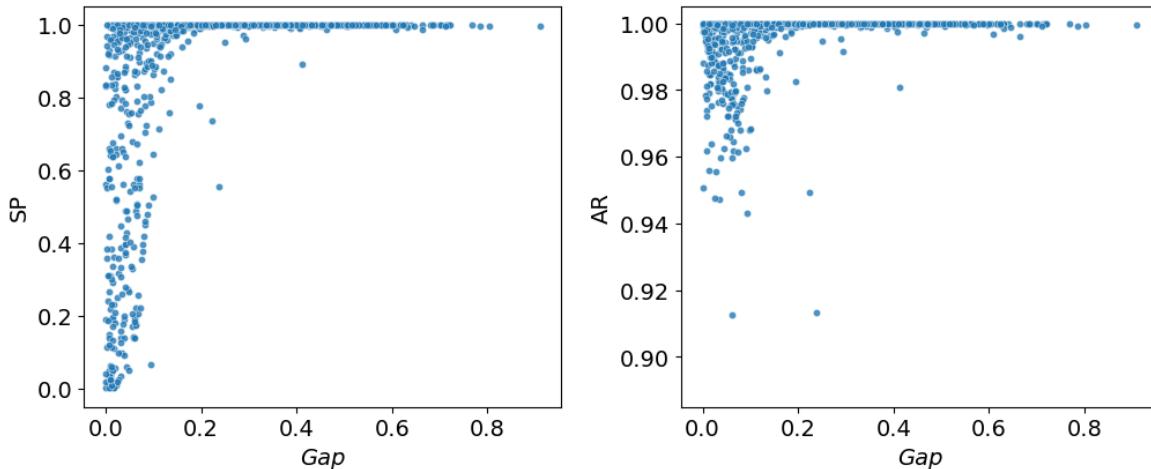
El *grid search* se hizo fijando los parámetros que habíamos optimizado hasta ahora en  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}] = [1.0, 0.0035, 0.9849, 0.4757]$ . El rango sobre el que se estudió el efecto de estos tres parámetros en la función de coste fue  $[0, 1]$ , ya que se normalizaron los resultados con sus valores máximos respectivos. Asimismo, debido a que el número de parámetros es menor, pudimos evaluar 30 puntos por parámetro.

De esta manera, la configuración de valores máximos que mayor rendimiento dieron fue:  $\Omega_{\max} = 1.0$ ,  $\Delta_{\text{global},\max} = 0.413793$  y  $\Delta_{\text{local},\max} = 1.0$ . Así pues, se volvió a realizar una simulación del conjunto de grafos completo con los parámetros óptimos encontrados hasta este momento (Tabla 5.3).

**Tabla 5.3:** Valores de los parámetros que caracterizan los *drivings* óptimos.

Parámetro	$t_{\max}$	$\Omega_{\max}$	$\Delta_{\text{global},\max}$	$\Delta_{\text{local},\max}$	$a_{\text{global}}$	$b_{\text{global}}$	$a_{\text{local}}$	$b_{\text{local}}$
Valor	$4 \mu\text{s}$	$25 \cdot 10^6 \text{ Hz}$	$51,72 \frac{\text{rad}}{\mu\text{s}}$	$125 \frac{\text{rad}}{\mu\text{s}}$	1	0,0035	0,9849	0,475

En la Figura 5.9 se muestran las gráficas de dispersión de SP y AR frente al *gap*. Con esta configuración, se obtuvo un AR promedio de 0,9964, el mejor hasta el momento.



**Figura 5.9:** Dispersion del SP (izquierda) y AR (derecha) al realizar la simulación con los *drivings* definidos por los parámetros que se muestran en la Tabla 5.3.

#### 5.2.4. Análisis de resultados

Como se observa en la Figura 5.9, obtener mejores valores de AR promedio no solo implica conseguir mejores resultados, sino que nos permite apreciar con mayor claridad el efecto que tiene el *gap* en el resultado. Vemos que para valores de *gap* < 0,2 los resultados empeoran tanto para el SP como para el AR. Como se ha comentado, el Teorema de Adiabaticidad explica que si la diferencia de energía entre el estado fundamental y resto es muy pequeña, podría haber transiciones a estados excitados, lo que implica que el sistema no acabe en el estado óptimo, resultando en la bajada de rendimiento del SP y AR.

En la gráfica de la izquierda de la Figura 5.9 vemos que la bajada del rendimiento del SP es gradual, de manera que para valores más bajos de *gap*, mayor variabilidad hay en el resultado. Sin embargo, en la figura de AR frente al *gap* esto no es así. Se observa que la variabilidad no aumenta tanto y además aparece una figura en forma de triángulo invertido cuando el *gap* se hace menor que 0,2 con el pico en  $gap \approx 0,1$ . La razón por la que el AR no disminuye tanto se podría explicar mediante el siguiente argumento.

Consideremos un grafo con los dos estados de menor energía muy alejados del resto del espectro. La diferencia entre el valor de coste de estos sería el *gap*. Cuando se va haciendo más pequeño, la probabilidad de que el sistema salte al estado excitado se hace más grande, por lo que a partir de cierto punto, el AR empieza a disminuir. Sin embargo, al hacer el *gap* muy pequeño, aunque la probabilidad de transición sea mucho más grande, ambos estados son casi igual de óptimos, por lo que el AR aumenta mientras que el SP puede disminuir. Este argumento se puede complicar cuando tenemos en cuenta el resto de los estados.

Se ha observado que este efecto de la disminución del rendimiento con el *gap* aparece para cualquier configuración de *drivings* con los que se haya simulado el conjunto de grafos, lo que nos hace preguntarnos: ¿la bajada de rendimiento para *gaps* pequeños está relacionada con los *drivings* escogidos, o es inherente a los grafos? ¿Es

posible encontrar unos *drivings* que sobrepasen este problema? Esto se estudiará en mayor profundidad en la Sección 5.4.

En resumen, se han probado tres maneras para realizar una búsqueda de los parámetros que maximizan el rendimiento al promediar el resultado de 1000 grafos, como se muestran en la Tabla 5.2. El mejor rendimiento lo ha dado el algoritmo de COBYLA. Sin embargo, este es un optimizador local, por lo que corre el peligro de quedarse estancado en un mínimo local. Por otro lado, el algoritmo de BO permite sobrepasar este problema al realizar una búsqueda global inteligente. Un posible flujo de trabajo en futuras optimizaciones sería comenzar por una optimización Bayesiana para encontrar el mínimo global, y después realizar una optimización por COBYLA para hacer *fine tuning* de los resultados.

En cuanto a los resultados del *grid search* de los parámetros  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}]$ , el hecho de que tres de ellos se encuentren en un extremo del rango de valores estudiados, nos hace pensar que sería interesante extender el estudio a valores fuera de estos, ya que podría encontrarse una configuración aún mejor. Sin embargo, incluso ampliando ese rango, seguiría siendo imposible concluir que se ha encontrado el conjunto óptimo, ya que otros parámetros permanecen fijos. Es por eso que, en este trabajo, nos hemos centrado en buscar la existencia de unos *drivings* que fueran aplicables a todos los grafos y que proporcionasen resultados razonablemente buenos. Con una mayor capacidad computacional y tiempo, sería posible encontrar el conjunto de parámetros completos que maximicen el AR promedio.

### 5.3. Generalización de los *drivings*

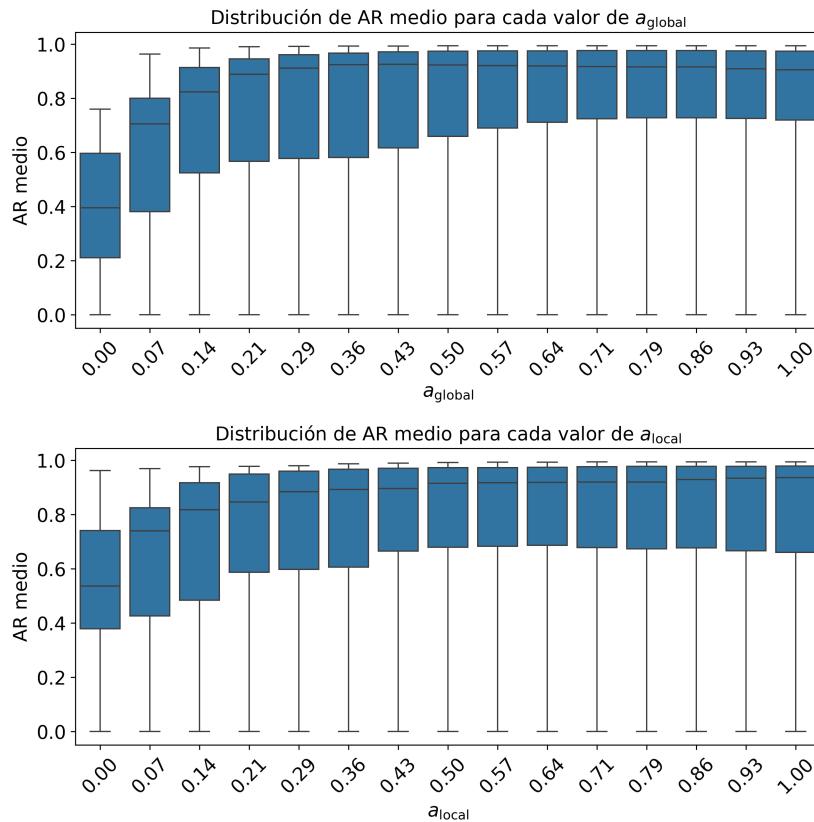
En esta sección se van a analizar los resultados del *grid search* con el objetivo de estudiar cómo varía la función de coste en función de los *drivings*. Ello, nos permitirá tener un conocimiento más profundo del problema y poder hacer una mejor selección de *drivings* en un futuro.

Dado que la función objetivo depende de cuatro variables, su representación completa en un único gráfico no es viable. Por tanto, se optó por analizar la contribución marginal de cada parámetro mediante gráficos tipo *boxplot*. En todos los *boxplot* que se muestren en este trabajo se han alargado los bigotes, ya que puntos alejados de la distribución no son indicativos de ningún *outlier* o anomalía.

Para cada uno de los cuatro parámetros, se ha fijado su valor y se han representando, en forma de *boxplot*, las distribuciones del coste obtenido en todas las combinaciones posibles de los otros tres parámetros. De este modo, se visualiza cómo varía el AR promedio de las soluciones en función de cada valor del parámetro considerado, teniendo en cuenta el comportamiento estadístico del resto de combinaciones.

Este enfoque permite detectar valores de los parámetros que sistemáticamente conducen a mejores (o peores) resultados, proporcionando así información útil sobre su relevancia y posible interacción con los demás.

En la Figura 5.10 se observa que prácticamente para cualquier valor fijo de  $a_{\text{global}}$  y  $a_{\text{local}}$  hay una configuración con la que se puede obtener cualquier valor de AR promedio cercano a 0 o 1. No obstante, para ambos parámetros resulta más favorable

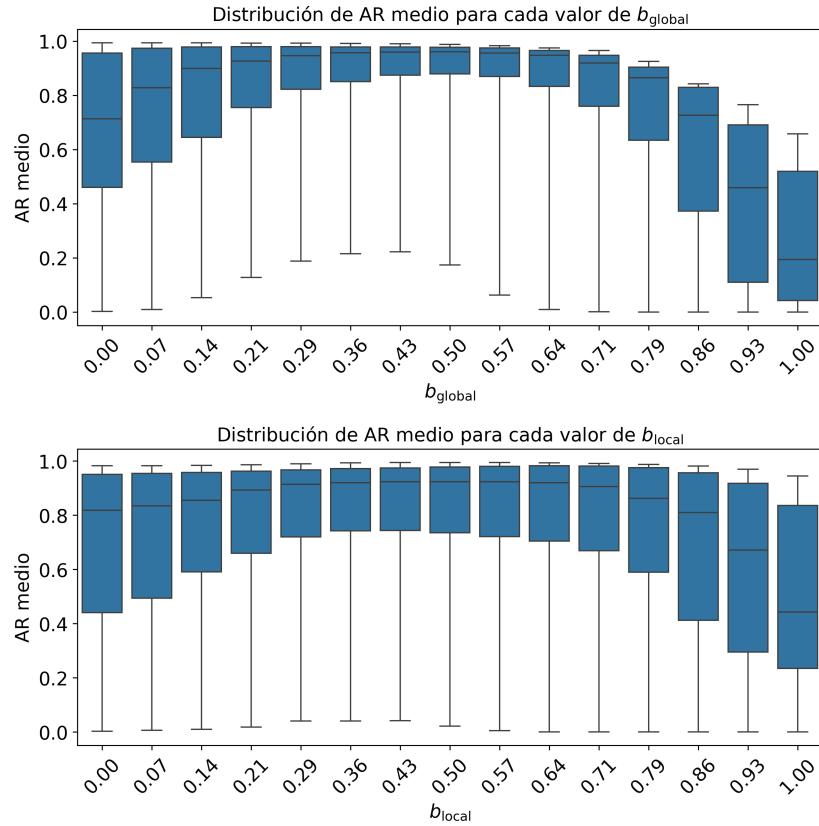


**Figura 5.10:** Boxplot de AR promedio al marginalizar por diferentes valores de  $a_{\text{global}}$  (superior) y  $a_{\text{local}}$  (inferior).

escoger valores cercanos a 1, ya que tanto el rango intercuartílico como la mediana se encuentran en valores más altos. Estos resultados sobre los parámetros que definen la anchura temporal en la que se producen la transición, indican que es preferible tiempos de transición más grandes. Esto coincide con la intuición física, ya que transiciones más suaves permiten que el sistema permanezca en su estado fundamental.

Por otro lado, en la Figura 5.11 se aprecia que las mejores distribuciones de AR se obtienen cuando  $b_{\text{global}}$  y  $b_{\text{local}}$  se sitúan alrededor de 0.5. De nuevo, aunque no existe un valor único que garantice un buen resultado, se observa que estas regiones tienden a concentrar más configuraciones de alto rendimiento. Esto sugiere que situar el centro de la transición en la mitad del tiempo total de evolución puede ser una estrategia favorable para guiar el sistema hacia el estado fundamental de forma eficiente.

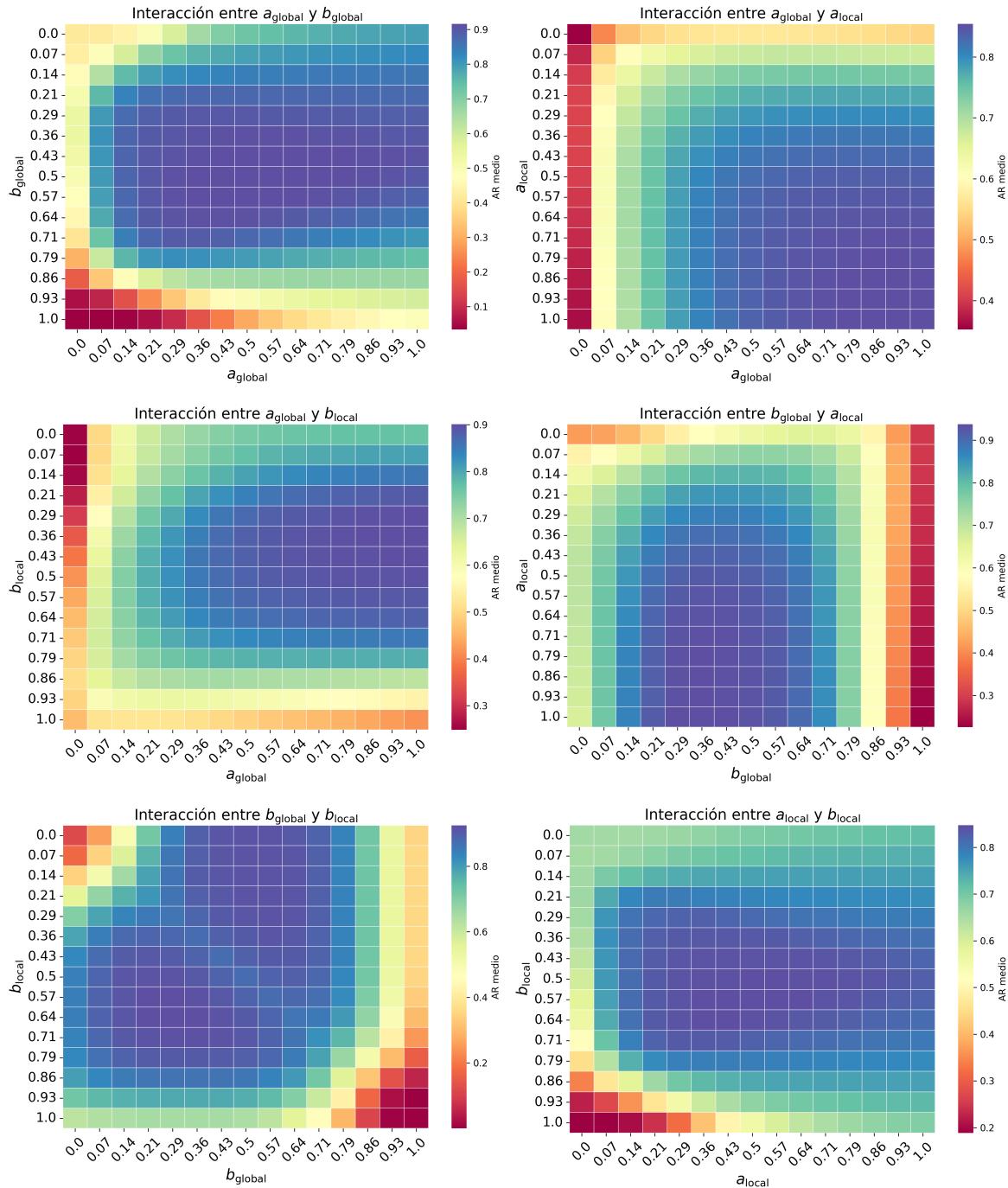
No obstante, como se ha comentado, estas figuras no permite identificar configuraciones singulares que produzcan buenos resultados. La Figura 5.12 muestra el AR promedio al marginalizar por dos variables. Como observamos al representar  $b_{\text{global}}$  y  $b_{\text{local}}$ , parece contraproducente establecer ambos valores en 0,5. En cambio, parece más favorable que uno suceda antes que el otro. Si razonamos con el conocimiento que tenemos del problema, es mejor que primero el  $\Delta_{\text{global}}$  transicione y se vaya apagando y que después  $\Delta_{\text{local}}$  vaya aumentando el valor. De esta manera, el Hamiltoniano inicial trata a todos los átomos por igual y hacia el final va adquiriendo su carácter individual al tener en cuenta el peso de cada uno. Este razonamiento coincide con todos los resultados obtenidos de optimizaciones, donde primero  $\Delta_{\text{global}}$  va a apagándose y después  $\Delta_{\text{local}}$  va haciéndose más intenso.



**Figura 5.11:** Boxplot de AR promedio al marginalizar por diferentes valores de  $b_{\text{global}}$  (superior) y  $b_{\text{local}}$  (inferior).

Por otro lado, la Figura 5.12 muestra que las configuraciones de menor rendimiento se producen únicamente en los valores extremos. El hecho de que los tonos rojo y naranjas se encuentren tan asilados indica que al alejarnos de estos valores, la función de coste varía rápidamente a valores más altos, donde esta empieza a tomar una forma más suave. Cabe recordar que, cada celda que se muestra es el promedio de todas las configuraciones posibles de dos parámetros, por lo que esta ilustración únicamente nos da una idea de la forma de la función de coste.

Además, parece que los mejores resultados se encuentran en los extremos del rango estudiado, señalando que puede haber mejores configuraciones fuera de este. No obstante, el hecho de que parte de la transición de  $\Delta_{\text{global}}(t)$  suceda fuera del rango temporal, puede ser debido a la necesidad de un valor más bajo del mismo, como se ha encontrado en la Sección 5.2.3.

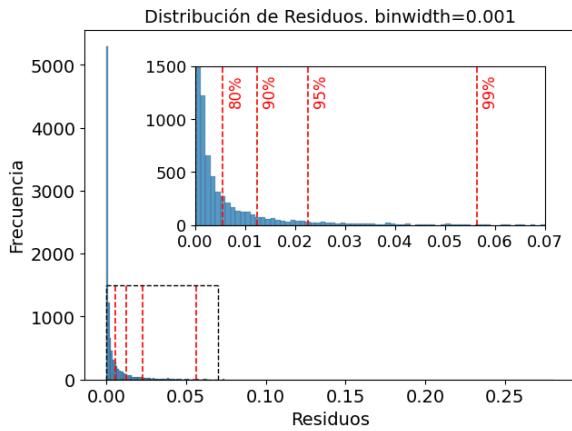


**Figura 5.12:** AR promedio al marginalizar por dos de las cuatro variables con las que se parametrizan los *drivings*.

### Modelización de la función de coste

Al ver los indicios de que el AR promedio variaba de una manera suave y que no muestra discontinuidades con los parámetros que variamos, se decidió hacer una regresión con el objetivo de modelizar la función de coste. Para ello, se separó el conjunto de configuraciones en grupos de entrenamiento (80 %) y test (20 %) y se entrenó un modelo de regresión Random Forest con 180 estimadores.

Mediante el conjunto de test, probamos el rendimiento del modelo, obteniendo un  $R^2 = 0,998$  y  $MAE = 0,0048$  en el conjunto de test. Teniendo en cuenta que la variable de predicción varía entre 0,90 y 1,00, vemos que el modelo es capaz de entender correctamente la manera en la que la función de coste varía con los cuatro parámetros.



**Figura 5.13:** Histograma de los residuos del modelo Random Forest en el conjunto de test. Las líneas intermitentes verticales indican los percentiles 80, 90, 95 y 99.

La Figura 5.13 muestra el histograma de los residuos junto con los percentiles 80, 90, 95 y 99. Como se observa, casi el 90 % de las instancias de test tienen un error menor que el 1 %.

Estos resultados indican que la función objetivo (AR promedio) varía de forma suave y continua con respecto a las curvas de *drivings* seleccionadas, lo cual hasta ahora se había asumido implícitamente. Esta observación sugiere además que podría ser posible predecir la calidad de una configuración concreta sin necesidad de realizar simulaciones completas, abriendo la puerta a enfoques basados en aprendizaje supervisado que aceleren significativamente el proceso de optimización.

En resumen, se ha analizado la dependencia del AR promedio respecto a los parámetros de los *drivings*:  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}]$ . El análisis realizado, marginalizando sobre uno o dos parámetros y considerando el efecto promedio de los demás, permite extraer conclusiones generales sobre tendencias, aunque no permite conclusiones exactas sobre todas las interacciones posibles. De este estudio se concluye que es preferible escoger valores cercanos a 1 para los parámetros que controlan la anchura de las transiciones temporales, es decir,  $a_{\text{global}}$  y  $a_{\text{local}}$ .

En cuanto a los parámetros que definen el instante central de la transición ( $b_{\text{global}}$  y  $b_{\text{local}}$ ), los resultados indican que valores alrededor de 0.5 tienden a ser favorables, pero la interacción entre ambos parámetros también resulta relevante. En particular, la Figura 5.12 muestra que asignar valores idénticos a ambos parámetros no es la mejor estrategia. De hecho, la mejor combinación obtenida ha sido  $[1, 0, 1, 0, 5]$ , reforzando la idea de que una transición del *detuning* global debe ocurrir antes que la del *detuning* local ( $b_{\text{global}} < b_{\text{local}}$ ), lo que coincide con la intuición física acerca de una evolución más efectiva del sistema.

Finalmente, estos hallazgos, junto con el modelo predictivo obtenido mediante

Random Forest, sugieren claramente que la función objetivo es suave y no presenta discontinuidades abruptas, lo que refuerza la viabilidad de un futuro estudio basado en técnicas de aprendizaje supervisado para predecir y optimizar los *drivings* sin necesidad de recurrir a simulaciones intensivas.

## 5.4. Optimización por grafo

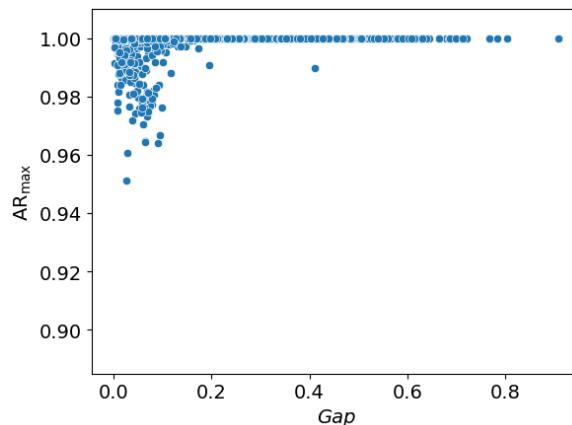
Como se ha mencionado en la Sección 5.2.2, independientemente de los *drivings* utilizados, el rendimiento tiende a empeorar significativamente cuando el *gap* entre el estado fundamental y el subóptimo es inferior a aproximadamente 0,2. Este hecho nos motivó a investigar si dicha limitación está relacionada con los protocolos de *drivings* escogidos o con propiedades intrínsecas de los grafos.

Para analizar la necesidad de *drivings* únicos para cada grafo, se decidió realizar una optimización individualizada. De esta manera, se buscaron los parámetros óptimos de los *drivings* que maximizaran el AR para cada grafo. Si todos los grafos alcanzasen un AR cercano a 1 con parámetros específicos, se confirmaría que es imprescindible considerar la individualidad de cada grafo al optimizar.

Por otro lado, si tras esta optimización específica por grafo aún persistiera la disminución del rendimiento observada, las causas podrían atribuirse a dos factores:

1. La parametrización elegida podría no ser suficientemente flexible para capturar soluciones óptimas en todos los grafos.
2. Existen propiedades intrínsecas en ciertos grafos que dificultan fundamentalmente su resolución.

La optimización por grafo se llevó a cabo mediante optimización Bayesiana, estableciendo un máximo de 400 iteraciones. La función de adquisición utilizada y el hiperparámetro que controla el balance entre exploración y explotación fueron los mismos definidos anteriormente.



**Figura 5.14:** Dispersión del AR máximo alcanzado al optimizar los *drivings* para cada grafo en función del *gap*.

La Figura 5.14 muestra una gráfica de dispersión del AR máximo alcanzado al

optimizar individualmente cada grafo frente al valor del *gap*. Se observa claramente que la disminución del rendimiento para  $gap < 0,2$  persiste incluso tras la optimización específica por grafo. Para valores de *gap* mayores que 0,2, solamente una instancia no alcanza un AR superior a 0,999.

Este análisis no solo establece un límite superior del AR alcanzable para cada grafo bajo la parametrización actual, sino que además proporciona una referencia ideal hacia la cual cualquier otro conjunto de *drivings* debería aspirar. Esta referencia puede emplearse para definir métricas adicionales que permitan evaluar la eficiencia relativa de diferentes estrategias de optimización. Por ejemplo, si consideramos este, como el escenario ideal, los *drivings* óptimos encontrados muestran un error promedio del 1,7 %.

La Figura 5.14 también refuerza la idea de que la caída en rendimiento está relacionada directamente con la magnitud del *gap* entre el estado fundamental y el siguiente estado subóptimo. Este *gap* depende exclusivamente de la función de coste, que, al tratarse de un problema combinatorio, presenta un número discreto y finito de soluciones. El valor de este *gap* viene determinado por la topología específica del grafo y los pesos asignados a sus nodos. Esto implica que ciertos grafos, debido a su estructura inherente, presentan una dificultad fundamental mayor para ser resueltos de manera efectiva.

Explorar otras parametrizaciones o protocolos alternativos de *drivings* queda fuera del alcance de este trabajo, pues las opciones posibles son infinitas y no existe garantía de mejora. Como se mencionó anteriormente, la existencia de un *gap* energético suficientemente grande es imprescindible para aplicar el Teorema de la Adiabaticidad. Cuanto menor es este *gap*, más suaves y lentas deben ser las transiciones en el sistema para evitar excitaciones no deseadas, lo cual puede verse restringido por limitaciones prácticas del *hardware*.

Adicionalmente, en este estudio solo se ha analizado el *gap* asociado a la función de coste al final de la evolución y no el correspondiente al Hamiltoniano durante la evolución misma. Esto no representa un problema conceptual, dado que el Hamiltoniano final emula directamente la función de coste del MWIS, lo que implica que sus espectros de estados son equivalentes al final de la evolución. No obstante, para que el Teorema de la Adiabaticidad se aplique rigurosamente, se requiere que este *gap* se mantenga grande durante toda la evolución del sistema.

En etapas previas del proyecto se intentó llevar a cabo un análisis exhaustivo para medir la adiabaticidad de la evolución respecto a los *drivings* utilizados. Sin embargo, debido a la complejidad generada por estados altamente superpuestos durante la evolución, este análisis resultó inviable y tuvo que descartarse.

Por tanto, el enfoque final del proyecto se centró en analizar la estructura de los grafos y determinar qué propiedades estructurales contribuyen a que algunos grafos sean intrínsecamente más difíciles de resolver. En la literatura especializada se han propuesto diversos parámetros para medir dicha dificultad en problemas relacionados con el MIS [28].

Uno de estos parámetros, conocido como *Hardness Parameter* (HP), se define como el ratio entre la degeneración del estado fundamental y el primer estado subóptimo. La intuición detrás de esta definición es que, si el estado óptimo está muy degenerado, existe una gran cantidad de soluciones igualmente válidas, facilitando la resolución

del problema. Sin embargo, esta definición no se aplica directamente al problema de MWIS, ya que al incorporar pesos específicos se rompe esta degeneración. Por esta razón, y siguiendo esta línea conceptual, se propone en este trabajo una nueva definición de parámetro que capture de manera más precisa la densidad de estados cercanos al estado fundamental, permitiendo evaluar de forma efectiva la dificultad intrínseca asociada a cada grafo.

## 5.5. Estudio de características de grafo que dificultan su resolución

Esta sección tiene como objetivo identificar y analizar las características de los grafos que están relacionadas con la disminución del rendimiento de ciertos grafos para el problema del MWIS. El análisis se centra en dos tipos de variables:

1. **Propiedades derivadas del espectro de la función de coste:** como el *gap* de coste entre el estado fundamental y los primeros estados excitados, el coste del estado fundamental y un nuevo parámetro denominado *Hardness Parameter* (HP), entre otros.
2. **Propiedades intrínsecas del grafo:** propiedades relacionadas con la topología de los grafos tales como la densidad de aristas, el diámetro o el grado medio entre otros.

Es importante destacar que el cálculo de las propiedades espectrales requiere la evaluación de todos los posibles estados del sistema, lo cual representa una limitación computacional significativa para el estudio de grafos de gran tamaño.

A continuación, se desarrolla y entrena una serie de modelos de clasificación binaria. El propósito de estos modelos es predecir si, para una instancia de grafo dada y utilizando los protocolos *drivings* óptimos previamente determinados en la Sección 5.2.3, el AR alcanzado será inferior a un umbral de alta fidelidad (específicamente, 0,999).

Finalmente, aunque visualmente los resultados presentados en la Figura 5.14 podrían sugerir que el rendimiento general es satisfactorio y los errores residuales son asumibles, un análisis más profundo de las instancias difíciles es crucial para mitigar nuevas posibles fuentes de error y la comparación entre diferentes ordenadores cuánticos y otros algoritmos clásicos.

### 5.5.1. Definición del *Hardness Parameter* (HP) para el MWIS

Se recomienda que el lector ejecute el Dashboard que se encuentra en la carpeta Dashboard del repositorio GitHub. Aquí, se muestra la dispersión de AR frente al *gap* de manera interactiva. El lector puede seleccionar una instancia deseada y observar tanto el grafo como el espectro de su función de coste. Además, puede filtrar por diferentes valores de HP. A pesar de que se mostrarán las figuras más relevantes para la comprensión del estudio, el *dashboard* puede ser de ayuda para ilustrar y comprender diferentes casuísticas.

La hipótesis utilizada para buscar una nueva definición del HP ha sido que cuanto más aislado se encuentre el estado fundamental (energéticamente), más difícil será que acabe en un estado diferente al óptimo. Por un lado, si el *gap* es muy grande, el estado fundamental se encuentra aislado, ya que el *gap* se calcula con el segundo estado más óptimo. Por otro lado, cuando el *gap* es muy pequeño, no se puede asegurar nada, ya que existen una variedad de escenarios. Puede ser que justo estos dos estados se encuentren aislados del resto del espectro (Figura 5.15 (a)), o que haya una multitud de estados en las cercanías (Figura 5.15 (b)).

Para cuantificar esta “densidad de estados cercanos” en torno al estado fundamental ( $C_0$ ), se propone el HP calculado como la suma de los valores de una función Gaussiana evaluada en el coste ( $C_i$ ) de cada uno de los  $N$  posibles estados (soluciones) del sistema:

$$\text{HP} = \sum_{i=0}^N G(C_i; b), \quad (5.2)$$

donde  $G(x; b)$  es la función Gaussiana centrada en la energía del estado fundamental  $C_0$ :

$$G(x; b) = \exp\left(-\frac{(x - C_0)^2}{2\sigma^2}\right) \quad (5.3)$$

y

$$\sigma = \frac{b}{2\sqrt{2\ln(2)}}, \quad (5.4)$$

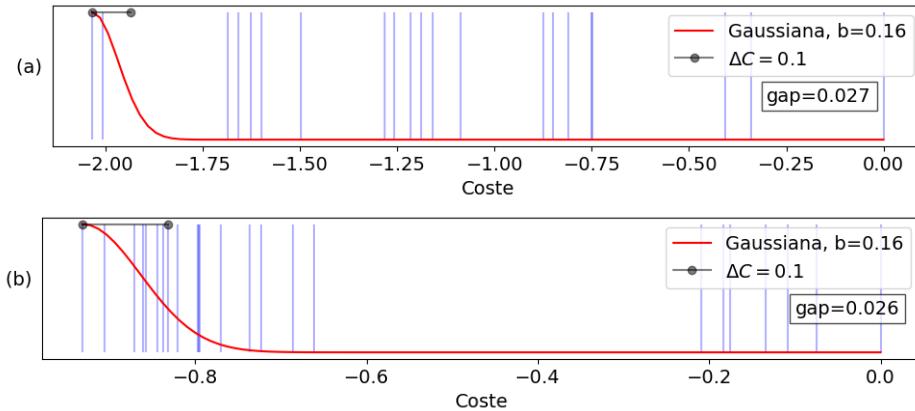
donde  $b$  es la anchura a media altura (FWHM) de la Gaussiana. Esta, actúa como un hiperparámetro que define la “vecindad” energética alrededor de  $C_0$  que se considera relevante. Un valor mayor de  $b$  incluye estados energéticamente más alejados en el cálculo del HP.

La Figura 5.15 ilustra esta idea mostrando el espectro de coste de dos grafos con *gaps* muy similares. Las líneas verticales azules representan el coste de los posibles estados (conjuntos independientes). La curva roja es la función Gaussiana ( $G(x; b)$ ) centrada en  $C_0$  con  $b = 0,16$  (valor elegido para ilustración; la optimización de  $b$  se discute más adelante). La línea horizontal gris marca una diferencia de energía de 0,1 respecto a  $C_0$  y sirve como referencia.

Como se observa en la Figura 5.15, aunque ambos grafos tienen *gaps* similares, la densidad de estados cercanos al fundamental difiere notablemente. El HP captura esta diferencia: el grafo (a), hipotéticamente más fácil, tiene un  $\text{HP}_a = 1,94$ , mientras que el grafo (b), hipotéticamente más difícil, tiene un  $\text{HP}_b = 6,54$ . Un valor de HP más alto indicaría una mayor concentración de estados cerca del óptimo, lo que podría facilitar la transición a estados subóptimos.

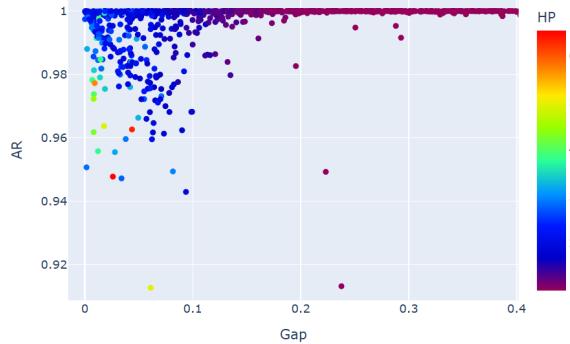
Para encontrar el valor del hiperparámetro óptimo se ha definido un rango de posibles valores y se ha escogido el valor que mayor capacidad de predicción ha proporcionado. De esta manera, se ha obtenido que el valor óptimo es de  $b = 0,105$ .

La Figura 5.16 muestra la dispersión del AR (obtenido con los *drivings* óptimos) frente al *gap* (restringido al rango  $[0, 0,4]$  para enfocar en instancias potencialmente problemáticas). Los puntos están coloreados según su valor de HP, donde colores más oscuros (azules/morados) corresponden a estados fundamentales más aislados (HP ba-



**Figura 5.15:** Espectro de la función de coste de dos grafos con un *gap* muy similar. Las líneas azules verticales indican los diferentes estados accesibles y la curva roja es una Gaussiana centrada en el estado que minimiza el coste y con  $b = 0,16$ . La línea horizontal negra indica el punto donde la diferencia del valor de coste es de 0,1. Según nuestra hipótesis, el grafo respectivo al espectro (a) sería fácil de resolver y el grafo de (b) difícil.

jo) y colores brillantes (amarillos/rojos) indican mayor densidad de estados cercanos (HP alto).



**Figura 5.16:** Dispersión de AR alcanzado con los *drivings* óptimos frente al *gap*. Se muestran las instancias en el rango de [0, 0,4] del *gap*. Cada punto se ha coloreado en función de su valor del HP. Se ha utilizado un valor del hiperparámetro  $b = 0,1809$ .

La Figura 5.16 muestra la dispersión de AR frente al *gap* para el caso de los *drivings* óptimos. Las instancias se han coloreado según su valor de HP, calculado para  $b = 0,1809$  y se muestra únicamente el rango  $[0, 0,4]$ , donde se encuentran las instancias más interesantes. Se ha escogido un valor de HP que ilustre de una manera clara la diferencia de espectro de grafos.

Las instancias coloreadas con granate son las que tienen el *gap* más grande, por lo que el estado fundamental está más aislado. Después, se encuentran las instancias coloreadas por azul oscuro, que son las que tienen un estado cercano al óptimo. Estos son los que forman el triángulo invertido mencionado anteriormente. Una explicación a este fenómeno podría ser que, cuando el *gap* se va haciendo menor que 0,2, aumenta la probabilidad de que el sistema acabe en el estado subóptimo, por lo que el AR disminuye. No obstante, llega un punto en el que a pesar de que esto suceda, el estado

subóptimo se encuentra tan cerca al óptimo que el AR vuelve a aumentar.

Los colores a partir del azul claro indican la presencia de más de un estado con un coste similar al estado fundamental. Observamos que estos son, en general, los que peor resultado proporcionan en comparación con otros grafos con un mismo *gap*.

El análisis de la Figura 5.16 revela que, aunque existe una tendencia, la relación entre el HP y AR no es determinista. Por ejemplo, se observa una instancia con *gap*  $\approx 0,02$  con un AR  $\approx 1$ , y sin embargo, tiene un HP relativamente alto (HP  $\approx 3,8$ ). Según la hipótesis inicial, este grafo debería ser difícil de resolver. El análisis de su espectro (verificable en el Dashboard) confirma la presencia de estados cercanos al fundamental. Esto demuestra que, aunque el HP captura información relevante sobre la estructurapectral cercana al estado fundamental, esta información por sí sola no es suficiente para predecir de manera concluyente el rendimiento de la simulación, el cual también depende en gran medida de los protocolos de *drivings* escogidos. No obstante, como se verá en la siguiente subsección, el HP sí resulta ser una variable útil para los modelos de clasificación.

### 5.5.2. Entrenamiento del clasificador

Para profundizar en la comprensión de qué características influyen en el rendimiento final de la simulación, se desarrolló un clasificador binario. El objetivo es predecir si una instancia de grafo será resuelta con alta probabilidad de éxito o no.

Primero, se estableció un criterio para definir una “resolución correcta”: se etiquetaron como ‘resueltas correctamente’ (clase negativa) aquellas instancias que alcanzaron un AR  $> 0,999$  con los *drivings* óptimos. Este umbral se eligió tras analizar el histograma de los valores de AR, y resultó en que aproximadamente el 72 % de las instancias pertenecen a la clase positiva.

Como se ha comentado, se exploraron dos conjuntos de variables diferentes, unas relacionadas con las características del espectro de la función de coste, y las otras con las propiedades de los grafos. Por su versatilidad y su explicabilidad, se escogió el RandomForest como modelo a entrenar. En un estudio preliminar también se utilizó el XGBoost pero, al observar que el RandomForest proporcionaba resultados mejores se decidió descartar el XGBoost. El proceso de entrenamiento y evaluación se realizó de la siguiente manera:

- Se dividió el conjunto de datos en entrenamiento (80 %) y test (20 %).
- Se utilizó GridSearchCV de scikit-learn para realizar una búsqueda de hiperparámetros óptimos mediante validación cruzada de 5 *folds*.
- Debido al desequilibrio de clases (72 % vs 28 %), se seleccionaron los modelos que maximizaban la métrica F1-score, la cual ofrece un balance entre precisión y sensibilidad, siendo más informativa que el ACC en casos de desbalanceo.

### Modelos con variables predictoras de la función de coste

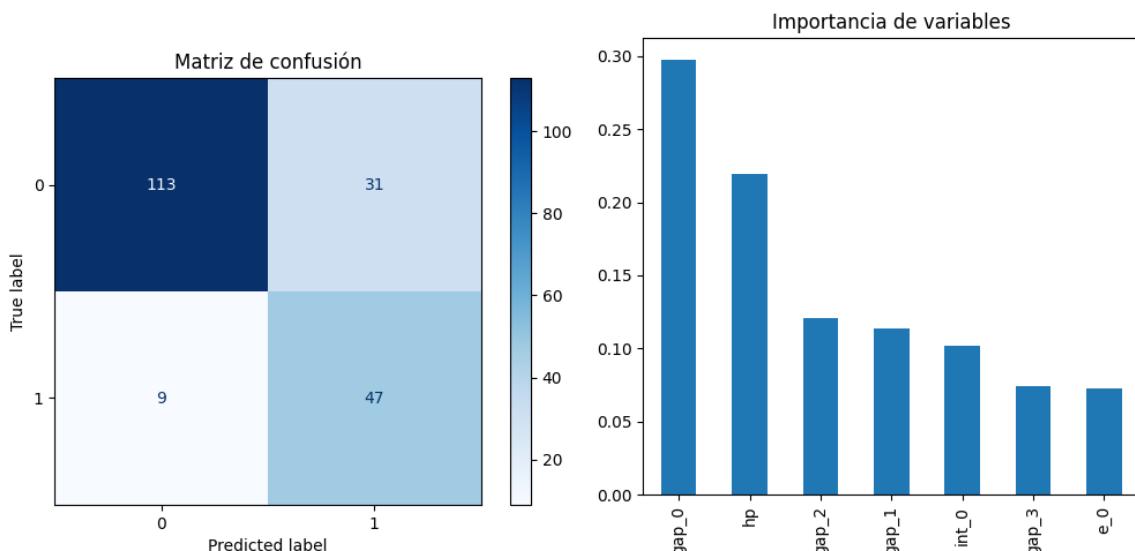
En vista de los resultados obtenidos hasta el momento, se sospechaba que estas variables iban a ser las más relevantes, por lo que se comenzó por su estudio. Las variables escogidas fueron:

- **gap\_0, gap\_1, gap\_2, gap\_3:** Diferencia de coste entre el estado óptimo y los primeros cuatro subóptimos.
- **e\_0:** Coste del estado fundamental.
- **hp:** Valor del *Hardness Parameter* (HP) calculado mediante la Ecuación (5.2).
- **int\_0:** Número de estados con un *gap* relativo al fundamental menor que 0,2.

En primer lugar se entrenó un modelo utilizando únicamente **gap\_0** como variable predictora. De esta manera, este serviría como línea de base, al ser el modelo más simple posible con las variables consideradas. Los resultados obtenidos por este modelo fueron: AUC=84,89 %, F1=68,00 % y ACC=76,00 %. Vemos que a pesar de que el ACC no mejore mucho con respecto a un modelo nulo, el AUC muestra que sí tiene cierta capacidad discriminatoria entre las dos clases.

Al considerar todas las variables mencionadas, se encontró que el mejor modelo se obtuvo con 200 estimadores y una profundidad máxima de 6. De esta manera, los resultados mejoraron a: AUC=87,39 %, F1=70,15 % y ACC=80,00 %.

La Figura 5.17 muestra la matriz de confusión y la importancia de las variables para este modelo óptimo.



**Figura 5.17:** Matriz de confusión (izquierda) e importancia de las variables predictoras (derecha) del mejor modelo encontrado con el conjunto de variables predictoras espectrales.

El análisis de importancia de variables (Figura 5.17, derecha) indica que **gap\_0** es, con diferencia, la variable más relevante, seguida por el **hp**. Los *gaps* con el resto estados subóptimos (**gap\_1, gap\_2** y **gap\_3**) y **e\_0** tienen una importancia considerablemente menor. La matriz de confusión (Figura 5.17, izquierda) indica que el modelo

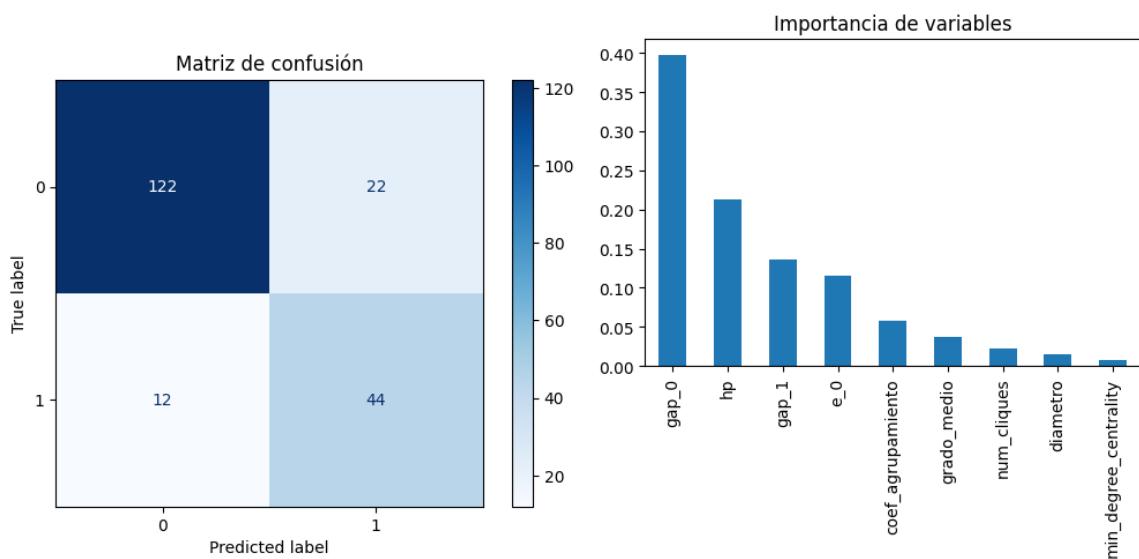
tiene una buena sensibilidad (tasa de verdaderos positivos) del 83.93 %, identificando correctamente una alta proporción de las instancias 'resueltas correctamente'.

### Incorporación de características topológicas del grafo

A continuación, se investigó si añadir propiedades estructurales del grafo mejoraba el rendimiento del clasificador. Se extrajeron diversas métricas topológicas utilizando la librería NetworkX. Inicialmente se obtuvieron 29 variables. Para reducir la redundancia, se calculó la correlación de Spearman entre ellas y se eliminaron aquellas con una correlación superior a 0.8, resultando en 10 variables.

Después, estas nuevas variables se añadieron al conjunto de variables del espectro y se realizó el entrenamiento del modelo Random Forest con el conjunto de variables completo.

Los resultados obtenidos fueron prácticamente idénticos a los del modelo que usaba solo variables espetrales: AUC=87,88 %, F1=72,13 % y ACC=83 %. El diagrama de importancia de variables para este modelo (Figura 5.18, derecha) reveló que las características topológicas añadidas se encontraban entre las menos relevantes, confirmando que aportan poca información adicional para la predicción en este contexto.



**Figura 5.18:** Matriz de confusión (izquierda) e importancia de las variables predictoras (derecha) del mejor modelo encontrado con el conjunto de variables predictoras completo.

Por último, se entrenó un modelo únicamente con estas variables topológicas, el cual obtuvo un AUC=51,85 %, reflejando la poca capacidad predictiva de estas.

### Conclusiones del análisis de clasificación

En resumen, se ha tratado de entrenar un clasificador binario para predecir el éxito de la simulación ( $AR > 0,999$ ), el cual nos ha permitido estudiar las variables que más influyen en el resultado. Dentro de los dos grupos de variables estudiados, las variables

relacionadas con las soluciones de la función de coste son las que mejor resultado han proporcionado, siendo el *gap* y HP las variables más relevantes. Las características topológicas del grafo, al menos las estudiadas en este trabajo, han mostrado poca capacidad predictiva en comparación con las variables espectrales.

**Tabla 5.4:** Resultados de los modelos más relevantes que se han entrenado. Se ha resaltado el modelo que mayor rendimiento ha proporcionado.

	AUC	F1-score	ACC
Modelo nulo	0,5000	0,4375	0,28
Modelo Baseline	0,8489	0,6800	0,76
Modelo variables espectrales	0,8739	0,7015	0,80
<b>Modelo variables completo</b>	<b>0,8788</b>	<b>0,7213</b>	<b>0,83</b>

En la Tabla 5.4 se resumen los resultados de los diferentes modelos que se han entrenado. Vemos que a pesar de que añadir las variables topológicas no aporte demasiada información, ayuda a mejorar el rendimiento del modelo, haciendo del modelo con el conjunto de variables completo el que mayor rendimiento ha obtenido.

Analizando los resultados de este modelo vemos que aunque muestre una capacidad predictiva razonable ( $AUC \approx 87\%$ ), existe margen de mejora. Es fundamental recordar que el rendimiento de la simulación no solo depende de las características estáticas del grafo y su espectro, sino también de los protocolos *drivings*. Por tanto, no es sorprendente que un clasificador basado únicamente en propiedades estáticas de los grafos no alcance una precisión perfecta.

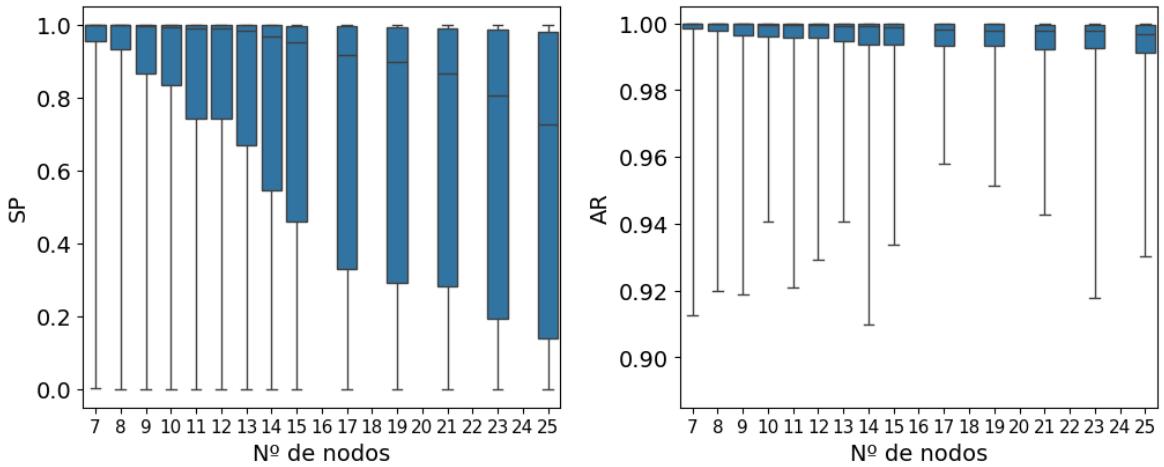
No obstante, en este análisis se ha observado que ciertas propiedades espectrales (especialmente *gap* y HP) están correlacionadas con la dificultad intrínseca de las instancias para el MWIS, abriendo vías para futuros estudios sobre la clasificación de la dificultad de problemas de optimización combinatoria.

## 5.6. Simulación de grafos de mayor tamaño

Finalmente, para evaluar la capacidad de generalización de los *drivings* obtenidos en la Sección 5.2.3, se realizaron simulaciones sobre diferentes conjuntos de 1000 grafos. Cada conjunto está formado por grafos con un número de nodos específico, variando desde 7 hasta 25.

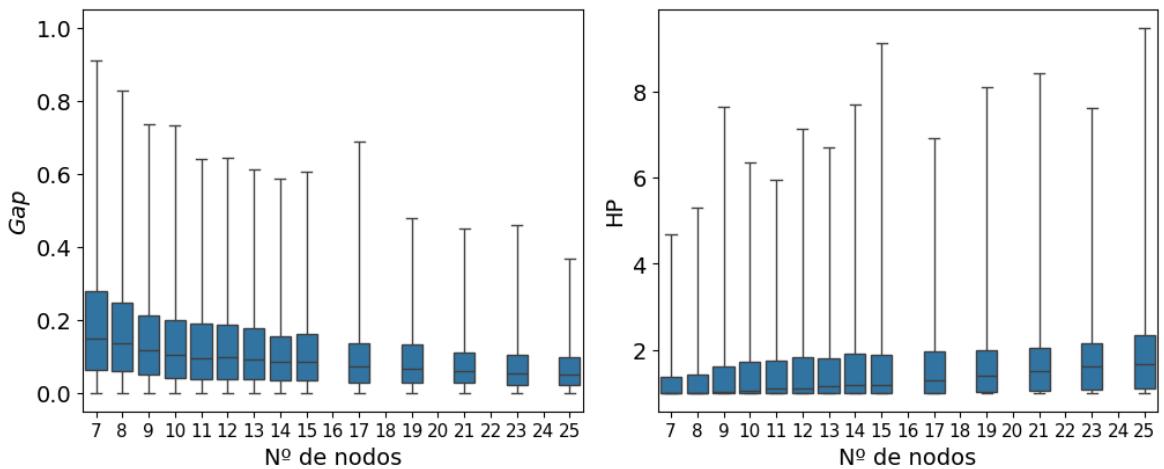
La Figura 5.19 presenta, mediante diagramas de caja, cómo varían las distribuciones del SP y del AR al aumentar el tamaño de los grafos. Como era de esperar, se observa una reducción progresiva del rendimiento a medida que aumenta el tamaño de los grafos. No obstante, destaca el hecho de que el 84,6 % de las instancias alcanzan un AR superior al 99 %, lo cual representa un resultado altamente satisfactorio.

Por otro lado, la Figura 5.20 muestra la evolución de las distribuciones del *gap* y del HP en función del número de nodos. Se aprecia que al aumentar el tamaño de los grafos, la distribución del *gap* tiende a concentrarse hacia valores más bajos. Este



**Figura 5.19:** Diagrama de cajas del SP (izquierda) y AR (derecha) en función del número de nodos del conjunto de grafos.

resultado es esperable, ya que el incremento en el tamaño del grafo implica un aumento exponencial del número de soluciones posibles, incrementando así la probabilidad de encontrar configuraciones con valores de coste muy cercanos al óptimo.



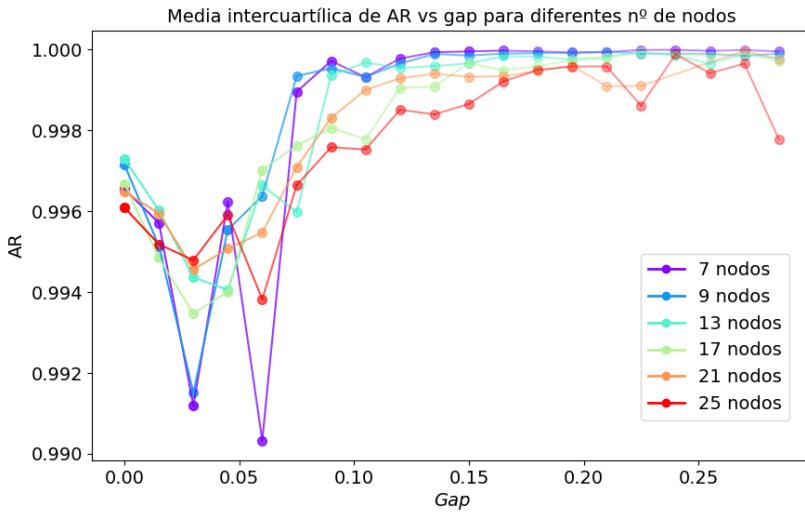
**Figura 5.20:** Diagrama de cajas de las variables *gap* y HP en función del número de nodos del conjunto de grafos.

La Figura 5.20 (derecha) indica, como era de esperar, un aumento en los valores del HP con el número de nodos, dada la estrecha relación existente entre HP y *gap*.

Adicionalmente, podemos analizar los diagramas de dispersión del SP y AR en función del *gap* para distintos tamaños de grafo. En la Figura 5.21 se presentan estos resultados para grafos de 10, 17 y 25 nodos, utilizando la misma escala en los ejes para facilitar comparaciones. Tal y como anticipaba la Figura 5.20, al incrementar el número de nodos, los valores del *gap* se concentran en valores menores.

No obstante, la gráfica de AR frente al *gap* de la Figura 5.21 muestra que, los valores mínimos de AR permanecen constantes independientemente del tamaño del grafo. Esto sugiere que la reducción observada en el AR promedio (Figura 5.19, derecha) al aumentar el tamaño de los grafos está relacionada principalmente con la concentración

ción de valores bajos del *gap*. En otras palabras, el deterioro del rendimiento promedio podría atribuirse más al hecho de que los grafos se concentran en regiones del *gap* donde es más difícil alcanzar buenos resultados, que al incremento del número de nodos *per se*.



**Figura 5.22:** Relación entre el *gap* y AR para distintos números de nodos. La variable *gap* ha sido discretizada en intervalos. Para cada intervalo y cada valor del número de nodos, se ha calculado la media de AR considerando únicamente los valores dentro del rango intercuartílico (IQR). Los resultados se representan mediante puntos unidos por líneas, con un color distinto para cada número de nodos. La transparencia de cada punto refleja la cantidad de muestras disponibles en ese intervalo para ese número de nodos.

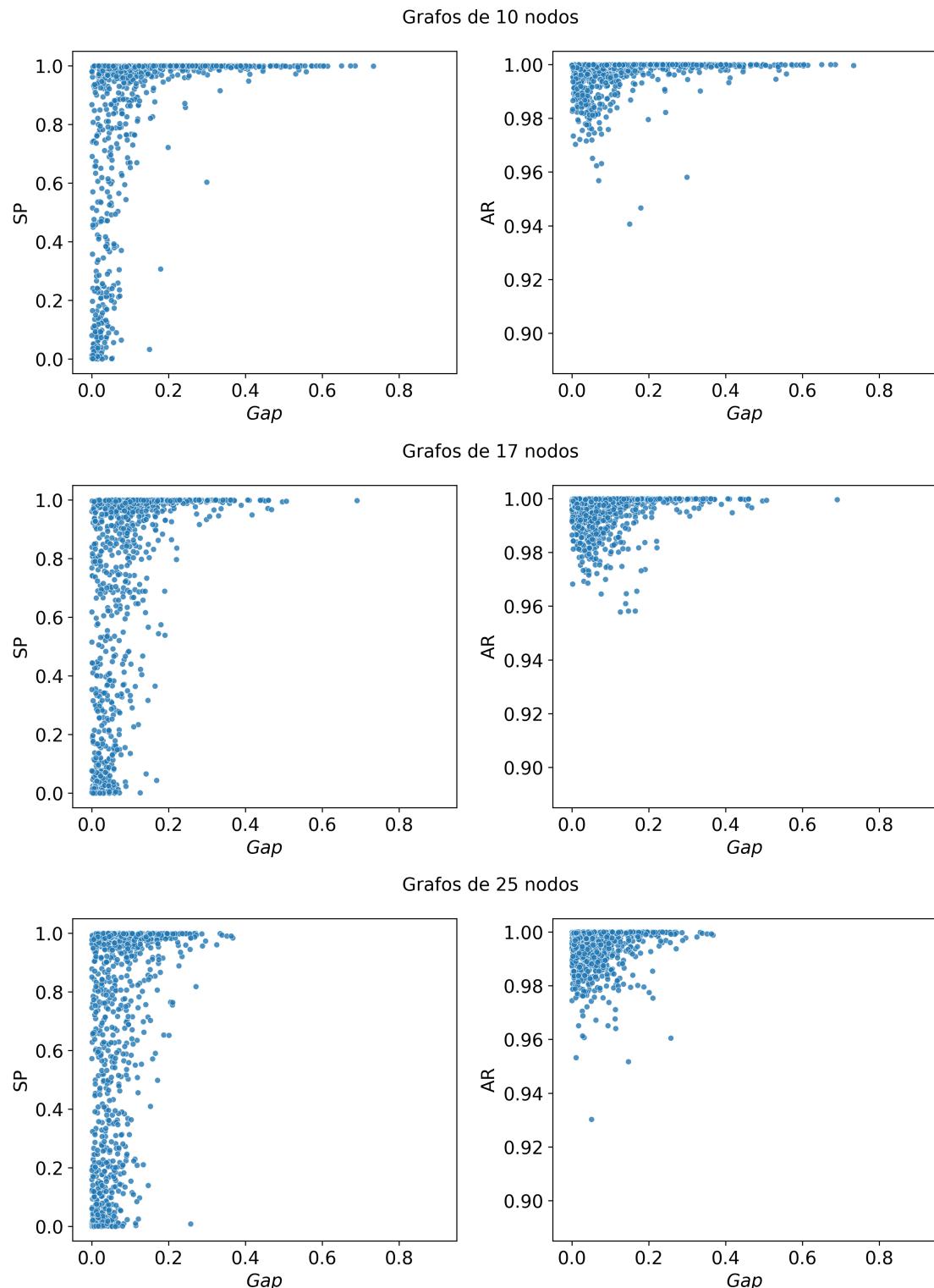
La Figura 5.22 muestra la relación entre el *gap* y el AR para diferentes valores del número de nodos. Se ha discretizado la variable *gap*, calculando así la media del rango intercuartílico para cada intervalo y número de nodos. La razón por la cual se ha decidido calcular esta y no la media común es que la media es muy sensible a valores alejados, lo cual dificultaba la comprensión de la gráfica debido a la fluctuación de los resultados. Asimismo, cada punto se muestra con una opacidad dependiente del número de cuentas en cada intervalo, mostrando la robustez de cada media calculada. En la región de valores de *gap* más altos, vemos que la opacidad es menor, por lo que se han considerado un menor número de puntos, haciendo de este cálculo más sensible a valores lejanos.

Como se observa, el rendimiento para cada conjunto de grafos en función del *gap* es similar, mostrando valores más altos para *gaps* mayores y con una bajada al disminuir el valor de este. Resulta notable la presencia de un valle consistente en diferentes tamaños de grafo, fenómeno también observado en representaciones previas de AR frente a *gap*.

Al comparar los conjuntos extremos (7 y 25 nodos), sorprendentemente los grafos de mayor tamaño presentan mejores resultados en algunas instancias para valores de *gap* bajos, fortaleciendo la idea de que los *drivings* optimizados mantienen su efectividad incluso al aumentar significativamente el número de nodos.

En conclusión, la evaluación realizada sobre grafos de distintos tamaños demues-

tra una reducción progresiva del rendimiento al aumentar el número de nodos, atribuible principalmente al hecho de que los grafos más grandes presentan mayor densidad en regiones de *gap* bajo, los cuales han mostrado ser más difíciles de resolver. No obstante, los *drivings* optimizados presentan una robusta capacidad de generalización, manteniendo un rendimiento notablemente alto, con valores de AR superiores al 99 % en la mayoría de las instancias. Este análisis destaca la importancia del *gap* como indicador fundamental de dificultad y abre la puerta a futuras investigaciones sobre estrategias adaptativas que aborden específicamente grafos con *gaps* bajos para mejorar el rendimiento global.



**Figura 5.21:** Diagrama de dispersión del SP (izquierda) y AR (derecha) en función del *gap* para grafos de 10, 17 y 25 nodos.

---

# 6. Conclusiones y líneas de futuro

---

## 6.1. Resumen de resultados

En este trabajo se ha abordado la resolución del problema *Maximum Weighted Independent Set* (MWIS) mediante simulaciones de computación cuántica en un procesador basado en átomos neutros. El enfoque utilizado busca aprovechar las propiedades físicas de los sistemas cuánticos para resolver problemas de optimización combinatoria que resultan intratables mediante métodos clásicos. Estos algoritmos se encuentran dentro de los avances recientes en computación cuántica, siendo tecnologías prometedoras por su escalabilidad y resultados frente a otros modelos cuánticos universales [56].

A lo largo del proyecto se han llevado a cabo diferentes estrategias de optimización que maximicen la probabilidad de alcanzar la solución óptima, y con ello, la adiabaticidad del proceso. Mediante estos resultados, se ha podido estudiar la influencia que diferentes protocolos tienen en los resultados.

En cuanto a los algoritmos de búsqueda de hiperparámetros empleados, el *grid search* es el método que más información aporta. No obstante, en el estudio de los parámetros  $[a_{\text{global}}, b_{\text{global}}, a_{\text{local}}, b_{\text{local}}]$  más de 50.000 evaluaciones han sido necesarias, por lo que resulta inviable escalar el algoritmo a grafos de mayor tamaño. El tiempo de ejecución necesario fue de 89 horas, repartiendo la carga de trabajo en 4 tareas, cada una paralelizando en 32 CPUs.

Por otro lado, se ejecutó la optimización COBYLA en el servidor para comparar los tiempos de ejecución. Con 110 iteraciones fue suficiente para alcanzar resultados competitivos, ejecutándose en tan solo 26 minutos con 32 CPUs. No obstante, en los resultados de la optimización por grafo se observó que es sensible a los mínimos locales, indicando que su uso es más apropiado cuando se estudia el rendimiento promedio de varios grafos. Por último, el BO ofrece un balance entre coste computacional y capacidad de encontrar el mínimo global. Con tan solo 800 iteraciones y 3 horas de ejecución ha sido capaz de encontrar el mínimo local, y a resultado un algoritmo versátil que permite controlar la exploración frente a la explotación. En este caso, también se hizo uso de 32 CPUs, al igual que con el algoritmo COBYLA.

Entre los resultados más relevantes, destaca la identificación de un conjunto de *drivings* generales (Tabla 5.3), capaces de resolver correctamente una amplia variedad de grafos con un rendimiento muy alto (AR promedio del 99,65 % para 14.000 grafos de entre 7 y 25 nodos).

Asimismo, se ha estudiado la manera en la que diferentes parámetros de los *drivings* influyen en el rendimiento, donde se ha visto que un amplio rango de valores de estos son capaces de producir buenos resultados. Esto demuestra la gran versatilidad de la parametrización escogida. Además, el *grid search* realizado ha permitido entrenar un modelo de regresión que modela el AR promedio obtenido en 1000 grafos en función de los parámetros escogidos, obteniendo un  $R^2 = 0,998$ .

Por otro lado, se ha llevado a cabo un estudio sobre la estructura espectral de

la función de coste, así como de otras propiedades topológicas del grafo, identificando que la proximidad de coste entre el estado óptimo y otros estados cercanos dificulta la convergencia del problema. A partir de este análisis, se ha propuesto una nueva métrica que capta la estructura de los estados cercanos al óptimo. La definición de esta nueva variable denominada como *Hardness Parameter* (HP) se inspira en artículos anteriores donde se aplicaba para el caso del *Maximum Independent Set*.

Asimismo, se ha entrenado un modelo de clasificación supervisado capaz de predecir con una precisión razonable, si un grafo será fácil o difícil de resolver para el sistema cuántico simulado. Se ha probado a alimentar el modelo con diferentes conjuntos de variables, siendo las relacionadas con el espectro de la función de coste las más informativas. La influencia que el HP y el *gap* tienen sobre el resultado, junto con el estudio de optimización por grafo realizado, podrían indicar una posible nueva línea de investigación en la selección anticipada de protocolos optimizados por instancia.

Por último, se ha estudiado la capacidad de generalización de los *drivings* óptimos encontrados, simulando conjuntos de grafos de 1000 instancias con diferentes número de nodos, observando un descenso del rendimiento con el aumento del número de nodos del conjunto. No obstante, este fenómeno parece estar más relacionado con el hecho de que la distribución del *gap* en los conjuntos con más nodos se concentra en regiones donde la resolución del problema es más difícil, y no tanto con el hecho de que los *drivings* funcionen peor.

## 6.2. Contribuciones principales

A lo largo del desarrollo de este trabajo se han obtenido diversos resultados de interés, tanto desde el punto de vista experimental como conceptual. A continuación se resumen las principales contribuciones:

- Optimización global de *drivings* que permiten alcanzar valores altos de *Approximation Ratio* (AR) de forma consistente en múltiples grafos. Se ha logrado identificar una parametrización general robusta, válida para una amplia variedad de instancias sin necesidad de ajuste por grafo.
- Generalización del protocolo a través de un modelo de regresión, capaz de predecir el rendimiento del sistema en función de los parámetros que definen los *drivings*. Este modelo facilita la identificación de combinaciones efectivas sin necesidad de recurrir a simulaciones costosas, lo cual podría resultar de gran ayuda en el caso de necesitar una optimización individualizada.
- Definición de una métrica estructural (*Hardness Parameter*, HP) que cuantifica la densidad espectral en torno al estado óptimo para la función de coste del MWIS. Este parámetro podría ser de utilidad a la hora de caracterizar los grafos.
- Entrenamiento de un clasificador supervisado que, a partir de características del grafo, predice si un problema será fácil o difícil para el sistema cuántico considerado. Este modelo permite observar variables que pueden afectar al rendimiento de la simulación, abriendo la posibilidad de cuantificar la dificultad de resolución de diferentes grafos. Esto sería de gran ayuda a la hora de comparar el rendimiento de diferentes algoritmos de resolución del MWIS, tanto clásicos como cuánticos.

- Estudio de la generalización de los *drivings* generales obtenidos en la optimización de grafos de 7 nodos. A pesar de haber restringido la optimización a grafos de un tamaño específico, se ha observado que el rendimiento de este se mantiene en grafos de mayor tamaño. Asimismo, este estudio ha permitido observar que, como era de esperar, los grafos de mayor tamaño tienen una mayor probabilidad de tener un *gap* menor, evidenciando de nuevo la dificultad de este tipo de grafos y la influencia de esta variable sobre el resultado.

Estas contribuciones permiten no solo avanzar en la resolución cuántica del problema MWIS, sino también en el diseño más eficiente y automatizado de protocolos *drivings* en sistemas cuánticos, así como la caracterización de grafos.

## 6.3. Limitaciones del estudio

Aunque los resultados obtenidos en este trabajo son prometedores, existen varias limitaciones que conviene tener en cuenta y que condicionan el alcance de las conclusiones.

- **Tamaño de grafos analizados.** La gran mayoría del trabajo se basa en el análisis de grafos de 7 nodos, lo que ha permitido hacer un estudio bastante satisfactorio. Esto, es debido a la capacidad computacional disponible. Asimismo, no ha sido posible estudiar grafos con más de 25 nodos, lo cual dificulta analizar la escalabilidad de resultados en tamaños mayores.
- **Restricciones en la definición de *drivings*.** Para simplificar el proceso de optimización y hacer viable la comparación entre múltiples instancias, se ha restringido el análisis a una familia concreta de funciones para los *drivings*, controlada por pocos parámetros. Esto limita la flexibilidad del protocolo y puede excluir configuraciones más eficientes que escapan a la parametrización escogida.
- **Estimación del *gap* como medida indirecta de la adiabaticidad.** Aunque el análisis espectral ha proporcionado información valiosa sobre la dificultad del problema, no se ha calculado el *gap* mínimo a lo largo de toda la evolución, que es el factor crítico para garantizar la validez del Teorema de la Adiabaticidad. Esta simplificación puede ocultar ciertas dinámicas no triviales que afectan a la fidelidad de la solución final.
- **Evaluación en simulación idealizada.** Todo el estudio se ha realizado sobre simulaciones numéricas, sin considerar fuentes de error propias del *hardware* físico como la decoherencia o errores de medición. Por tanto, los resultados representan un límite ideal superior y podrían verse perjudicados en implementaciones experimentales reales.

Reconocer estas limitaciones permite contextualizar los resultados y orientar de forma más precisa los esfuerzos futuros hacia una mejora sistemática del protocolo propuesto.

## 6.4. Líneas de investigación futuras

El trabajo desarrollado en este proyecto abre las puertas a múltiples extensiones y mejoras que pueden abordarse en trabajos futuros. A continuación se enumeran algunas posibles líneas de investigación, tanto desde el punto de vista teórico como experimental:

- Buscar una manera de medir “cómo de adiabática” ha sido la evolución del sistema podría ser clave a la hora de evaluar los resultados y seleccionar los protocolos *drivings*. Asimismo, podría aportar una valiosa información sobre diferentes variables clave que influyan en el resultado.
- La parametrización escogida en este trabajo, se basa en el conocimiento físico del problema y en los resultados experimentales. No obstante, pueden existir otras definiciones que puedan superar los resultados mostrados. Finžgar *et al.* (2024), en su artículo, proponen diversas opciones de diferente grado complejidad [52].
- En este trabajo, se han optimizado los *drivings* sin imponer más restricciones que los rangos accesibles por los parámetros. No obstante, una vez analizado las diferentes condiciones necesarias para que la evolución sea adiabática, añadir restricciones que aseguren la adiabaticidad pude resultar de gran ayuda en la implementación de *hardware* real.

Estas líneas representan algunas direcciones prometedoras para continuar el trabajo desarrollado, ya sea profundizando en la comprensión de los mecanismos físicos que afectan la evolución del sistema o mejorando el diseño y control de los protocolos. Avanzar en estos aspectos no solo permitiría aumentar la eficacia y robustez de las soluciones obtenidas, sino también acercar este tipo de simulaciones al ámbito experimental, contribuyendo al desarrollo de estrategias cuánticas prácticas para la resolución de problemas combinatorios complejos.

---

# Bibliografía

---

- [1] Dowling, Jonathan P. y Gerard J. Milburn: *Quantum technology: the second quantum revolution*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 361(1809):1655–1674, 2003. <https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2003.1227>.
- [2] Saffman, M.: *Quantum computing with atomic qubits and Rydberg interactions: progress and challenges*. Journal of Physics B: Atomic, Molecular and Optical Physics, 49(20), oct 2016. <https://dx.doi.org/10.1088/0953-4075/49/20/202001>.
- [3] Loïc Henriet, Lucas Beguin, Adrien Signoles Thierry Lahaye Antoine Georges Olivier Reymond y Christophe Jurczak: *Quantum computing with neutral atoms*. Quantum, 4:327, Septiembre 2020.
- [4] M. Saffman, T. G. Walker y K. Mølmer: *Quantum information with Rydberg atoms*. Rev. Mod. Phys., 82:2313–2363, Aug 2010. <https://link.aps.org/doi/10.1103/RevModPhys.82.2313>.
- [5] Garey, Michael R. y David S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Enero 1979. <http://www.gbv.de/dms/hebis-darmstadt/toc/125242654.pdf>.
- [6] Brent N. Clark, Charles J. Colbourn y David S. Johnson: *Unit disk graphs*. Discrete Mathematics, 86(1):165–177, 1990. <https://www.sciencedirect.com/science/article/pii/0012365X90903580>.
- [7] Alireza Vahdatpour, Foad Dabiri, Maryam Moazeni y Majid Sarrafzadeh: *Theoretical Bound and Practical Analysis of Connected Dominating Set in Ad Hoc and Sensor Networks*. Distributed Computing, 5218:481–495, 2008.
- [8] Pankaj K. Agarwal, Marc van Kreveld y Subhash Suri: *Label placement by maximum independent set in rectangles*. Computational Geometry, 11(3):209–218, 1998, ISSN 0925-7721. <https://www.sciencedirect.com/science/article/pii/S0925772198000285>.
- [9] Semra Ağralı, Z. Caner Taşkın y A. Tamer Ünal: *Employee scheduling in service industries with flexible employee availability and demand*. Omega, 66:159–169, 2017, ISSN 0305-0483. <https://www.sciencedirect.com/science/article/pii/S0305048316000475>.
- [10] Nielsen, Michael A. y Isaac L. Chuang: *Quantum Computation and Quantum Information*. Cambridge University Press, 2010, ISBN 978-1-107-00217-3.
- [11] Karen Wintersperger, Florian Dommert, Thomas Ehmer Andrey Hoursanov Johannes Klepsch Wolfgang Mauerer Georg Reuber Thomas Strohm Ming Yin y Sebastian Luber: *Neutral atom quantum computing hardware: performance and end-user perspective*. EPJ Quantum Technology, 10(21), Agosto 2023, ISSN 2196-0763. <https://doi.org/10.1140/epjqt/s40507-023-00190-1>.

- [12] Gennady P Berman, Gary D Doolen, Ronnie Mainier y iVladimir I Tsifrinovich: *Introduction to Quantum Computers*. World Scientific, Enero 1998.
- [13] Wittek, Peter: *Quantum Machine Learning*. Academic Press, Septiembre 2014, ISBN 978-0-12-800953-6.
- [14] Roland, Jérémie y Nicolas J. Cerf: *Quantum search by local adiabatic evolution*. Phys. Rev. A, 65, Marzo 2002. <https://link.aps.org/doi/10.1103/PhysRevA.65.042308>.
- [15] Constantin Dalyac, Lucas Leclerc, Louis Vignoli Mehdi Djellabi Wesley da Silva Coelho Bruno Ximenez Alexandre Dareau Davide Dreon Vincent E. Elfving Adrien Signoles Louis Paul Henry y Loïc Henriet: *Graph algorithms with neutral atom quantum processors*. Eur. Phys. J. A, 60(9):177, 2024. <https://doi.org/10.1140/epja/s10050-024-01385-5>.
- [16] Shor, P. W.: *Algorithms for quantum computation: discrete logarithms and factoring*. En *Proceedings 35th Annual Symposium on Foundations of Computer Science*, páginas 124–134, 1994. <https://ieeexplore.ieee.org/document/365700>.
- [17] Hannah J. Manetsch, Gyohei Nomura, Elie Bataille Kon H. Leung Xudong Lv y Manuel Endres: *A tweezer array with 6100 highly coherent atomic qubits*. 2024. <https://api.semanticscholar.org/CorpusID:268531555>.
- [18] Jan Balewski, Milan Kornjača, Katherine Klymko Siva Darbha Mark R. Hirsbrunner Pedro L. S. Lopes Fangli Liu y Daan Camps: *Engineering quantum states with neutral atoms*. 2024.
- [19] Hannes Pichler, Sheng-Tao Wang, Leo Zhou Soonwon Choi y Mikhail D. Lukin: *Quantum Optimization for Maximum Independent Set Using Rydberg Atom Arrays*. 2018.
- [20] Wikipedia: *Maximal independent set*, Agosto 2024. [https://en.wikipedia.org/w/index.php?title=Maximal\\_independent\\_set&oldid=1240725767](https://en.wikipedia.org/w/index.php?title=Maximal_independent_set&oldid=1240725767), visitado el 2025-01-24.
- [21] Erciyes, Kayhan: *Distributed graph algorithms for computer network.pdf*. Springer Science Business Media, 1<sup>a</sup> edición, Mayo 2013, ISBN 978-1-4471-5172-2.
- [22] Tomohiro Imanaga, Koji Nakano, Ryota Yasudo Yasuaki Ito Yuya Kawamata Ryota Katsuki Yusuke Tabata Takashi Yazane y Kenichiro Hamano: *Simple iterative trial search for the maximum independent set problem optimized for the GPUs*. Concurrency and Computation: Practice and Experience, 35(14), 2023. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6681>.
- [23] Robson, J.M: *Algorithms for maximum independent sets*. Journal of Algorithms, 7(3):425–440, 1986, ISSN 0196-6774. <https://www.sciencedirect.com/science/article/pii/0196677486900325>.
- [24] Xiao, Mingyu y Hiroshi Nagamochi: *Exact algorithms for maximum independent set*. Information and Computation, 255:126–146, 2017, ISSN 0890-5401. <https://www.sciencedirect.com/science/article/pii/S0890540117300950>.
- [25] *Gurobi Optimizer Reference Manual*, 2015. <https://docs.gurobi.com/projects/optimizer/en/current/index.html>.

- [26] Chuixiong Wu, Jianan Wang y Fen Zuo: *From Maximum Cut to Maximum Independent Set*. 2024.
- [27] Hristo Djidjev, Guillaume Chapuis, Georg Hahn y Guillaume Rizk: *Efficient Combinatorial Optimization Using Quantum Annealing*, Enero 2018. <http://arxiv.org/abs/1801.08653>, visitado el 2025-01-27, arXiv:1801.08653 [quant-ph].
- [28] S. Ebadi, A. Keesling, M. Cain T. T. Wang H. Levine D. Bluvstein G. Semeghini A. Omran J. G. Liu R. Samajdar X. Z. Luo B. Nash X. Gao B. Barak E. Farhi S. Sachdev N. Gemelke L. Zhou S. Choi H. Pichler S. T. Wang M. Greiner V. Vuletic y M. D. Lukin: *Quantum Optimization of Maximum Independent Set using Rydberg Atom Arrays*. Science, 376(6598):1209–1215, 2022. <https://www.science.org/doi/abs/10.1126/science.abo6587>.
- [29] Nguyen, Minh-Thi, Liu Jin Guo Wurtz Jonathan Lukin Mikhail D. Wang Sheng Tao Pichler y Hannes: *Quantum Optimization with Arbitrary Connectivity Using Rydberg Atom Arrays*. Quantum, 4(1), Febrero 2023, ISSN 2691-3399.
- [30] Kato, Tosio: *On the Adiabatic Theorem of Quantum Mechanics*. Journal of the Physical Society of Japan, 5(6):435–439, 1950. <https://doi.org/10.1143/JPSJ.5.435>.
- [31] Wu, Zhaoyan y Hui Yang: *Validity of the quantum adiabatic theorem*. Phys. Rev. A, 72, Jul 2005. <https://link.aps.org/doi/10.1103/PhysRevA.72.012114>.
- [32] Born, M. y V. Fock: *Beweis des Adiabatensatzes*. Zeitschrift für Physik, 51(3):165–180, 1928, ISSN 0044-3328. <https://doi.org/10.1007/BF01343193>.
- [33] I. M. Georgescu, S. Ashhab y Franco Nori: *Quantum simulation*. Rev. Mod. Phys., 86:153–185, Mar 2014. <https://link.aps.org/doi/10.1103/RevModPhys.86.153>.
- [34] Jonathan Wurtz, Alexei Bylinskii, Boris Braverman Jesse Amato Grill Sergio H. Cantu Florian Huber Alexander Lukin Fangli Liu Phillip Weinberg John Long Sheng Tao Wang Nathan Gemelke y Alexander Keesling: *Aquila: QuEra’s 256-qubit neutral-atom quantum computer*, Junio 2023. <http://arxiv.org/abs/2306.11727>, visitado el 2024-11-25, arXiv:2306.11727.
- [35] Edelstein, Atoms S.A. y T.F. Gallagher: *Rydberg*. Volumen 14 de *Advances in Atomic and Molecular Physics*, páginas 365–392. Academic Press, 1979. <https://www.sciencedirect.com/science/article/pii/S0065219908601323>.
- [36] M. D. Lukin, M. Fleischhauer, R. Cote L. M. Duan D. Jaksch J. I. Cirac y P. Zoller: *Dipole Blockade and Quantum Information Processing in Mesoscopic Atomic Ensembles*. Phys. Rev. Lett., 87, Jun 2001. <https://link.aps.org/doi/10.1103/PhysRevLett.87.037901>.
- [37] C. S. Adams, J. D. Pritchard y J. P. Shaffer: *Rydberg atom quantum technologies*. Journal of Physics B: Atomic, Molecular and Optical Physics, 53(1), dec 2019. <https://dx.doi.org/10.1088/1361-6455/ab52ef>.
- [38] Gallagher, T. F.: *Rydberg atoms*. Reports on Progress in Physics, 51(2):143, feb 1988. <https://dx.doi.org/10.1088/0034-4885/51/2/001>.

- [39] A. K. Mohapatra, T. R. Jackson y C. S. Adams: *Coherent Optical Detection of Highly Excited Rydberg States Using Electromagnetically Induced Transparency*. Phys. Rev. Lett., 98, Mar 2007. <https://link.aps.org/doi/10.1103/PhysRevLett.98.113003>.
- [40] Loudon, Rodney: *The Quantum Theory of Light*. Oxford University Press, Septiembre 2000, ISBN 9780198501770. <https://doi.org/10.1093/oso/9780198501770.001.0001>.
- [41] E. Urban, T. A. Johnson, T. Henage L. Isenhower D. D. Yavuz T. G. Walker y M. Saffman: *Observation of Rydberg blockade between two atoms*. Nature Physics, 5(2):110–114, 2009, ISSN 1745-2481. <https://doi.org/10.1038/nphys1178>.
- [42] Martin Lanthaler, Kilian Ender, Clemens Dlaska y Wolfgang Lechner: *Quantum optimization with globally driven neutral atom arrays*, 2024. <https://arxiv.org/abs/2410.03902>.
- [43] Frederiksen, Lillian: *Pulse programming on Rydberg atom hardware*, Mayo 2023. [https://pennylane.ai/qml/demos/ahs\\_aquila](https://pennylane.ai/qml/demos/ahs_aquila), visitado el 2025-02-04.
- [44] Martin Lanthaler, Clemens Dlaska, Kilian Ender y Wolfgang Lechner: *Rydberg-Blockade-Based Parity Quantum Optimization*. Phys. Rev. Lett., 130, May 2023. <https://link.aps.org/doi/10.1103/PhysRevLett.130.220601>.
- [45] Camille Grange, Michael Poss y Eric Bourreau: *An introduction to variational quantum algorithms for combinatorial optimization problems*. 4OR, 21(3):363–403, 2023, ISSN 1614-2411. <https://doi.org/10.1007/s10288-023-00549-1>.
- [46] Preskill, John: *Quantum Computing in the NISQ era and beyond*. Quantum, 2:79, Agosto 2018, ISSN 2521-327X. <https://doi.org/10.22331/q-2018-08-06-79>.
- [47] Choi, Jaeho y Joongheon Kim: *A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications*. En 2019 International Conference on Information and Communication Technology Convergence (ICTC), páginas 138–142, 2019.
- [48] Elijah Pelofske, Andreas Bärtschi y Stephan Eidenbenz: *Quantum Annealing vs. QAOA: 127 Qubit Higher-Order Ising Problems on NISQ Computers*. En Bhatele, Abhinav, Jeff Hammond, Marc Baboulin y Carola Kruse (editores): *High Performance Computing*, páginas 240–258, Cham, 2023. Springer Nature Switzerland.
- [49] Arul Rhik Mazumder, Anuvab Sen y Udayon Sen: *Benchmarking Metaheuristic-Integrated QAOA Against Quantum Annealing*. En Arai, Kohei (editor): *Intelligent Computing*, páginas 651–666, Cham, 2024. Springer Nature Switzerland.
- [50] Finley Alexander Quinton, Per Arne Sevle Myhr, Mostafa Barani Pedro Crespo del Granado y Hongyu Zhang: *Quantum annealing applications, challenges and limitations for optimisation problems compared to classical solvers*. Scientific Reports, 15(1), 2025, ISSN 2045-2322. <https://doi.org/10.1038/s41598-025-96220-2>.
- [51] Leon, Kohei M. Itoh, Dohun Kim Karan K. Mehta Tracy E. Northup Hanhee Paik B. S. Palmer N. Samarth Sorawis Sangtawesin Nathalie P. de y D. W. Steuerman: *Materials challenges and opportunities for quantum computing hardware*. Science, 372, Abril 2021. <https://doi.org/10.1126/science.abb2823>.

- [52] Jernej Rudi Finzgar, Martin J. A. Schuetz, J. Kyle Brubaker Hidetoshi Nishimori y Helmut G. Katzgraber: *Designing quantum annealing schedules using Bayesian optimization*. Informe técnico, 2024.
- [53] Michel Fabrice Serret, Bertrand Marchand y Thomas Ayral: *Solving optimization problems with Rydberg analog quantum computers: Realistic requirements for quantum advantage using noisy simulation and classical benchmarks*. Physical Review A, 102(5), Noviembre 2020, ISSN 2469-9934. <http://dx.doi.org/10.1103/PhysRevA.102.052617>.
- [54] Kapil Goswami, Rick Mukherjee, Herwig Ott y Peter Schmelcher: *Solving optimization problems with local light-shift encoding on Rydberg quantum annealers*. Physical Review Research, 6(2):023031, Abril 2024, ISSN 2643-1564.
- [55] Frazier, P.: *A Tutorial on Bayesian Optimization*. ArXiv, abs/1807.02811, 2018. <https://api.semanticscholar.org/CorpusID:49656213>.
- [56] Albash, Tameem y Daniel A. Lidar: *Adiabatic quantum computation*. Rev. Mod. Phys., 90, Jan 2018. <https://link.aps.org/doi/10.1103/RevModPhys.90.015002>.